Bachelorarbeit 2010

E-Voting Web-Client in JavaScript

Sonam Samkang und Paolo Dorigo

Betreut durch Prof. Dr. Andreas Steffen

Hochschule für Technik Rapperswil

Inhaltsverzeichnis

Abstract

Ein E- Voting System muss einerseits sicher sein andererseits muss es gewährleisten dass jede Wahl berücksichtigt und anonym ausgewertet wird.

Dieses Projekt befasst sich mit der Aufgabe, ein solch sicheres, stabiles und transparentes Wahlsystem anzubieten. Das Ziel der Arbeit ist es, einen Web Client zu entwickeln, der mit Hilfe einer Big- Integer Library und JavaScript Funktionen das für E- Voting geeignete Damgård- Jurik Kryptosystem zu implementiert. Weiter galt es, die Verfügbarkeit von echten Zufallszahlen in JavaScript abzuklären. Die generierten Datensätze sind in der JavaScript Object Notation (JSON) an den Server weiterzuleiten. In einem optionalen Teil der Arbeit sollte die Wohlgeformtheit der verschlüsselten Stimmzettel mittels eines Zero-Knowledge Proofs Protokoll nachgewiesen werden können.

Der verwendete asymmetrische Damgård- Jurik Verschlüsselungsalgorithmus, hat die homomorphe Eigenschaft, dass die Multiplikation chiffrierter Werte gleich der verschlüsselten Summe der entsprechenden Klartexte entspricht. Da nur das aufkumulierte Schlussresultat der Abstimmung entschlüsselt wird, ist dadurch die Anonymität der einzelnen Stimmen gewährleistet. Um die Gültigkeit eines verschlüsselten Wahlzettels zu überprüfen, wird ein Zero-Knowledge Proofs Protokoll verwendet. Da JavaScript nur mit Zahlen bis 53 Bit rechnen kann, ist für die Berechnungen eine JavaScript Big- Integer Library notwendig. Es wurden verschiedene Bibliotheken getestet und im Detail ausgewertet. Ebenfalls wurde die Möglichkeit echte Zufallszahlen plattformunabhängig zu generieren abgeklärt.

Die Wahl der Big-Integer Library ist auf jsbn.js der Stanford University gefallen und für die Generierung von Zufallszahlen auf jscrypto.js, welche durch die Auswertung von Mausbewegungen Entropie gewinnt. Der implementierte JavaScript Client verschlüsselt erfolgreich Stimmzettel mittels Damgård- Jurik Verschlüsselungsalgorithmus und leitet das Chiffrat mit zusätzlichen, für den Zero-Knowledge Proofs Protokoll notwendigen Parametern im JSON Format an den Server weiter, der die empfangenen Daten auf ihre Wohlgeformtheit prüft. Die Messungen haben ergeben, dass der Google Chrome Browser am optimalsten mit den angewendeten Libraries läuft. Bei der Verwendung eines 1024 Bit RSA Modulus benötigt der Google Chrome Browser 1.7 Sekunden für die Damgård- Jurik Verschlüsselung und zusätzlich 8.8 Sekunden für die Generierung der Zero-Knowledge Proofs Protokoll notwendigen Parametern.

Aufgabenstellung

Moderne Electronic-Voting Verfahren (E-Voting) erlauben es jedem Stimmbürger sich zu vergewissern, dass seine Stimme richtig im Endresultat gezählt wurde. Damit gleichzeitig das Stimmgeheimnis garantiert werden kann, wird jede Stimme vor der Abgabe an den Stimmenzählserver mit einem starken kryptografischen Algorithmus verschlüsselt. Im Rahmen dieser Arbeit soll ein Web-Client entwickelt werden, der mit Hilfe einer Big-Integer Library und JavaScript Funktionen das Damgård-Jurik Kryptosystem implementiert, das auf dem bekannten RSA Public Key Verfahren basiert.

- Evaluierung einer geeigneten JavaScript Big-Integer Library Bewertungskriterien: Performance und Funktionsumfang.

- Abklärung der Verfügbarkeit von echten Zufallszahlen unter JavaScript.

- Implementierung der Stimmzettelverschlüsselung mit dem Damgård-Jurik Crypto-system mittels JavaScript. Zielplattform: Mozilla Firefox, falls möglich, weitere Browser, wie Internet Explorer, Chrome, Opera, etc.

- Erstellen eines einfachen Web-basierten GUIs, das Ja/Nein Abstimmungen, sowie Wahlen mit mehreren Kandidaten unterstützt, wobei jeweils nur ein Kandidat gewählt werden darf. Eine Stimmenthaltung durch das Einlegen einer leeren Stimme soll möglich sein.

- Versenden des generierten Datensatzes:
  {Wahl-ID, Benutzer-ID, verschlüsselter Stimmzettel, Zero-Knowledge Proofs} an den Web-Server im JavaScript Object Notation (JSON) Format.

Optional die Wohlgeformtheit der verschlüsselten Stimmzettel soll mittels Zero-Knowledge Proofs für N-te Potenzen nachgewiesen werden.

1 Einführung

Das E-Voting System ist die elektronische Umsetzung der Abstimmung. E-Voting ermöglicht über das Internet von jedem beliebigen Ort abzustimmen. Es gibt zwei Arten von Abstimmungen:

- die Ja / Nein Abstimmung
- die Wahlen mit mehreren Kandidaten.

In beiden Fällen ist eine Enthaltung der Stimme möglich.

Die Stimmberechtigten müssen in der Schweiz folgende Kriterien erfüllen, um abstimmen zu können:

- die Person ist schweizerischer Nationalität;
- die Person hat das 18. Altersjahr zurückgelegt;
- es liegt kein Stimmrechtsausschluss gemäss Art. 136 BV vor;
- die Meldegemeinde ist der politische Wohnsitz der Person (gemäss Bundesgesetz über die politischen Rechte). [1]

Diese Kriterien müssen vom Staat überprüft werden und jedem Stimmberechtigten eine eindeutige Authentifizierung zuweisen.

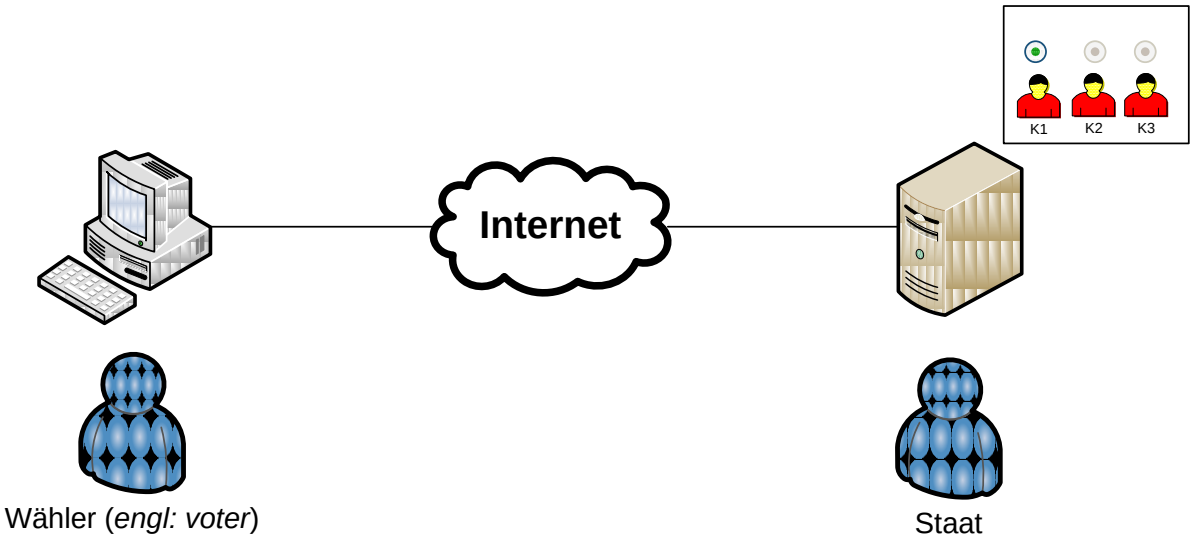


Abbildung 1: Übersicht eines E-Voting Systems

1.1 Motivation

Nachdem die stimmberechtigte Person ihr Stimmzettel abgegeben hat, muss die abgegebene Stimme vom Staat ausgewertet werden. Einerseits hat der Staat die Aufgabe die Stimme auszuwerten ohne den Inhalt zu kennen, damit das Wahlgeheimnis garantiert werden kann. Andererseits soll der Staat die Garantie bekommen, dass der eingereichte elektronische Stimmzettel (*engl.: ballot*) eine gültige Stimme enthält.

Um die bestmögliche Transparents anzubieten wird die Lösung auf der Seite des Clients als Fat-Client JavaScript Lösung implementiert. Der Wähler kann jederzeit den interpretierten JavaScript Code überprüfen und wie seine Stimme clientseitig bearbeitet wird. Der verschlüsselte Stimmzettel wird anschliessen im JSON-Format (JavaScript Object Notation) zum Server geschickt.

Das Wahlgeheimnis wird mit der Damgård-Jurik [8] Verschlüsselung garantiert und die Garantie, dass die Stimme gültig ist mit dem Zero Knowledge Proofs Protokoll [2].

Die Abklärung der Realisierbarkeit dieser JavaScript Lösung, aus Sicherheits- und Performance Aspekte ist Bestand dieser Arbeit. Diese Beinhalten die Überprüfung von Big-Integer Library und die Generierung von echten Zufallszahlen in JavaScript.

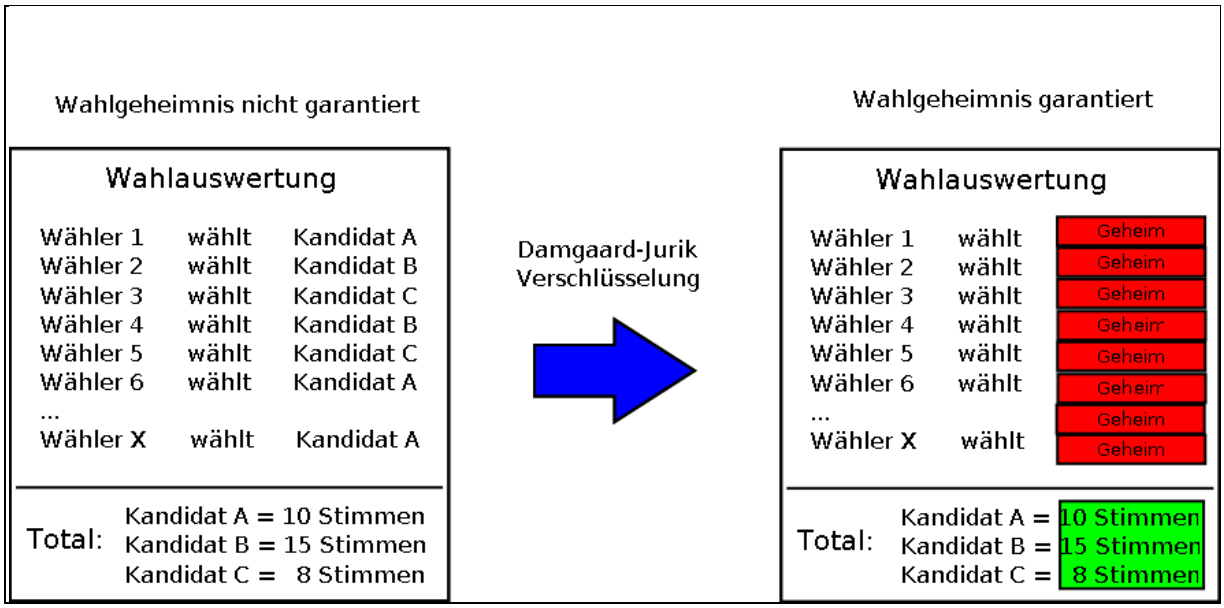


Abbildung 2: Durch die Damgård-Jurik Verschlüsselung garantiert das Wahlgeheimnis

2 Analyse

2.1 Die Wahl der Schlüssellänge

JavaScript kann architekturbedingt nur mit Zahlen bis 53 Bit rechnen [3]. Für Berechnungen mit grösseren Zahlen braucht es eine spezielle Bibliothek. Unter Java gibt es eine Referenz-Implementation namens BigInteger. Für JavaScript gibt es keine offizielle Bibliothek, jedoch gibt es Implementationen von Drittanbietern.

Der rasante Anstieg der Rechenleistung hat zur Folge, dass immer längere Schlüssel verwendet werden müssen, um die Sicherheit der Verschlüsselung zu gewährleisten. Nach dem heutigen Stand der Technik (2010) ist es möglich einen Modulus von 768 Bit Länge in etwa 2 Jahre Rechenzeit zu faktorisieren [4].

Unter Berücksichtigung der Empfehlung von Lenstra und Verheul wird ein Modulus von 1536 Bit getestet, einen guter Kompromiss zwischen Sicherheit und Zeitaufwand für die Verschlüsselung. Eine konservative Schätzung betrachtet diese Schlüssellänge als sicher bis im Jahre 2020 [5].

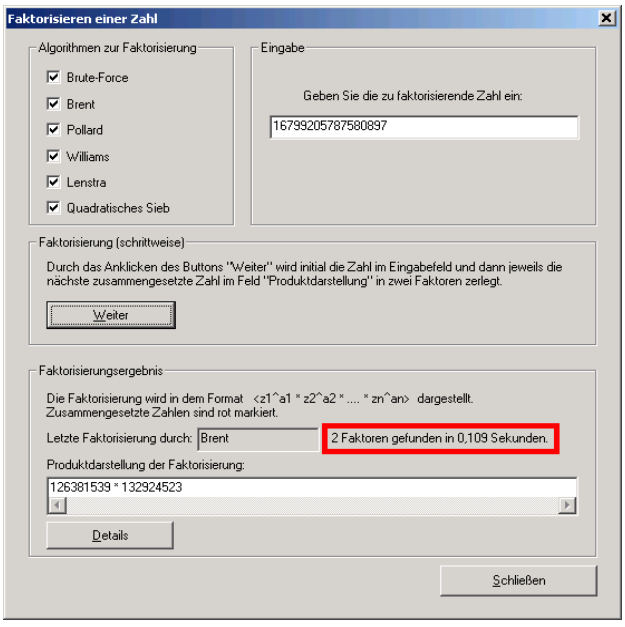


Abbildung 3: Faktorisierung einer 53 Bit Zahl mit Cryptool in 0.1 Sekunden

Das Beispiel in der Abbildung 4 zeigt, dass eine 53 Bit grosse Zahl sich in 0.1 Sekunden auf einem gängigen Desktop Rechner faktorisieren lässt. Eine sichere JavaScript Lösung ohne Big-Integer Library wäre somit nicht realisierbar.

2.2 Rechnen mit grossen Zahlen

Die grösste Zahl, die sich in JavaScript als primitiven Datentypen darstellen lässt beträgt 53 Bit. Wie im Abbildung 4 ersichtlich, ist diese Zahl zu klein um einen kryptografisch sicheren Schlüssel darzustellen. JavaScript kennt, in Gegensatz zu Java, das seit JDK 1.1 diese Bibliothek standardmässig enthält, keine Big-Integer Library. Diese Bibliothek nimmt praktisch beliebig lange Strings oder Byte Arrays entgegen und betrachtet sie als eine numerische Zahl. Die Grösse der berechneten Werte wird lediglich durch den Arbeitsspeicher und die Rechenleistung des Rechners begrenzt.

In dieser Arbeit wurde eine Produktanalyse durchgeführt und den Anforderungen passende Big-Integer Library von Drittanbietern getestet und eingesetzt. Die folgende Beschreibung dient zur Erläuterung der Funktionsweise einer solchen Bibliothek anhand der Potenz-Modulo Funktion.

Die Potenz-Modulo Funktion kann mit der Binäre Exponentiation berechnet werden [6].

Nach dem Prinzip des Teilen und Herrschens (lat. divide et impera) werden die Zwischenergebnisse in verarbeitbaren Einheiten von maximal 32 Bit aufgespaltet (weil die meisten Rechner noch eine 32 Bit Architektur besitzen wäre die Verarbeitung von 53 Bit mit mehreren Taktzyklen verbunden). Die Teile werden in den Speicher abgelegt und am Schluss als Ergebnis wieder zusammengesetzt.

Um das Vorgehen zu erläutern werden im folgenden Beispiel kleine Zahlen verwendet. Gegeben seien drei Zahlen x, y und z, wobei:

- x = 5
- y = 12
- z = 3

Wir betrachten folgende Formel:

$$c = x^y mod\ z$$

Die Zahlen werden eingesetzt:

$$c = 5^{12} mod\ 3 = 1$$

y wird zunächst in die binäre Schreibweise geschrieben:

$$y = 0 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 = 12$$

Da $0 * 2^0 = 0$ und $0 * 2^1 = 0$ sind, so können diese eliminiert werden.

Die Berechnung kann umgeformt werden als:

$$c = 5^{2^2 + 2^3} \, mod \, 3$$

$$c = \left(5^{2^2} \, mod \, 3 * 5^{2^3} \, mod \, 3\right) mod \, 3$$

$$c = (5^4 mod \, 3 * 5^8 mod \, 3) \, mod \, 3$$

$$c = (1 * 1) \, mod \, 3$$

$$c = 1$$

Formel 1: Beispiel einer Binären-Modulo Exponentiation

In der Praxis werden in gängigen Big-Integer Libraries die Montgomery Reduktion [7] für modulare Arithmetik Operationen verwendet.

2.3 Echte Zufallszahlen in JavaScript

Grundsätzlich unterscheidet man zwei Typen von Zufallszahlen, nämlich:

1. Pseudo Zufallszahlen (*engl.: Pseudo Random Number [PRN]*)
2. Echte Zufallszahlen (*engl.: True Random Number [TRN]*)

1. Pseudo Zufallszahlen (PRN)

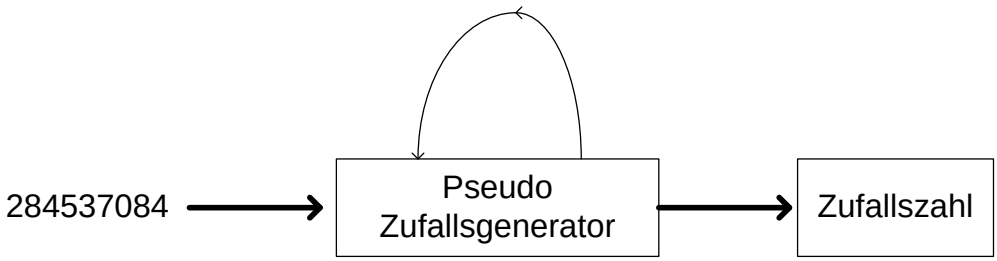


Abbildung 4: Pseudo Zufallszahlengenerator

Die zufälligen Zahlen werden durch deterministische Berechnungen erzeugt.

Prinzip: Man wählt einen Startwert, der sogenannten Saat (*engl.: seed*), zufällig aus und generiert durch rekursive Berechnungsvorschriften die Pseudozufallszahlen.

Die erzeugten Zahlen sehen zwar zufällig aus, sind aber in Wirklichkeit keine. Denn durch die zyklischen Wiederholungen der Pseudozufallszahlenfolge findet man den angewendeten Algorithmus heraus. Aus diesem Grund sind Pseudozufallsgeneratoren (PRNG) ungeeignet für kryptografisch starke Zufallszahlengenerierung. Sie werden hauptsächlich

In JavaScript gibt's die Funktion *Math.random()* um pseudozufällige Zahlen zu erzeugen. Diese Funktion liefert Zahlen im Bereich von 0 und 1.

Als Alternative gibt es die Funktion *window.crypto.random()*, die kryptografisch starke Zufallszahlen aus pseudozufälligen Werten generiert. ist diese Funktion bis zur jetzigen Zeitpunkt in keinem Browser implementiert. Diese Art von Zufallszahlengenerator nennt man Cryptographically Secure Pseudo-Random Number Generator (CSPRNG). Die Voraussetzungen gegenüber PRNG sind dieselben, nur dass für die Sicherheit noch einige zusätzliche Bedingungen erfüllt sein müssen. Die von ihm erzeugte Zahlenfolge sollte für einen Beobachter nicht von einer echten Zufallszahlenfolge unterscheidbar sein. Ausserdem sollte für einen Beobachter sein interner Zustand geheim bleiben, d.h. der Generator stellt eine Black Box dar.

2. Echte Zufallszahlen (TRN)

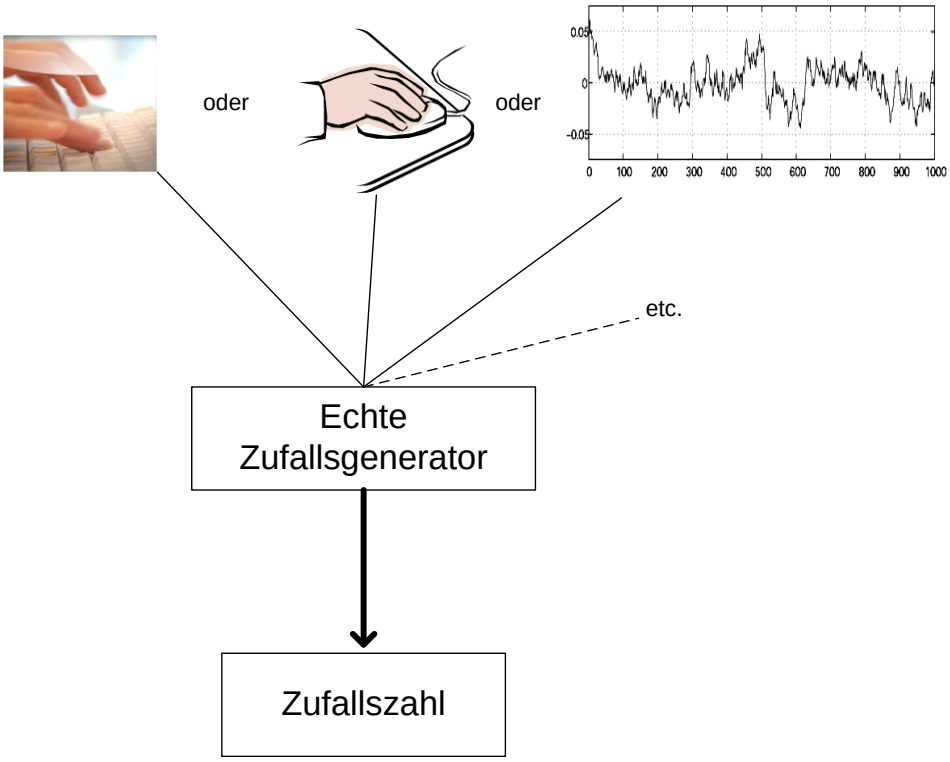


Abbildung 5: Echte Zufallszahlengenerator

Die zufälligen Zahlen werden durch nicht deterministische Berechnungen erzeugt.

Prinzip: Mit Hilfe physikalischer Phänomene werden echte Zufallszahlen erzeugt.
z.B.: Rauschen elektronischer Bauelemente, radioaktive Zerfallsprozesse, quantenphysikalische Effekte oder Sequenzen der Mausbewegung

Die erzeugten Zahlen sind nicht vorhersehbar oder durch einen Algorithmus wie bei den Pseudozufallszahlen berechenbar.

Im JavaScript Standard Library gibt's noch keine Funktion die echte Zufallszahlen generiert.

Unter vielen UNIX- und UNIX-ähnlichen Betriebssystem steht der Zufallsgenerator /dev/random bzw. /dev/urandom zur Verfügung. Dieser Zufallsgenerator sammelt Umgebungsrauschen von Gerätetreibern und andere Quellen in einem Entropiepool. Von diesem Entropiepool werden die Zufallszahlen generiert. Die Qualität der Zufälligkeit weist sehr hohe Qualität, d.h. die erzeugten Zufallszahlen zeigen praktisch keinerlei statische Auffälligkeiten, und sind deshalb ideal für Verschlüsselungsanwendungen.

Es gibt auch andere Möglichkeiten welche hohe Kosten voraussetzt. z.B. Anwendung von Hardware basierten Produkten.

3. PRNG vs. TRNG

Nachfolgende Tabelle zeigt eine Gegenüberstellung der Unterschiede zwischen PRNG und TRNG.

	PRNG	TRNG
Determinismus	deterministisch	nicht deterministisch
Vorhersehbarkeit	vorhersehbar	unvorhersehbar
Periodizität	periodisch	nicht periodisch
Performance	schnell	mittel-schnell
Aufwand	gering	gering-gross
Kosten	keine	günstig-teuer

4. Entscheidung des Zufallszahlengenerators

Für unseren E-Voting Web-Client kommt die Lösung /dev/random bzw. /dev/urandom anzuwenden nicht in Frage, da der Zufallsgenerator die Plattformunabhängigkeit nicht gewährleisten kann. Es bedingt dass die anwendende Person einen UNIX- bzw. UNIX-ähnlichen Betriebssystem benutzt. Deshalb wurde nach einer Alternativlösung gesucht, die diese Bedingung erfüllt gesucht.

Da wir für unseren E-Voting Webclient eine plattformunabhängige und kryptografisch-starke Bibliothek benötigen, haben wir uns für die jsCrypto-Library. Diese Library ist eine Verschlüsselungsbibliothek, entwickelt von den Wissenschaftlern Emily Stark, Mike Hamburg und Dan Boneh von der Stanford University.

Diese symmetrische Verschlüsselungsbibliothek nutzt die AES zur Ver- und Entschlüsselung von Daten sowie Offset Codebook (OCB) und Counter with CBC-MAC (CCM) Moden für authentifizierte Verschlüsselung.

Für die Generierung von Zufallszahlen sammelt die jsCrypto-Library Entropien von verschiedensten Quellen und hashed diesen Wert mittels SHA-256. In der aktuellsten Version der Bibliothek nutzt Sie die Mausbewegungssequenzen als Entropie. Dabei werden folgende Parameter verwendet:

- ⇨ x- und y-Koordinaten und jeweils die aktuelle Uhrzeit in Millisekunden

Diese Werte werden in einem Entropiepool gesammelt. Da der Generator nicht mehr als 256 Bits Entropie bearbeiten kann, verwendet dieser einen „Entropieschätzer“. Dieser stellt sicher dass das maximale Limit nicht überschritten wird. Die gewonnen Entropien werden durch SHA-256 berechnet. Die Ausgabe entspricht die Zufallszahl.

Hier einen Gesamtüberblick über die Eigenschaften von jsCrypto-Library:

	jsCrypto-Library
Determinismus	nicht deterministisch
Vorhersehbarkeit	unvorhersehbar
Periodizität	unperiodisch
Performance	schnell
Aufwand	gering
Kosten	günstig

4. E-Voting Zufallsgenerator Implementierung

Die effektive Implementierung im E-Voting Webclient sieht dann folgendermassen aus:

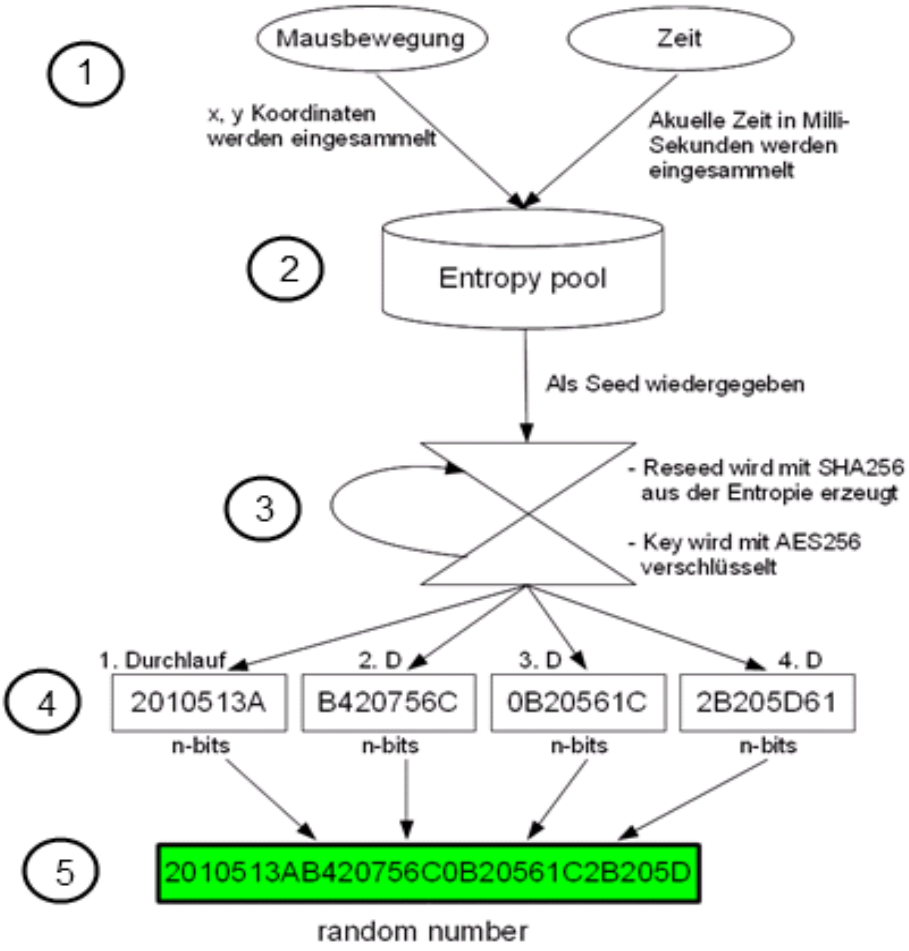


Abbildung 6: E-Voting Zufallszahlgenerator Implementierungsüberblick

Abläufe:

1. Durch die Mausbewegung werden Sequenzen von x- bzw. y- Koordinaten sowie die aktuelle Zeit t in Millisekunden ausgewertet.
2. Die gesammelten Werte gelangt in einem Entropiepool.
3. Ist die gewünschte Menge der Entropie (Default Entropie: 256 Bits) erreicht, werden diese Werte als Initialisierungsvektor (*engl.: seed*) weitergeleitet.
4. Dieser wird dann mittels SHA-256 Hashverfahren berechnet.
5. Bei einem Durchlauf entspricht die Ausgabe 256 Bit. Falls mehr Bit verlangt, durchläuft der Generator mehrmals die Schritte bis die gewünschte Anzahl Bit erreicht werden.
6. Alle Ausgaben werden dann zusammengehängt, welche schlussendlich der effektiven Zufallszahl entspricht.
7. Bei zu vielen Bits werden die überflüssigen Bits nicht verwendet.

2.4 Die Damgård-Jurik Verschlüsselung

Das Damgård-Jurik Kryptosystem [8] ist ein probabilistischer, asymmetrischer Algorithmus, der auf dem Paillier Algorithmus aufbaut. Es konnte bewiesen werden, dass der Modulus mit einem Exponent s erweitert werden kann, ohne die homomorphische Eigenschaft oder die Sicherheit zu beeinträchtigen. Als Generator wird n+1 vorgegeben. Das folgende Beispiel zeigt eine einfache Verschlüsselung.

$$c\,[m,r] = g^m * r^{(n^s)} mod_{n^{(s+1)}}$$

Formel 2: Die Damgård-Jurik Verschlüsselung

Initialisierung

n = Produkt zweier Primzahlen (Modulus) = 135991

g = n + 1 (Generator) = 135992

r = Zufallszahl < n und teilerfremd zu n = 127384

m = Stimme als Zahl kodiert = 2097152

Substitution

$a = g$

$b = m$

$x = r$

$y = n^s$

$z = n^{(s+1)}$

$c = (a^b * x^y)\ mod_z$

Umformung

$$c = (a^b * x^y)\ mod_z = (a^b\ mod_z * x^y mod_z)\ mod_z$$

Auflösung

$$a^{b}_{\mod z} = g^{m}_{\mod n^{(s+1)}} = 135'992^{2'097'152}_{\mod 135'991^{3}} = \underline{76'756'031'652'568}$$

$$X^{Y}_{\mod z} = r^{n^{s}}_{\mod n^{(s+1)}} = 127'384^{135'991^{2}}_{\mod 135'991^{3}} = \underline{1'119'677'084'914'195}$$

$$c = (a_{\mod z} * X^{Y}_{\mod z})_{\mod z} = \underline{641'956'776'186'386}$$

Überprüfung

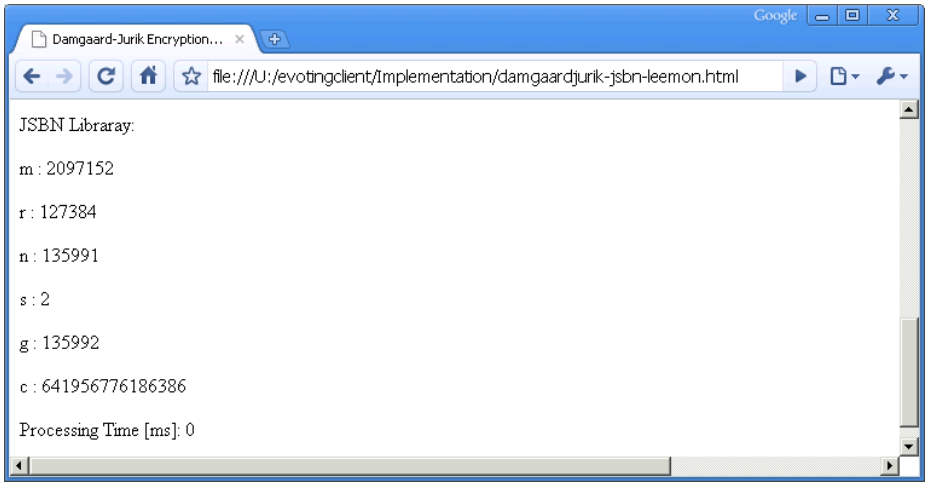


Abbildung 7: Berechnung in Javascript nachgerechnet

2.5 Das Zero Knowledge Proofs Protokoll

Nach einer erfolgreichen Verschlüsselung sendet die stimmberechtigte Person (*engl.: prover*) den Stimmzettel (*engl.: ballot*) zum Server (*engl.: verifier*). Der Verifizierer vergewissert sich, dass der Stimmzettel gültig ist, ohne jedoch ihn entschlüsseln zu müssen, damit das Wahlgeheimnis gewährleistet werden kann. Der Beweiser muss dem Verifizierer beweisen, dass die übermittelten Daten eine n-te Potenz darstellen. Dabei werden folgende 5 Phasen durchgelaufen [2]:

Anmerkung: Das Suffix i in den Variablen bezeichnet die gewählte Stimme (Nachricht), das k die nicht gewählte oder nicht bekannte Stimme.

1. Die Vorbereitung (*engl.: preparation*)

Der Verifizierer veröffentlicht der vom Beweiser erhaltene Stimmzettel c auf einem Anschlagbrett (*engl.: bulletin board*) in Form von u_k für jede Nachricht m_k.

$$uk = \frac{c}{g^{mk}} mod\ n^{s+1}$$

Damit der verschlüsselte Stimmzettel nicht zurückberechnet werden kann, muss der Beweiser, die in der Verschlüsselung verwendete Zufallszahl r maskieren. Aus diesem Grund wählt er einerseits stellvertretend für die geheime Zufallszahl r, für jede gültige Nachricht eine neue Zufallszahl z aus, wobei z teilerfremd zu n sein muss, andererseits für jede gültige Nachricht eine neue Zufallszahl e, wobei e teilerfremd zu p und q sein muss.

2. Die Verpflichtung (*engl.: commitment*)

Der Beweiser berechnet das Commitment für die Nachricht, die er gewählt hat. Dabei maskiert er das geheime r mit der Zufallszahl z_i.

$$ai = zi^{n^s} mod\ n^{s+1}$$

Für alle anderen Nachrichten, die er nicht gewählt hat, wird a_k rückwärts konstruiert.

$$a_k = \frac{{z_k}^{n^s}}{{u_k}^{e_k}} \ mod\ n^{s+1}$$

3. Die Aufforderung (*engl.: Challenge*)

Nachdem der Verifizierte das Commitment erhalten hat, berechnet er eine zufällige Challenge es. Die Challenge es wird als gebildet, wobei b eine zufällige, teilerfremde Zahl zu p und q ist.

4. Die Antwort (*engl.: response*)

Der Beweiser berechnet für die Nachricht, die er gewählt hat, die Differenz zwischen der Challenge es und alle im Commitment verwendeten e aus. Aus diesem Grund kann er sie nicht mehr unbemerkt nachträglich verändern. Die response e wird folgendermassen berechnet:

$$e_i = es - \sum e_k \ mod \ 2^b$$

Für die verwendete Nachricht wird die geheime Zahl r mit z_i maskiert, für alle anderen Nachrichten bleiben die z_k unverändert. Die response z wird folgendermassen berechnet:

$$\mathrm{zi} = \mathrm{zi} * \mathrm{r}^{\mathrm{ei}} \bmod \mathrm{n}$$

5. Die Überprüfung (*engl.: verification*)

Der Verifizierer führt nun zwei Überprüfungen durch. Einerseits muss die Summe aller empfangenden e_k mit der Challenge übereinstimmen, andererseits müssen alle z_k eine Potenz sein.

Überprüfung e_k:

$$\sum e_k \equiv es \ mod \ 2^b$$

Überprüfung z_k:

$$z_k{}^{n^s} \equiv a_k * u_k{}^{e_k} \ mod \ n^{s+1}$$

Falls die Überprüfung wahr ist, so kann mit hoher Wahrscheinlichkeit angenommen werden, dass der Beweiser seine Stimme richtig abgegeben hat.

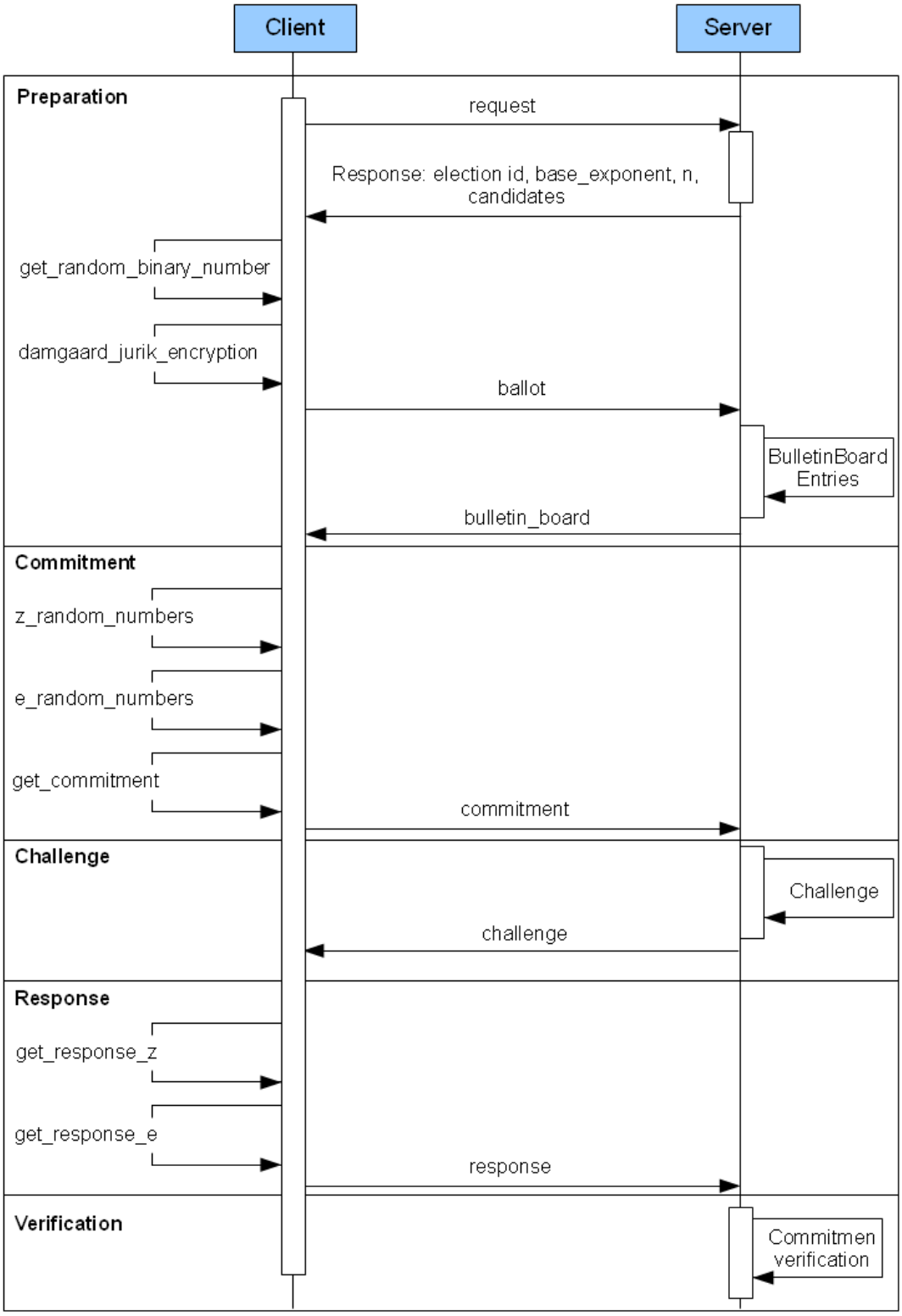


Abbildung 8: InteraktionsdiagrammVerschlüsselung und Zero-Knowledge-Proofs Protokoll

Non Interactive Zero Knowledge Proofs

Die Anzahl der ausgetauschten Nachrichten lasst sich von 3 auf 1 reduzieren, indem der Beweiser die Challenge es nicht vom Verifizier bekommt, sondern in Abhängigkeit der Election Id, Voter Id, der verschlüsselten Nachricht und des Commitments berechnet [9].

Dafür wird ein Hash-Wert aller Parameter mit der SHA256 Funktion berechnet. Daraus ergibt sich eine 256 Bit lange Zahl. Diese Zahl wird als Schlüssel für eine AES Verschlüsselung im Counter Mode mit dem Initialisierungsvektor 0 bis n verwendet, wobei n = b / 128 ist. Zur Erinnerung: b eine zufällige, teilerfremde Zahl zu p und q.

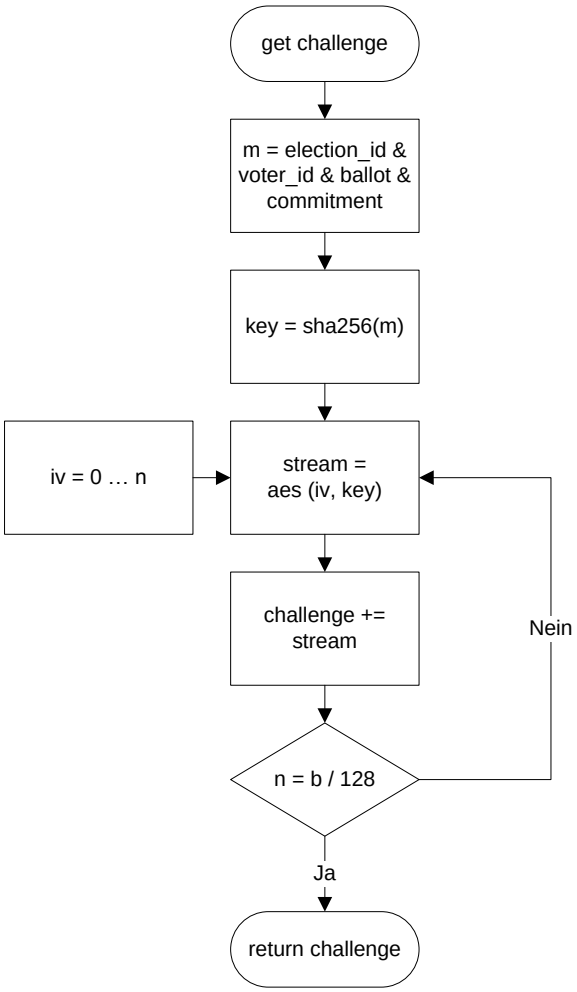


Abbildung 9: Flussdiagramm der Bildung der Challenge mit SHA256 und AES

2.6 Die Wahl der zu testenden Browser

Für diese Arbeit werden die gängigsten Web Browser gemäss der Statistik von Webhits.de berücksichtigt, diese Daten wurden unter 10000 WebHits-Abonnenten ermittelt.

(Stand 24. März 2010) [10].

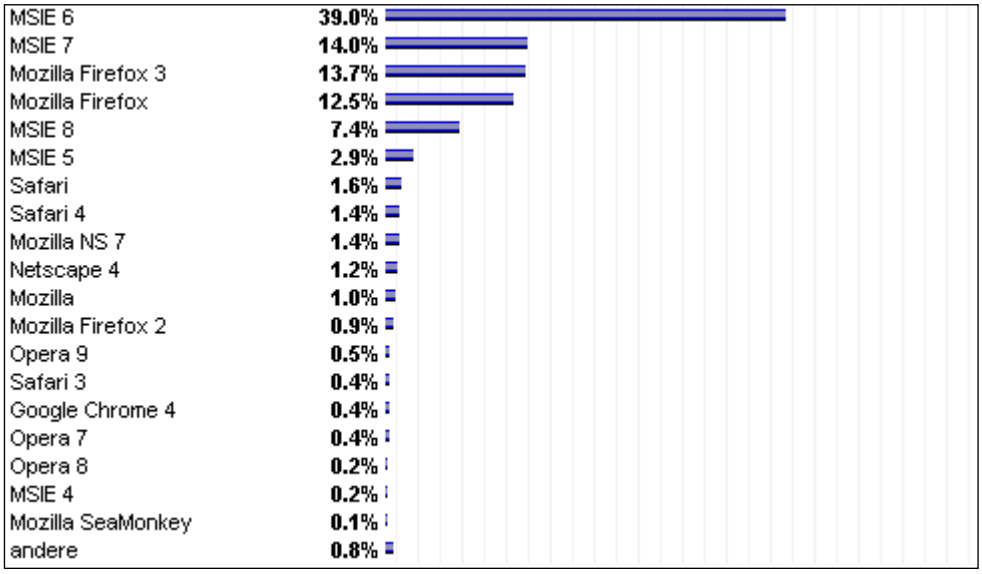


Abbildung 10: Die meist verwendeten Browser auf WebHints.de (2010)

Aus der Statistik ist ersichtlich, dass Internet Explorer, Mozilla Firefox, Apple Safari, Netscape Navigator, Opera und Google Chrome die am meist verwendeten Web Browser sind.

Damit die Tests bezüglich dem neusten Stand der Technik repräsentativ sind, werden die neusten verfügbaren Versionen getestet.

Webbrowser	Version	JavaScript Engine
Internet Explorer	8	Trident
Mozilla Firefox	3.6	SpiderMonkey
Apple Safari	4.0	JavaScriptCore
Netscape Navigator	4	Gecko
Opera	10.50	Carakan
Google Chrome	4	V8

Tabelle 1: Die getesteten Webbrowser

2.7 Use Cases

Die verwendete Vorlage stammt aus dem Buch Applying UML and Patterns von Craig Larman [11].

Für ein besseres Verständnis wurden die Fachbegriffe in englischer Sprache übernommen.

Use Case Section	Comment
Level	User Goal
Primary Actor	Stimmberechtigte Person
Stakeholders and Interests	Staat / Wahlaufsicht
Preconditions	Die stimmberechtigte Person muss ein gültiger Stimmzettel haben bzw. als solche registriert sein.
Postconditions (Success Guarantee)	Die Stimme wird auf Ihre Gültigkeit Überprüft und falls sie gültig ist, so wird sie berücksichtigt.
Special Requirements	Die Verschlüsselung muss innerhalb von 15 Sekunden stattfinden.
Technology and Data Variations List	JavaScript, JSON und die am meist verbreiteten Browser. Die Verschlüsselung erfolgt durch die Damgård-Jurik Verschlüsselung.

2.7.1 Main Success Scenario: Ja / Nein Abstimmung

Actor	Action
1. Stimmberechtigte Person	Meldet sich beim Server an.
2. Staat	Authentisiert die Person.
	Sendet die Webseite für die Abstimmung.
3. Stimmberechtigte Person	Wählt zwischen zwei Wahlmöglichkeiten aus (Ja / Nein).
	Verschlüsselt die Wahl.
	Sendet die Wahl zum Server.
4. Staat	Überprüft der verschlüsselte Stimmzettel mittels Zero Knowledge Proofs Protokoll.
	Bestätigt die Berücksichtigung der Stimme.

2.7.2 Main Success Scenario: Wahlen mit mehreren Kandidaten

Actor	Action
1. Stimmberechtigte Person	Meldet sich beim Server an.
2. Staat	Authentisiert die Person.
	Sendet die Webseite für die Abstimmung.
3. Stimmberechtigte Person	Wählt zwischen mehreren Kandidaten aus.
	Verschlüsselt die Wahl.
	Sendet die Wahl zum Server
4. Staat	Überprüft der verschlüsselte Stimmzettel mittels Zero Knowledge Proofs Protokoll.
	Bestätigt die Berücksichtigung der Stimme.

2.7.3 Alternate Flow: Der Stimmzettel wird abgelehnt

Actor	Action
3. Staat	Überprüft der verschlüsselte Stimmzettel mittels Zero Knowledge Proofs Protokoll.
	Lehnt die Stimme ab und warnt vor der Ablehnung der Stimme.

3 Produktanalyse

Die Produktanalyse dient dazu einen geeigneten Kandidaten für den Einsatz als Big-Integer Library zu finden.

3.1 Big-Integer Library

Falls beide Bibliotheken sich als ungeeignet erweisen, so werden die beiden Test-Phasen mit anderen Kandidaten wiederholt. Die ersten Kandidaten, die getestet werden wurden in der Aufgabenstellung vorgeschlagen. Es handelt sich um folgende Bibliotheken:

- JSBN JavaScript Big-Integer Library [12]
- Leemon Bairds JavaScript Big-Integer Library [13]

3.1.1 Testspezifikation

Die Tests erfolgen in zwei Phasen: die erste Phase überprüft die Bibliotheken auf die Korrektheit Ihrer Funktionen. Die zweite Phase überprüft die Bibliotheken auf die Tauglichkeit einer Damgård-Jurik Verschlüsselung. Folgender Test Infrastruktur wurde verwendet:

Merkmal	Beschreibung
Computer Marke	Fujitsu Siemens CELSIUS W360
Betriebssystem	Microsoft Windows XP Professional (Service Pack 3)
Prozessor	Intel Core 2 Duo CPU – E6750 @ 2.66 GHz
Arbeitsspeicher	3.00 GB DIMM DDR2 (666 MHz)

Tabelle 2: Konfiguration des Testrechners

Um die erste Phase zu bestehen, müssen folgende Kriterien genügen.

1. Die Aufgaben müssen richtig gelöst werden. Der berechnete Wert muss gleich dem Referenzwert sein, der mir der Java Referenz-Implementation BigInteger vorberechnet wurde. Folgende Funktionen werden getestet:

   - Addition / Subtraktion
   - Multiplikation / Division
   - Potenzieren
   - Modulo
   - Potenzieren-Modulo
   - Grösster gemeinsamer Teiler (ggT)

2. Als Referenz-Browser wird Firefox 3.6 verwendet.

Um die zweite Phase zu bestehen, muss folgende Berechnung in einer sinnvollen Zeit bis maximal 15 Sekunden berechnet werden [14]. Diese Operation wurde gewählt, weil sie die rechenaufwendigste Operation der Verschlüsselung ist.

$$r^{n^s} mod\ n^{(s+1)}$$

- r ist eine zufällig gewählte natürliche Zahl der Grösse 1024, 1536 oder 2048 Bit.

- n ist ein Produkt zweier zufällig gewählten Primzahlen. Die Grösse von n kann 1024, 1536 oder 2048 Bit betragen. Die Primzahl wurde mit dem Rabin-Miller Test ermittelt. Die Wahrscheinlichkeit, dass r eine Primzahl ist beträgt $1 – 1 / 2^{10} = 99.9$ % [21].

- s kann 1, 2, oder 3 sein.

3.1.2 Testdurchführung

3.1.2.1 Korrektheit der Bibliotheken

Der Soll-Wert wird durch die Java Klasse BigInteger vorberechnet. Falls der Soll-Wert gleich dem Ist-Wert ist, so wird der Vergleich als erfolgreich bezeichnet, sonst als fehlerhaft. Falls die Funktion nicht angeboten wird, so wird diese mit einem – gekennzeichnet. Die Funktion wird als korrekt bezeichnet, falls alle 3 Tests erfolgreich waren, sonst ist sie fehlerhaft. Die Messwerte zu der Tabelle 3 sind im Anhang zu finden.

Operation	JSBN JavaScript Big-Integer Library			Leemon Bairds JavaScript Big-Integer Library		
	1. Vergleich	2. Vergleich	3. Vergleich	1. Vergleich	2. Vergleich	3. Vergleich
Addition	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich
Subtraktion	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich
Multiplikation	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich
Division	erfolgreich	erfolgreich	erfolgreich	-	-	-
Potenzieren	erfolgreich	erfolgreich	erfolgreich	-	-	-
Modulo	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich
Potenzieren - Modulo	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich
Grösster gemeinsamer Teiler (ggT)	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich	erfolgreich

Tabelle 3: Ergebnisse der Testdurchführung

3.1.2.2 Tauglichkeit einer Damgård-Jurik Verschlüsselung in JavaScript

Internet Explorer 8.0

Grösse von n und r [Bit]	s = 1 Ausführungszeit [Sekunden]		s = 2 Ausführungszeit [Sekunden]		s = 3 Ausführungszeit [Sekunden]	
	JSBN	Leemon	JSBN	Leemon	JSBN	Leemon
1024	11.9	22.9	51.3	100.0	135.0	263.8
1536	38.7	74.9	171.2	334.7	453.3	884.0
2048	91.0	174.1	404.7	789.7	1068.7	2098.3

Mozilla Firefox 3.6

Grösse von n und r [Bit]	s = 1 Ausführungszeit [Sekunden]		s = 2 Ausführungszeit [Sekunden]		s = 3 Ausführungszeit [Sekunden]	
	JSBN	Leemon	JSBN	Leemon	JSBN	Leemon
1024	2.8	1.5	11.8	6.8	32.0	18.0
1536	9.2	5.1	40.5	23.0	107.5	59.6
2048	21.4	12.0	96.0	54.0	259.6	143.6

Apple Safari 4.0

Grösse von n und r [Bit]	s = 1 Ausführungszeit [Sekunden]		s = 2 Ausführungszeit [Sekunden]		s = 3 Ausführungszeit [Sekunden]	
	JSBN	Leemon	JSBN	Leemon	JSBN	Leemon
1024	9.6	1.0	42.8	4.6	111.8	12.0
1536	32.5	3.4	142.7	15.3	378.6	40.5
2048	75.7	7.9	338.4	36.1	887.4	96.8

Opera 10.50

Grösse von n und r [Bit]	s = 1 Ausführungszeit [Sekunden]		s = 2 Ausführungszeit [Sekunden]		s = 3 Ausführungszeit [Sekunden]	
	JSBN	Leemon	JSBN	Leemon	JSBN	Leemon
1024	1.2	1.3	5.2	5.6	14.0	15.5
1536	4.0	4.2	17.4	19.0	46.0	50.2
2048	9.3	9.7	40.8	44.5	108.9	115.4

Google Chrome 4.0

Grösse von n und r [Bit]	s = 1 Ausführungszeit [Sekunden]		s = 2 Ausführungszeit [Sekunden]		s = 3 Ausführungszeit [Sekunden]	
	JSBN	Leemon	JSBN	Leemon	JSBN	Leemon
1024	0.4	0.9	1.8	4.1	4.8	11.0
1536	1.3	3.2	6.0	14.0	15.8	37.1
2048	3.2	7.7	14.0	33.9	37.0	88.4

3.1.3 Testauswertung

Beide Bibliotheken sind korrekt und liefern bei allen unterstützen Operationen richtige Resultate. Die JSBN Bibliothek lehnt sich an die Syntax von der Java Big-Integer Library und ist daher sehr intuitiv zu bedienen. Man kann annehmen, dass diese Bibliothek eine gewisse Notorietät besitzt, da sie in der offiziellen V8 JavaScript Engine Benchmark von Google benutzt [15] wird. Die Unterstehende Abbildung zeigt eine Übersicht der Leistung beider Bibliotheken. Aus Übersichtlichkeitsgründen werden die Werte ab 40 Sekunden nicht mehr dargestellt. Eine vollständige Grafik ist am Schluss dieses Kapitels vorhanden.

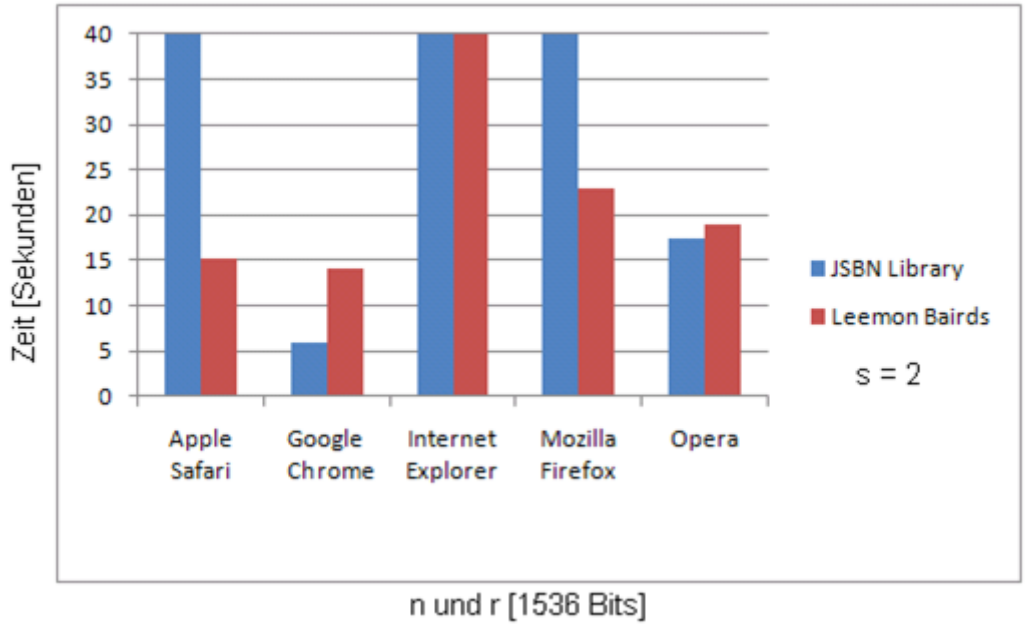


Abbildung 11: Leistungsübersicht der Bibliotheken auf verschiedenen Browser

Nach einer Profiler Analyse der JSBN Library ist ersichtlich, dass die hohen Berechnungszeiten des Internet Explorer 8.0 auf die Leistung des Bit-Shift Operators, im Vergleich zur Konkurrenz, zurückzuführen ist. Die Methode am3 der JSBN Library, nimmt am meisten Zeit in Anspruch und verwendet hauptsächlich den Bit-Shift Operator.

Self [%]	Total [%]	Function
77.10	77.10	am3 (jsbn.js)
10.80	59.79	montReduce (jsbn.js)
4.96	26.41	bnpSquareTo (jsbn.js)

Tabelle 4: Profiling mit Google Chrome der JSBN Library

Diese Vermutung wurde experimentell folgendermassen überprüft: Die Durchführungszeit von 10^7 Bit-Shift Operationen (0xffff << 2) mit verschiedenen Browser wurde gemessen und folgende Resultate bestätigen die oben erwähnte Vermutung.

Browser	Ausführungszeit für 10^7 Bit-Shift Operationen [ms]
Internet Explorer 8	515
Mozilla Firefox 3.6	29
Apple Safari 4.0	23
Opera 10.50	23
Google Chrome 4.0	16

Tabelle 5: Leistungsmessung der Bit-Shift Operation mit verschiedenen Browser

Desweiteren wird vermutet, dass die grossen Leistungsunterschiede der Browser durch interne Optimierungen der Big-Integer Libraries zu erklären sind. Die JSBN Bibliothek passt beispielsweise die Blockgrösse der Montgomery Reduktion je nach Browser zwischen 26 und 30 Bit an. Eine Profiler-Analyse des Codes konnte jedoch keine konkrete Hinweise auf bestimmen Schwächen der JavaScript Engines aufzeigen.

3.1.4 Schlussfolgerung

Die Wahl der Bibliothek hängt davon ab welchen Browser man unterstützen möchte. Für den Mozilla Firefox und Apple Safari ist die Leemon Bibliothek schneller, für Google Chrome und Opera die JSBN Bibliothek. Einen guten Ansatz wäre beide Bibliotheken zu verwenden und den Browser ermitteln lassen, welche Bibliothek im Konkreten Fall zur Anwendung kommt.

Diese Lösung wurde als Prototyp programmiert (damgaardjurik-jsbn-leemon.html) und initialisiert die performantere Bibliothek für den verwendeten Browser selbständig. Als Preprocessing Berechnung wird mit jede Bibliothek eine 300 Bit Verschlüsselung durchgeführt. Die Bibliothek mit der schnelleren Berechnung wird als Referenz für die richtige Berechnung verwendet.

Wegen der Transparenz des Codes wird diese Möglichkeit aber nicht eingesetzt. Stattdessen wird das beste Ergebnis in den Tests als Referenz benutzt: Google Chrome mit JSBN Bibliothek.

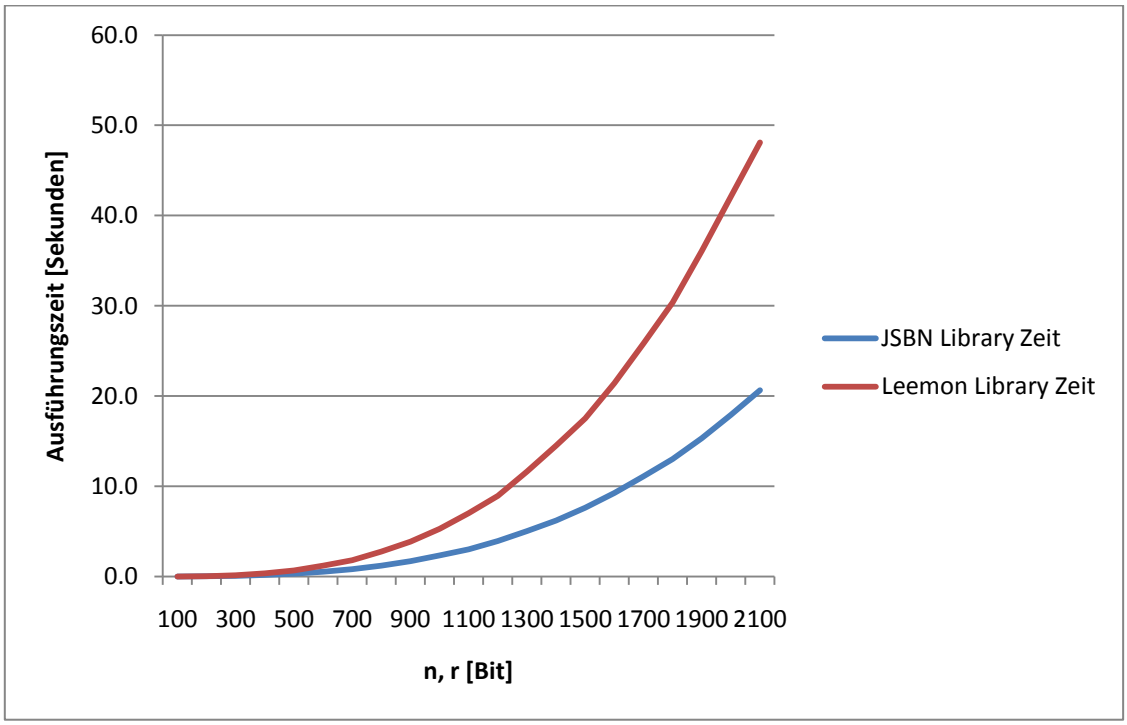
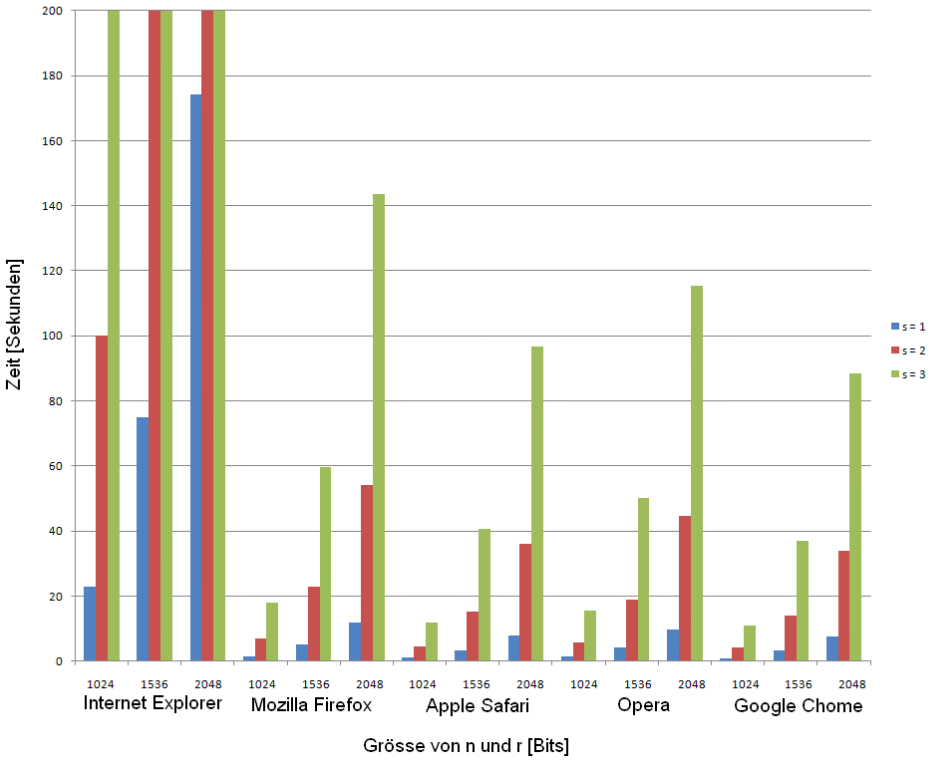


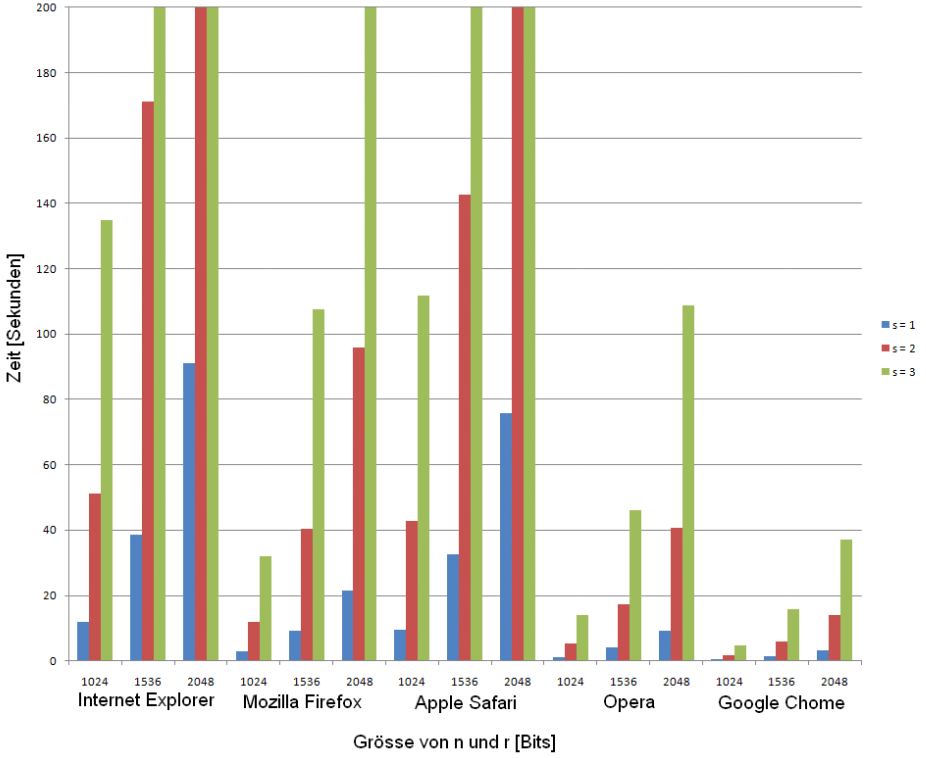
Abbildung 12: Leistungsvergleich mit steigender Schlüssellänge r und n in Google Chrome

Gesamtübersicht der Messresultate

JSBN JavaScript Big-Integer Library



Leemon Bairds JavaScript Big-Integer Library



4 Design

Das Client-Programm wird von der evotingwebclient.html implementiert. Diese besitzt folgende Abhängigkeiten.

4.1 Abhängigkeitsdiagramm

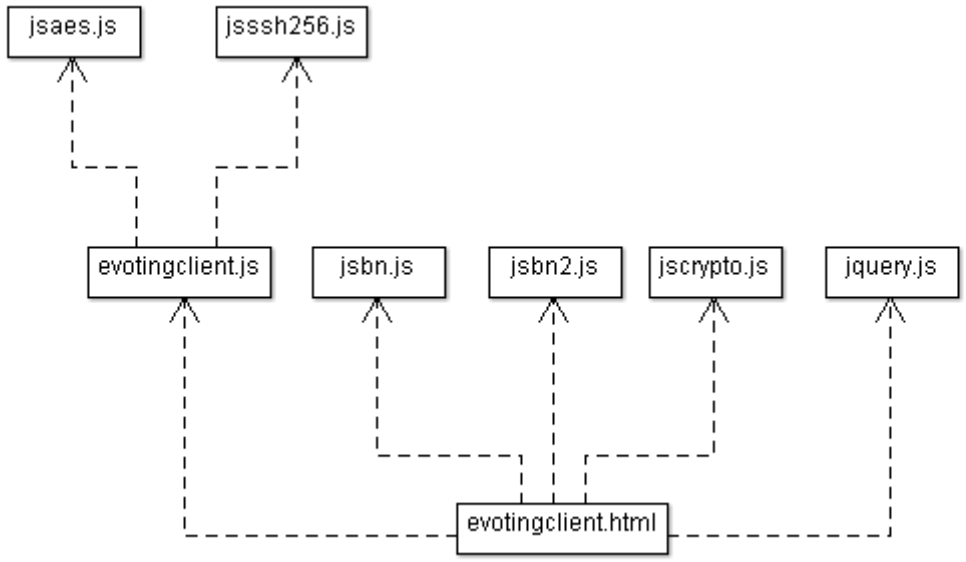


Abbildung 13: Abhängigkeitsdiagramm

Name	Beschreibung
evotingwebclient.html	Bildet die Wahlseite
jsbn.js	Grundfunktionalität der BigInteger Library
jsbn2.js	Erweiterte Funktionalität der BigInteger Library für Verschlüsslungen
jscrypto.js	Generierung von echten Zufallszahlen in JavaScript
evotingclient.js	Funktionalität der Damgård-Jurik Verschlüsselung
jsaes.js	AES Verschlüsselung in JavaScript
jssha256.js	SHA 256 Einwegfunktion in JavaScript
jquery.js	Funktionalität der JSON Kommunikation

4.1.1 evotingclient.js

Die evotingclient.js Bibliothek beinhaltet alle Funktionen für die Damgård-Jurik Verschlüsselung und das Non Interactive Zero Knowledge Proofs Protokoll sowie die dafür benötigten Hilfsfunktionen. Alle Übergabeparameter, die ein BigInteger Objekt repräsentieren werden mit „bn“ als Präfix gekennzeichnet. BigInteger Objekt-Arrays werden mit „bns“ gekennzeichnet. String und Integer Datentypen werden nicht im Dateinamen gekennzeichnet, sondern werden mit der check_type Methode am Anfang jeder Methode überprüft.

Folgende Tabellen geben eine Übersicht der Hilfsmethoden und der Hauptmethoden. Weitere Informationen zur Verschlüsselung und dem Zero Knowledge Proofs Protokoll sind im Analyse-Teil dieser Dokumentation beschrieben.

Hilfsmethoden

Methodennamen	Beschreibung	Eingabe	Ausgabe (Return)
check_type	Dient zur Überprüfung des Datentyps der Eingabe.	variable [string] expected_type [string] source [string]	[exceptio n]
get_string_with_lea ding_zero	Stellt sicher, dass die führende Null bei der Stringkonversion vorhanden ist.	bn_input [object] radix [number]	[string]
to_hex_array_with_ leading_zero	Konvertiert ein dezimales Array in ein Hexadezimales Array mit führenden Nullen.	input [object]	[object]
padding_to_messag e_length	Die führenden Nullen werden ergänzt gemäss der angegebenen Länge.	message [string] length [number]	[string]
get_zn	Ist die Verifikation des Zero-Knowledge-Proofs Protokolls, die vom Server berechnet wird. Wird clientseitig als Testfunktion optional verwendet.	bns_response_z [object] bn_ns [object] bn_ns1 [object]	[object]
get_challenge_verifi cation	Testfunktion	bns_response_e [object] b [number]	[object]
get_commitment_v erification	Testfunktion	bns_commitment [object] bns_bulletin_board [object] bns_response_e [object] bn_ns1 [object]	[object]

Hauptmethoden

Methodennamen	Beschreibung	Eingabe	Ausgabe (Return)
damgard_jurik_encryption	Führt eine Damgård-Jurik Verschlüsselung durch.	bn_m [object] bn_r [object] bn_n [object] s [object] bn_ns [object] bn_ns1 [object]	[object]
damgaard_jurik_encryption_optimized	Führt eine Damgård-Jurik Verschlüsselung durch, gemäss einer vorgeschlagenen Optimierung von J. Jurik.	bn_m [object] bn_r [object] bn_n [object] s [object] bn_ns [object] bn_ns1 [object]	[object]
get_messages	Bildet die gültigen Stimmzettel als $message\_base^{message\_exponent}$ Radix definiert in welchem Zahlensystem die message_base angegeben wird.	message_base [number] message_exponent[number] number_of_candidates [number] radix [number]	[object]
get_random_binary_number	Gibt eine Zufallszahl zurück gemäss bit_size Länge.	bit_size [number]	[object]
get_bulletin_board_entries	Bulletin Board	bns_message [object] bn_c [object] bn_n [object] bn_ns1 [object]	[object]
get_commitment	Commitment	bns_message [object] selected_message [number] bns_z [object] bns_e [object] bns_uk [object] bn_ns [object] bn_ns1 [object]	[object]
get_response_e	Response e	bn_challenge [object] bns_e [object] selected_message [object] b [number]	[object]
get_response_z	Response z	bns_z [object] bn_r [object] bns_response_e [object] selected_message [object] bn_n [object]	[object]
get_sha_256_aes_challenge	Non-Interactive-Zero-Knowledge-Proofs-Challenge	election_id [string] voter_id [string] bn_c [object] bns_commitment [object]	[object]

		b [number] radix [number]	

4.1.2 jsbn.js

Die jsbn.js Bibliothek ermöglicht die Darstellung von Integer Datentypen in beliebiger Genauigkeit (arbitrary-precision integer arithmetic) und bietet die grundlegende Funktionalität für eine RSA Verschlüsselung. Eine Auflistung der gegebenen Methoden mit Beispielen erläutert die Funktionsweise der jsbn.js Bibliothek.

Erzeugen eines BigInteger Objekts

- var bn_number = new BigInteger (“12 “, 10);
- var bn_number = new BigInteger (“d6 “, 16);
- var bn_number = new BigInteger (array);

Vorgegebene Konstanten

- var bn_zero = BigInteger.ZERO
- var bn_one = BigInteger.ONE;

Methoden

- var string_representation = bn_biginteger.toString(16);
- bn_negative = bn_positive.negate();
- bn_absolute = bn_biginteger.abs():
- compareTo(bn_fist, bn_second);
- bit_length = bn_biginteger.bitLength();
- bn_modulus = bn_a.mod(bn_b);
- bn_modPow = bn_a.modPowInt(b, bn_c);

4.1.3 jsbn2.js

Weitere Funktionen, die über eine RSA Verschlüsselung hinausgehen sind in der jsbn2.js Bibliothek enthalten. Dazu zählen die Grundrechenarten, Boolesche Operatoren, sowie Methoden für die Manipulation von Objekten. Eine Liste der Methoden ist am Ende der jsbn2.js Datei zu finden. Erläuterungen zu den einzelnen Methoden sind aus der Java API zu entnehmen.

4.1.4 jscrypto.js

Die jscrypto.js Bibliothek liefert zufällige Zahlen, die von verschiedenen Quellen stammen. Unter anderem wird die Mauszeiger Bewegung als Seed für das Entropiepool verwendet. Dafür ist die folgende Initialisierung der Random() Bibliothek durchzuführen.

Die eingebauten Entropy Collectors starten.

- Random.start_collectors();

Den EventListener binden.

- Random.addEventListener("seeded");

Random.random_words(1)[0]

- Ein Block von 32 Bit Länge beziehen.

4.1.5 jsaes.js

Die jsaes.js Bibliothek führt eine AES Verschlüsselung in JavaScript durch.

Initialisierung des Schlüssels.

- AES_ExpandKey(key);

Initialisierung der Verschlüsselung.

- AES_Init();

Verschlüsselung durchführen.

- AES_Encrypt(block, key);

Verschlüsselung abschliessen.

- AES_Done();

4.1.6 jssha256.js

Die jssha256.js Bibliothek berechnet einen SHA-256 Wert in JavaScript aus.

Initialisierung der SHA Funktion.

- SHA256_init();

Nachricht übergeben. Dieser Schritt kann mehrmals wiederholt werden.

- SHA256_write(message);

Hash Wert berechnen.

- var hash_value = SHA256_finalize();

4.2 Interaktionsdiagramm

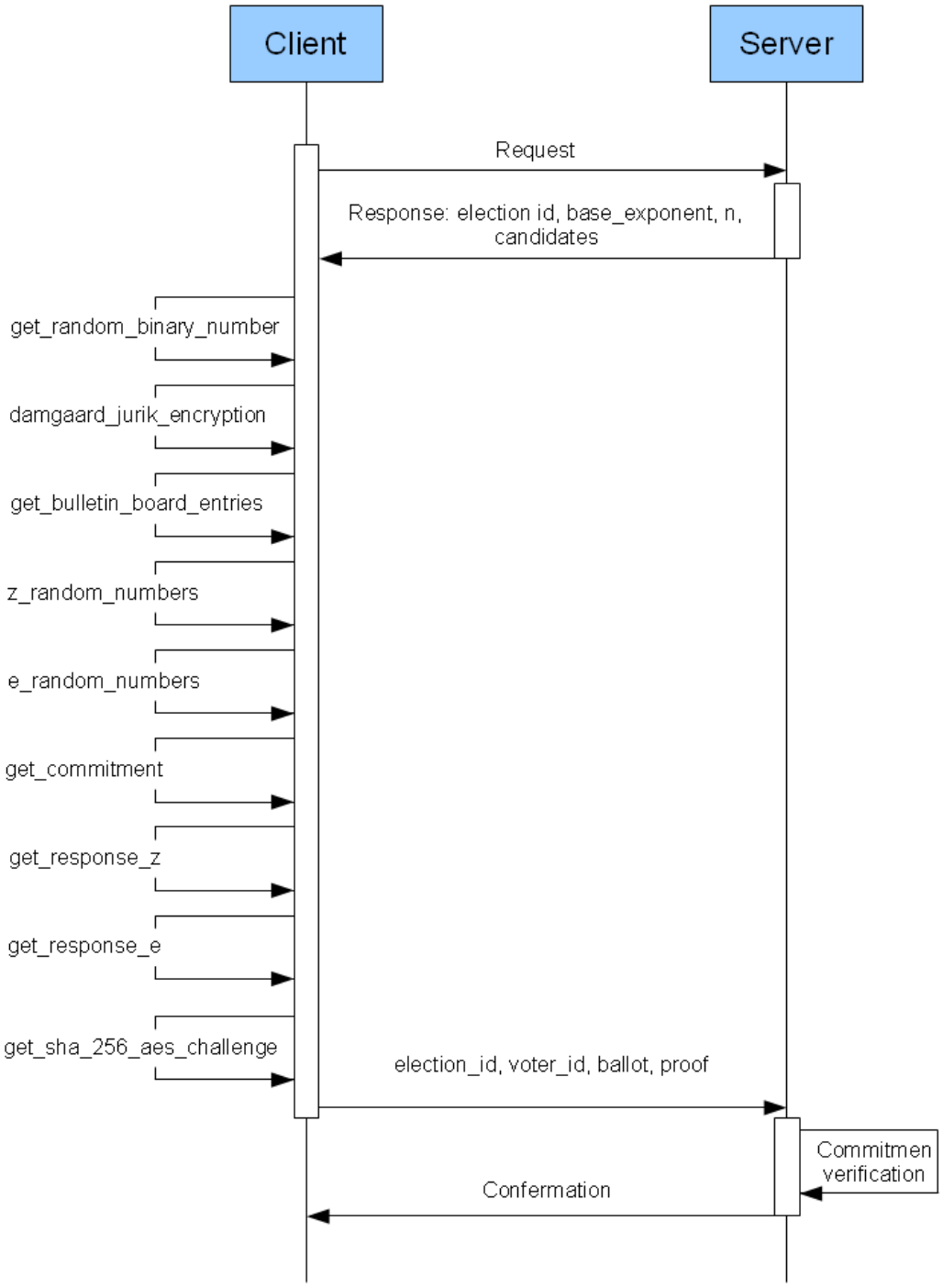


Abbildung 14: Interaktionsdiagramm Non Interactive Zero Knowledge Proofs Protokoll

4.3 Codeauszug der Hauptmethoden

```
function damgard_jurik_encryption (bn_m, bn_r, bn_n, s, bn_ns, bn_ns1) {
	// c[m,r] = g^m * r^(n^s) mod n^(s+1), with generator g = (n+1)
	var bn_gmz = (BigInteger.ONE.add(bn_n)).modPow(bn_m, bn_ns1);
	var bn_rns = bn_r.modPow(bn_ns, bn_ns1);
	return bn_gmz.multiply(bn_rns).mod(bn_ns1);
}

function damgaard_jurik_encryption_optimized (bn_m, bn_r, bn_n, s, bn_ns, bn_ns1) {
	// Source: http://www.brics.dk/~jurik/Research/Paillier/Cryptosystem/Cryptosystem.java
	var bn_c = BigInteger.ONE.add (bn_m.multiply (bn_n));
	var binomial = bn_m;
	var msg_i    = bn_m;
	var big_i    = BigInteger.ONE;
	var n_i      = bn_n;
	for (var i = 2; i <= s; i++) {
		n_i = n_i.multiply (bn_n);
		msg_i = msg_i.subtract (BigInteger.ONE);
		big_i = big_i.add (BigInteger.ONE);
		binomial = ((binomial.multiply (msg_i)).multiply (big_i.modInverse (bn_ns1))).mod (bn_ns1);
		bn_c = (bn_c.add (binomial.multiply (n_i))).mod (bn_ns1);
	}
	bn_c = (bn_c.multiply (bn_r.modPowInt (bn_ns, bn_ns1))).mod (bn_ns1);
	return bn_c;
}

function get_messages (message_base, message_exponent, number_of_candidates, radix) {
	// mi = message_base^message_exponent * i
	var bn_message_base = new BigInteger(message_base.toString(radix));
	var bn_messages = [];
	for (var i = 0; i < number_of_candidates; i++) {
		bn_messages[i] = bn_message_base.pow((i * message_exponent));
	}
	return bn_messages;
}

Random.nextBytes = function (x) {
	for (var i = 0; i < x.length; i++) {
		try {
			x[i] = Random.random_words(1);
		}
			catch (readiness_error) {
		}
	}
}
```

```
function get_random_binary_number (bit_size) {
	// Precondition : The random generator have to be inizialized.
	return new BigInteger(bit_size, Random);
}
function get_bulletin_board_entries (bns_message, bn_c, bn_n, bn_ns1) {
	// uk = c[m,r]/g^mk mod n^(s+1) must be an (n^s)-th power for k == i
	var bns_bulletin_board_entries = [];
	var len = bns_message.length;
	for (var i = 0; i < len; i++) {
		var bn_y = (BigInteger.ONE.add(bn_n)).modPow(bns_message[i],
bn_ns1).modInverse(bn_ns1);
		bns_bulletin_board_entries[i] = (bn_c.mod(bn_ns1).multiply(bn_y)).mod(bn_ns1);
	}
	return bns_bulletin_board_entries;
}
function get_commitment (bns_message, selected_message, bns_z, bns_e, bns_bulletin_board, bn_ns,
bn_ns1) {
	var bns_commitment = [];
	var len = bns_message.length;
	for (var i = 0; i < len; i++) {
		if (i == selected_message) {
			// ai = zi^(n^s) mod n^(s+1)
			bns_commitment[i] = bns_z[i].modPow(bn_ns, bn_ns1);
		}
		else {
			// ak = zk^(n^s) / uk^ek mod n^(s+1)
			var bn_ueInverse = bns_bulletin_board[i].modPow(bns_e[i],
bn_ns1).modInverse(bn_ns1);
			var bns_znue = bns_z[i].modPow(bn_ns, bn_ns1).multiply(bn_ueInverse);
			bns_commitment[i] = bns_znue.mod(bn_ns1);
		}
	}
	return bns_commitment;
}
function get_response_e (bn_challenge, bns_e, selected_message, b) {
	// response_ei = challenge - sum (ek) mod 2^b for k != i
	var bns_response_e = [];
	var bn_sum = BigInteger.ZERO;
	var len = bns_e.length;
	for (var i = 0; i < len; i++) {
		bns_response_e[i] = bns_e[i];
		if (i != selected_message) {
			bn_sum = bn_sum.add(bns_e[i]);
		}
	}
	bns_response_e[selected_message] = (bn_challenge.subtract(bn_sum)).mod((new
BigInteger("2")).pow(b));
```

```
	return bns_response_e;
}
function get_response_z (bns_z, bn_r, bns_response_e, selected_message, bn_n) {
	// response_zi = zi * r^response_e mod n for k == i
	var bns_response_z = [];
	var len = bns_z.length;
	for (var i = 0; i < len; i++) {
		bns_response_z[i] = bns_z[i];
		if (i == selected_message) {
			bns_response_z[i] = bns_z[i].mod(bn_n).multiply(bn_r.modPow(bns_response_e[i],
bn_n)).mod(bn_n);
		}
		else {
			bns_response_z[i] = bns_z[i];
		}
	}
	return bns_response_z;
}
function get_sha_256_aes_challenge (election_id, voter_id, bn_c, bns_commitment, b, radix) {
	// Precondition : b should be at least 127 bits.
	var aes_block_size = 128;
	var iterations = parseInt(b / aes_block_size + 1, 10);
	var message = election_id + voter_id + get_string_with_leading_zero(bn_c, radix);
	var len = bns_commitment.length;
	for (var i = 0; i < len; i++) {
		message += get_string_with_leading_zero(bns_commitment[i], radix);
	}
	SHA256_init();
	SHA256_write(message);
	initKey = SHA256_finalize();
	var challenge = [];
	for (var j = 0; j < iterations; j++) {
		key = initKey.slice();
		var block = [];
		for (var k = 0; k < 16; k++)  block[k] = 0;
		block[15] = j;
		AES_ExpandKey(key);
		AES_Init();
		AES_Encrypt(block, key);
		AES_Done();
		challenge = challenge.concat(block);
	}
	// Set Leading Bit to Zero
	if (challenge[0] > 127)  challenge[0] -= 128;
	return new BigInteger(to_hex_array_with_leading_zero(challenge).toString(radix), radix);
}
```

4.4 Performance Aspekte

4.4.1 Damgård-Jurik Verschlüsselung

Die Optimierung der Verschlüsselung ist durch die Verwendung von Binomialkoeffizient möglich. Da g = (n+1) ist, so kann g^m in folgender Form geschrieben werden.

$$g^m = 1 + m * n + \binom{m}{2} n^2 + \cdots + \binom{m}{s} n^s \bmod n^{s+1}$$

Diese Umformung erlaubt es die m-te Potenz in 5s Multiplikationen zu berechnen[8].

```
var bn_c = BigInteger.ONE.add (bn_m.multiply (bn_n));
var binomial = bn_m;
var msg_i    = bn_m;
var big_i    = BigInteger.ONE;
var n_i      = bn_n;
for (var i = 2; i <= s; i++) {
n_i = n_i.multiply (bn_n);
    msg_i = msg_i.subtract (BigInteger.ONE);
    big_i = big_i.add (BigInteger.ONE);
    binomial = ((binomial.multiply (msg_i)).multiply (big_i.modInverse (bn_ns1))).mod (bn_ns1);
    bn_c = (bn_c.add (binomial.multiply (n_i))).mod (bn_ns1);
}
bn_c = (bn_c.multiply (bn_r.modPowInt (bn_ns, bn_ns1))).mod (bn_ns1);
return bn_c;
```

Abbildung 15: Optimierte Verschlüsselung nach http://www.brics.dk/ jurik/research.html

4.4.2 Zero Knowledge Proofs

Die Methoden mit der längsten Ausführungszeit sind die Verschlüsselung, die Berechnung der Bulletin Board Einträge und das Commitment, alle andere Operationen benötigen einen vernachlässigbaren kleinen Zeitaufwand und werden hier nicht berücksichtigt.

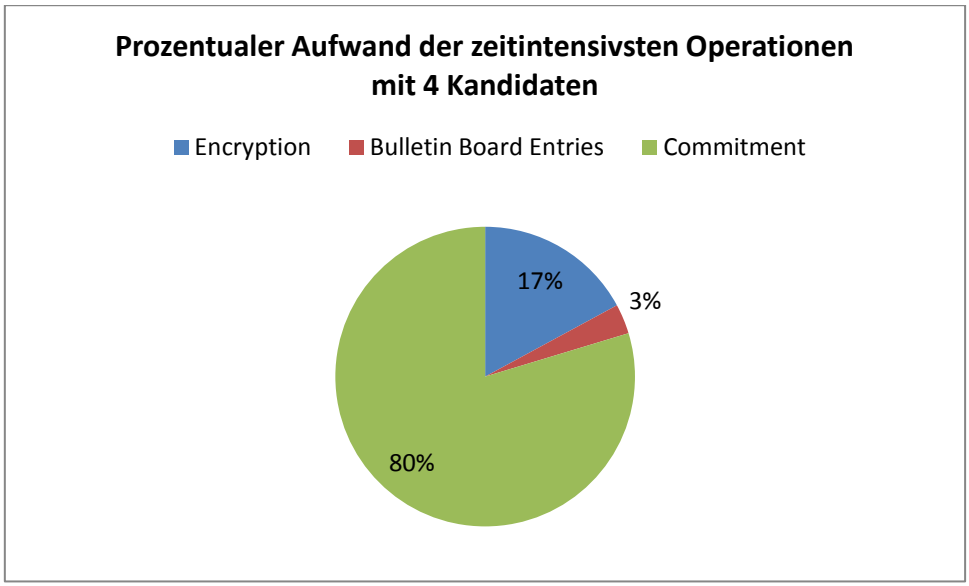


Abbildung 16: Prozentualer Aufwand der zeitintensivsten Operationen mit 4 Kandidaten

Die Berechnungszeit des Commitments ist Abhängig von der Anzahl der Kandidaten. Es ist wichtig zu Beachten, dass die Anzahl Kandidaten die gesamte Berechnungszeit linear ansteigen lässt. Eine Optimierungsmöglichkeit wäre, wenn der Server ein Teil der Bulletin Board Berechnung (g^m) vorberechnet und als JSON Parameter dem Client übergeben würde.

Folgende Grafik zeigt, wie sich die Anzahl Kandidaten auf die gesamte Ausführungszeit auswirken.

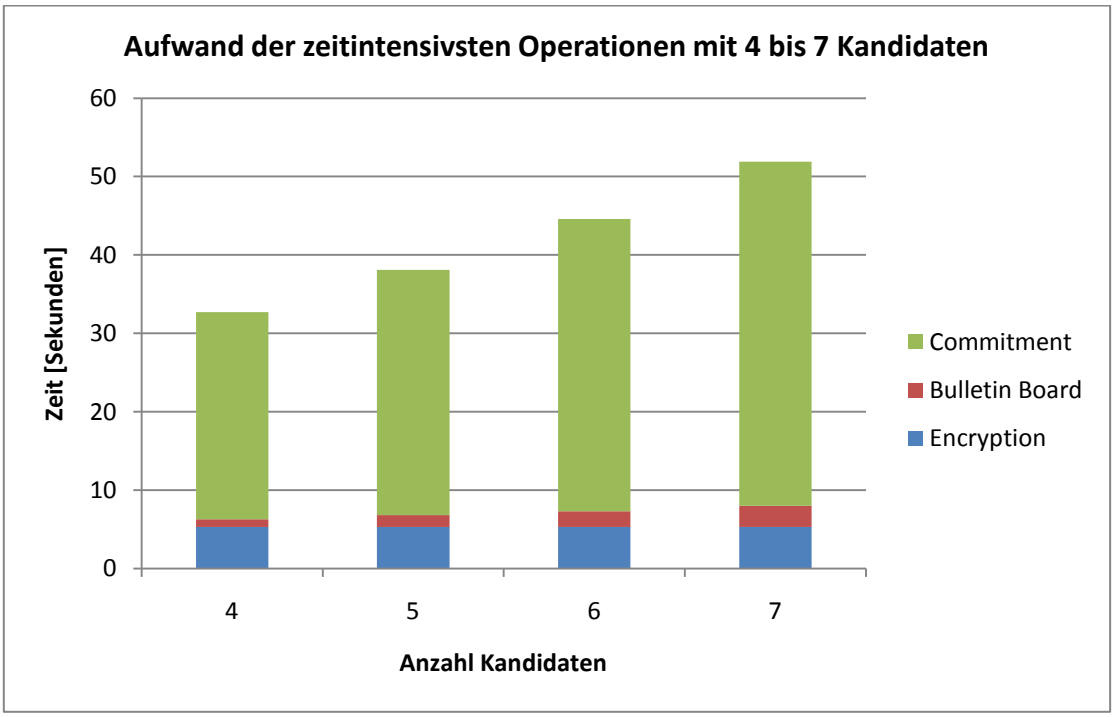


Abbildung 17: Aufwand der zeitintensivsten Operationen mit 4 bis 7 Kandidaten

5 E-Voting Web Client Bedienungsanleitung

5.1 Systemvoraussetzungen

Client:

Prozessor	mind. Intel Pentium IV 2.00 GHz oder gleichwertig
Arbeitsspeicher	mind. 512 MB
Betriebssystem	plattformunabhängig
Browser	• Google Chrome [empfohlen] (ab Version 5.0.375.70) • Mozilla Firefox (ab Version 3.6.3) • Opera (ab Version 10.53)

Der E-Voting Web-Client nutzt HTML, JavaScript und StyleSheets. Darum muss in jedem Fall JavaScript und StyleSheet-Unterstützung aktiviert werden.

Server:

Mind. JSON Parser muss vorhanden sein. Sonst sind keine besonderen Anforderungen verlangt.

5.2 Web-Client Bedienung

1. Installation

1. Entpacken Sie die *evotingwebclient.zip* Datei. Folgende Dateien müssen im entpackten Ordner enthalten sein:

-css
-----style.css
-images
-----images/hsr_logo.png
-----images/evoting_logo.png
-scripts
-----evotingclient.js
-----jquery-1.4.2.min.js
-----jsaes.js
-----jsbn.js
-----jsbn2.js
-----jscrypto.js
-----json.js
-----jssha256.js
evotingwebclient.html
index.html

2. Erstellen Sie einen Ordner „Evoting“ auf Ihrem Webserver und laden Sie alle Inhalte in diesem Ordner hoch.

2. Web-Client Start

Um den Webclient zu starten, navigieren Sie den Webbrowser zur entsprechende index.html Datei:

z.B.: http://..../evoting/index.html

3. Startseite-Anzeige

Die Startseite mit allem Parameter wird angezeigt.

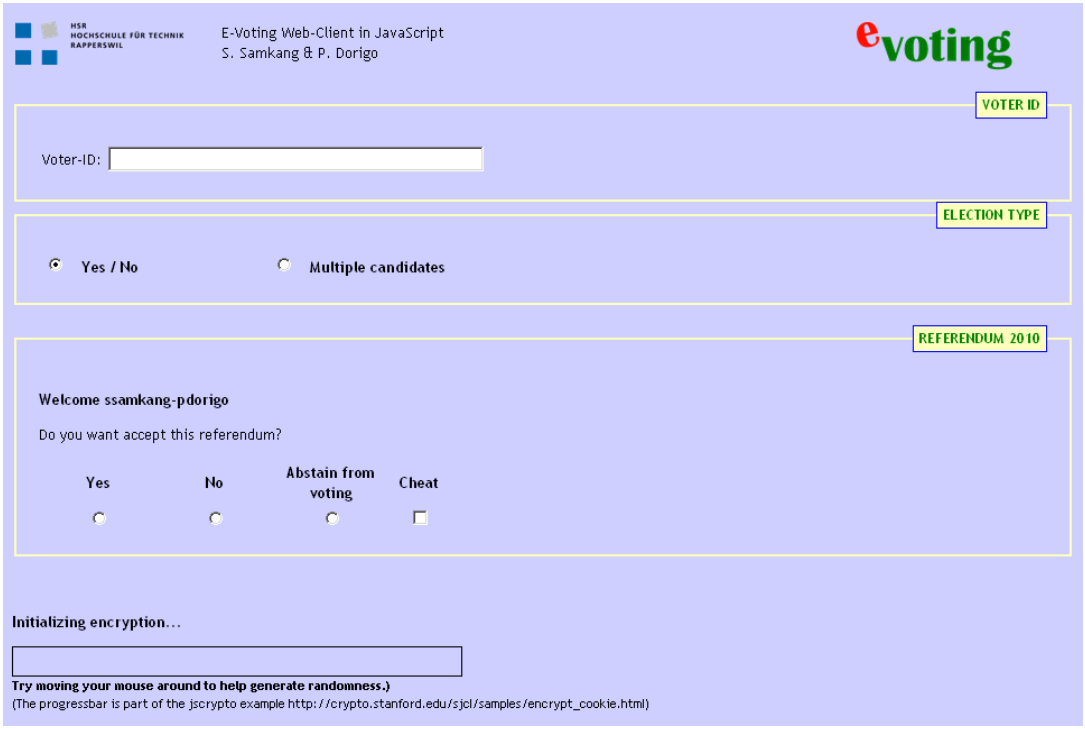


Abbildung 18: Startseite des Web-Clients

4. Zufallszahl Generierung

Sie müssen jetzt mit der Maus bewegen. Ist der Progressbalken nicht mehr sichtbar dann haben Sie erfolgreich eine zufällige Zahl generiert. Dieser Vorgang kann sehr schnell gehen.

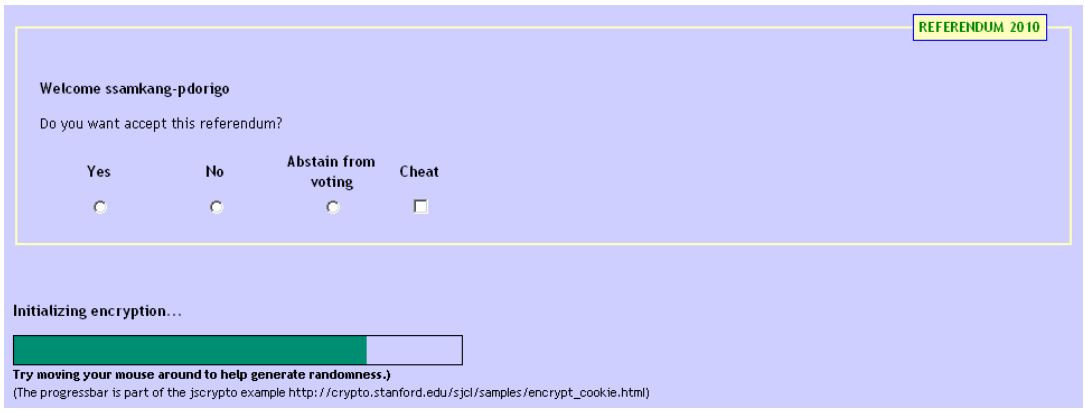


Abbildung 19: Zufallszahl Generierung

5. Voter-ID Eingabe

Der Encrypt-Button ist jetzt sichtbar. Geben Sie nun im leeren Textfeld eine Voter-ID ein, z.B. ssamkang-pdorigo.

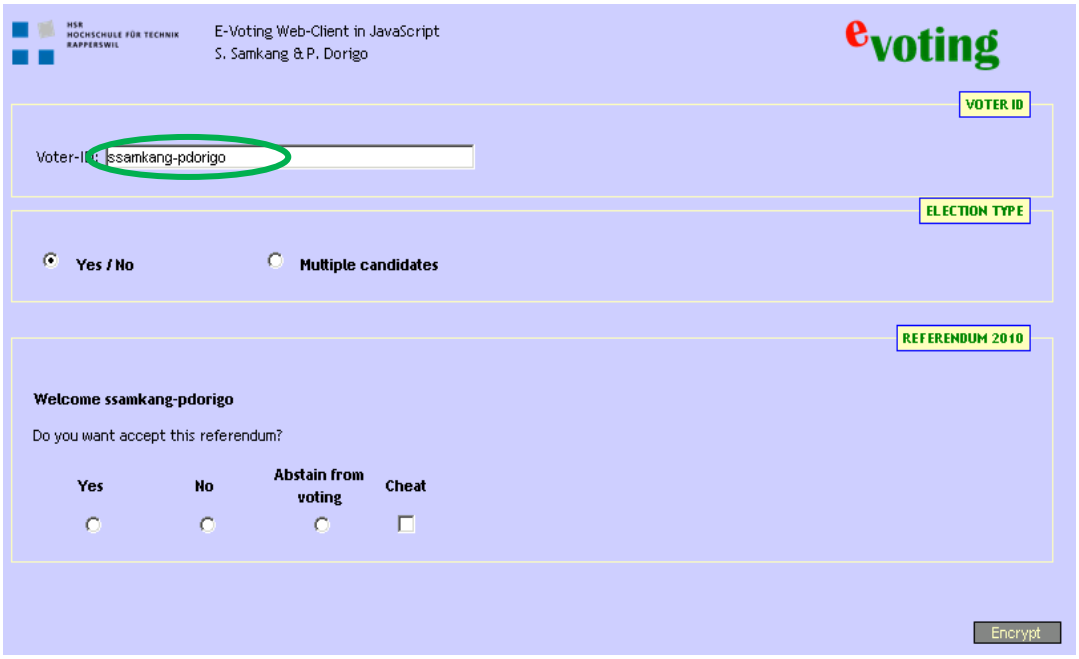


Abbildung 20: Voter-ID Eingabe

6. Wahl-Typ Auswahl

Treffen Sie ihren Wahl-Typ. Es steht Ihnen die Wahl Ja/Nein oder mehrfache Kandidaten zur Verfügung.

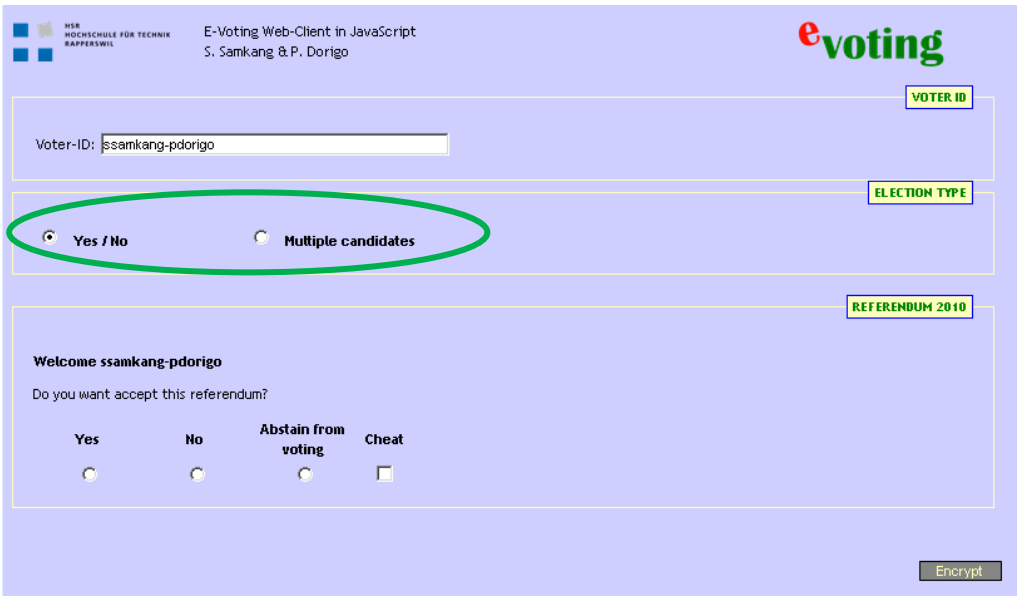


Abbildung 21: Wahl-Typ Auswahl

7. Wahlentscheidung

a. Ja/Nein Wahl

Wählen Sie Ja, Nein oder enthalten aus. Für den Testzweck können Sie auch die Cheat-Checkbox aktiviere. Bestätigen Sie ihre Auswahl mit „Encrypt“.

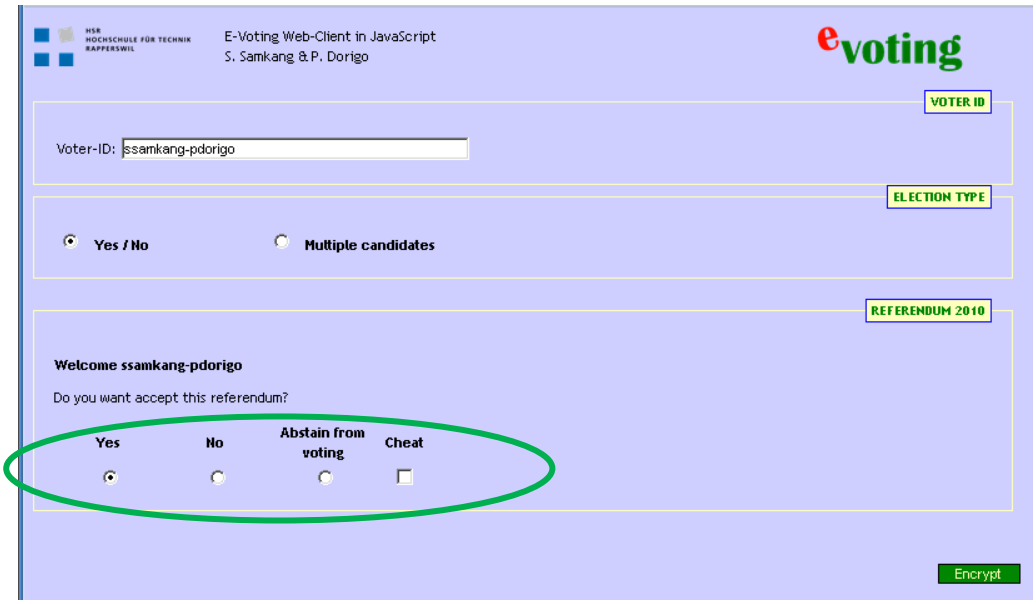


Abbildung 22: Ja/Nein Wahl

b. Wahl mit mehreren Kandidaten

Wählen Sie Ihren Kandidat aus oder enthalten Sie. Bestätigen Sie ihre Auswahl mit „Encrypt“.

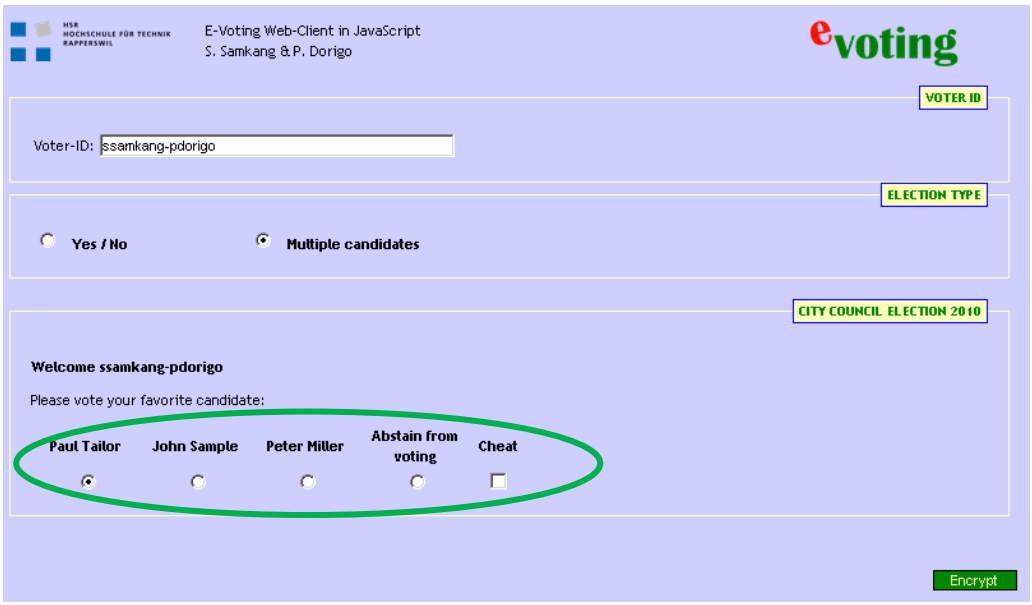


Abbildung 23: Wahl mit mehreren Kandidaten

c. Wahlmanipulation

Aktiviere den Cheat-Checkbox um die Wahl zu manipulieren. In diesem Prototyp wird der effektiv ausgewählte Punkt verdoppelt.

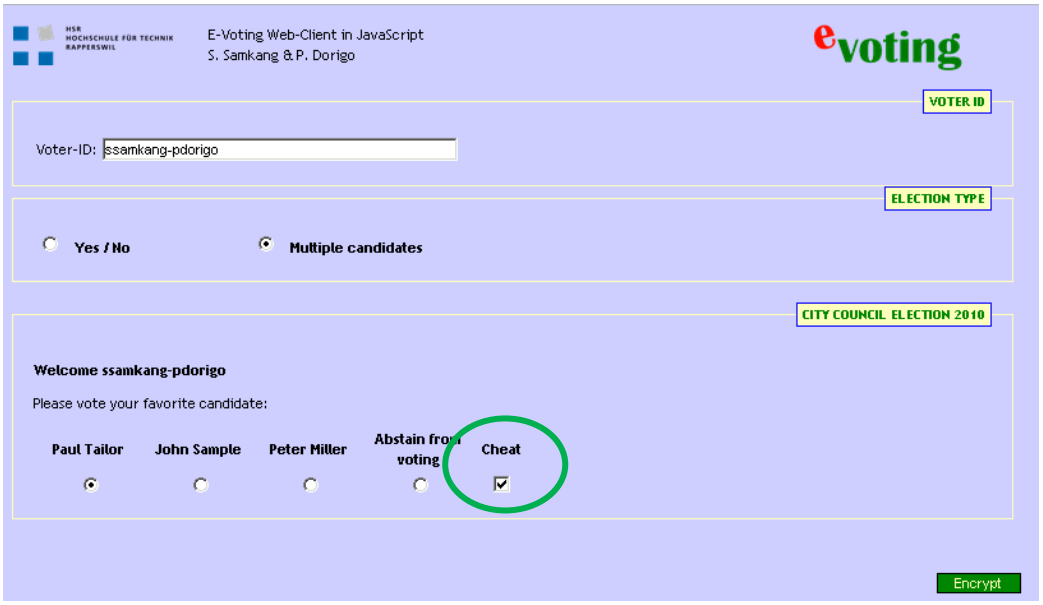


Abbildung 24: Wahlmanipulation

8. Auswertungsausgabe

Der Client führt die Damgård-Jurik Verschlüsselung und die Zero Knowledge Proofs durch und gibt alle Parameter individuell in separaten Registern aus.
Alle Ausgaben sind hexadezimaler Darstellung. Hier die Registerausgabe im Einzelnen.

a. Encryption

Im Encryption Register wird der Chiffrat ausgegeben.

Abbildung 25: Encryption

b. Bulletin Board

Im Bulletin Board Register sind die gültigen Nachrichten aufgelistet.

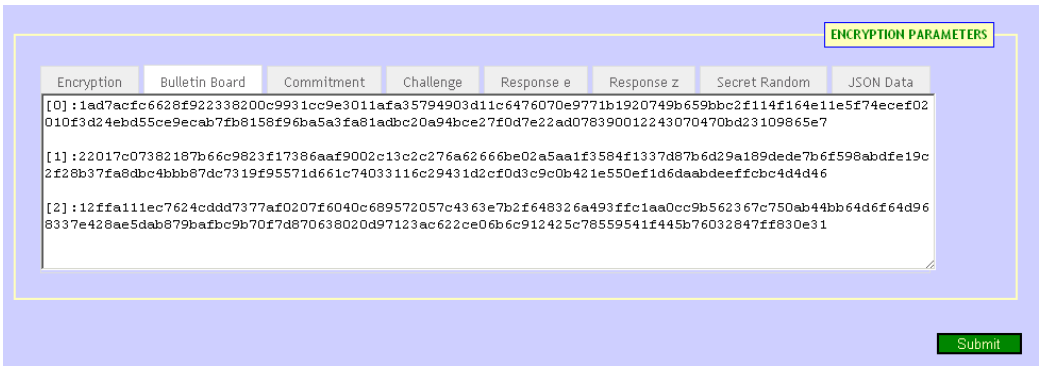


Abbildung 26: Bulletin board

c. Commitment

Im Commitment Register sind die Nachricht, die gewählt wurde sowie die Nachrichten, die nicht gewählte wurde, aufgelistet.

Abbildung 27: Commitment

d. Challenge

Im Challenge Register ist der Aufforderungswert (*engl. challenge*) aufgelistet.

Abbildung 28: Challenge

e. Response *e*

Im Response *e* Register ist die Nachricht, die gewählt wurde, nämlich die Differenz zwischen der Challenge und alle im Commitment verwendeten *e* aus.

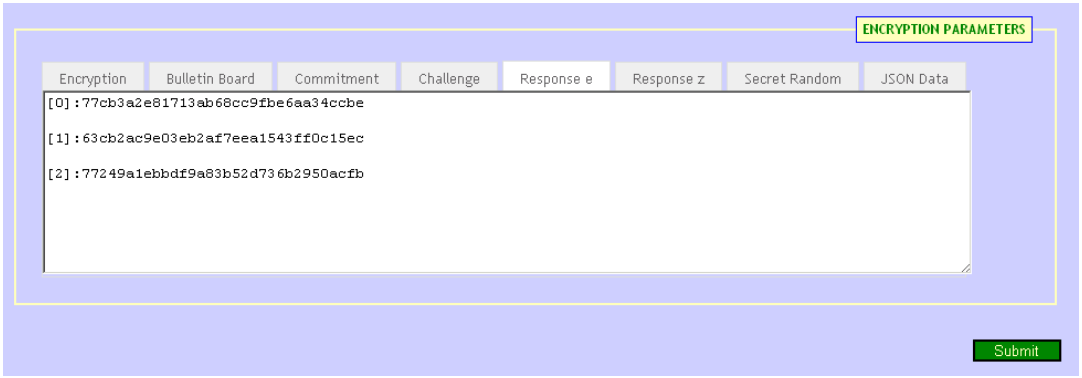


Abbildung 29: Response e

f. Response *z*

Im Response *z* Register ist die Nachricht, die nicht gewählt wurde, nämlich die Differenz zwischen der Challenge *es* und alle im Commitment verwendeten *e* aus.

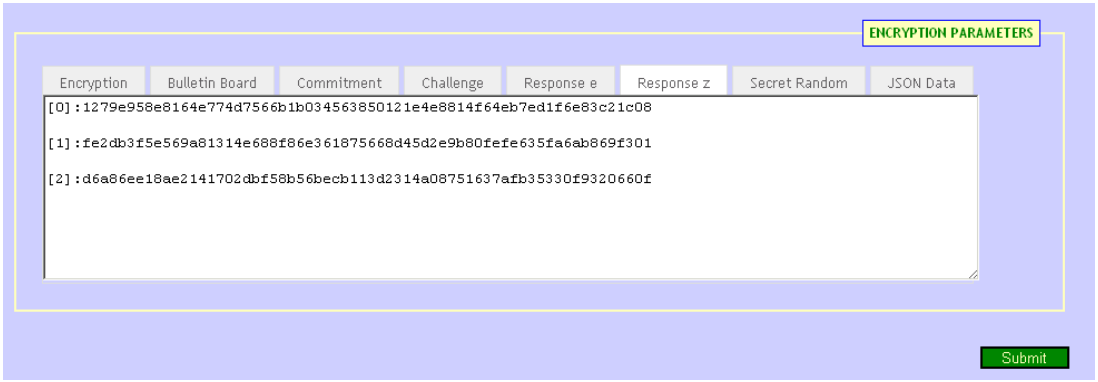


Abbildung 30: Response z

g. Secret Random

Im Secret Random Register ist die geheime Zahl *r*, die am Anfang zufällig generiert wurde, hinterlegt.

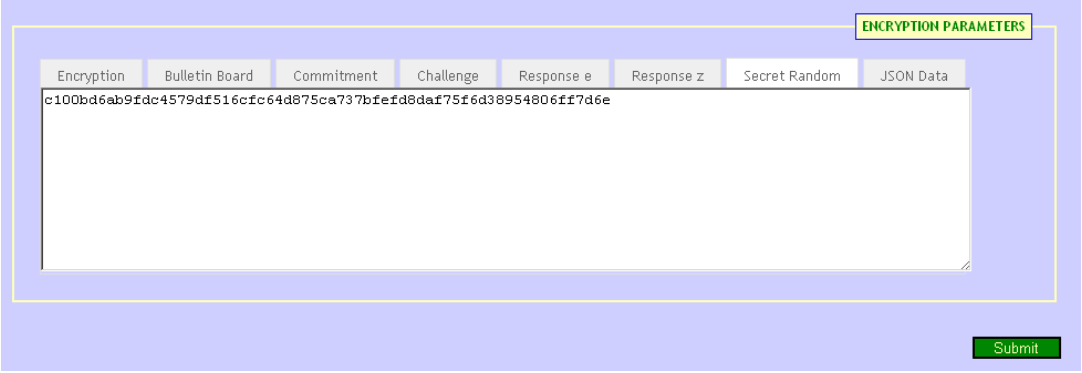


Abbildung 31: Secret Random

h. JSON Data

Im JSON Register sind folgende Werte auflistet:

i. die Wahl-Identifikation (engl.: electionID)
ii. die ID der stimmberechtigten Person (*engl.: voterID*)
iii. der eingereichte elektronische Stimmzettel (*engl.: ballot*)
iv. der Beweis (*engl.: proof*), beinhaltet die responses *e* und *z* und commitment *a*

Abbildung 32: JSON Data

9. Übermittlung der Daten

Drücken Sie auf den Submit-Button um die JSON Daten zu übermitteln. Der Server verifiziert die übermittelten Daten und bestätigt die Ergebnisse.

10. JSON Daten Verifikation

Die Ergebnisausgabe der Verifikation sieht dann folgendermassen aus:

JavaScript E-Voting Server

```
Election ID : City council election 2010
Voter ID    : ssamkang-pdorigo

Modulus:
n      =  c3:ff:27:4b:33:ce:e3:ea:21:99:96:97:34:fd:4a:65:67:d9:04:59:b2:20:5e:71:6f:6b:a5:35:f0:3a:25:6f
          fa:fa:db:5d:01:fc:8b:e0:4d:65:20:10:ae:ff:c4:7b:08:34:f6:ed:18:f5:4b:f6:c2:38:53:f4:9d:88:9c:0d
          15:c6:93:94:01:fb:65:1f:d3:c2:8c:7e:fe:ba:eb:3f:54:89:3d:ca:c0:f8:ce:8e:b0:16:6f:a0:f9:d9:f3:80
          a0:e5:15:c5:3d:ed:70:59:26:f2:b6:92:62:d3:ab:0e:8c:a0:f4:94:cf:b5:93:b7:59:6f:a9:de:b4:39:22:ab
          29:8c:30:5e:ed:2f:7e:86:0d:6b:d5:20:0b:2b:7d:cc:83:1c:12:dd:61:49:3b:84:d2:ae:c2:da:ff:48:74:4d
          78:28:82:9c:70:73:41:0b:af:93:0e:8a:32:ca:76:5d:eb:be:3e:e1:2d:97:d2:dc:d6:4d:f6:b3:54:6c:01:9f

Encrypted ballot:
c      =  2c:3d:8d:f5:72:73:f1:ea:d5:57:d9:0a:63:cd:95:09:96:46:a8:f6:71:70:6a:7c:85:1d:0a:9a:9a:ec:9e:42
          6d:ed:9c:2a:c6:37:c7:e4:0e:07:ff:1b:f4:77:7e:22:28:d1:6a:1c:d0:88:58:d1:f4:c2:64:1f:7e:2d:b3:1a
          53:05:a1:80:54:47:3f:87:a9:8b:30:ab:7e:98:42:c6:d9:5a:de:0f:6d:e5:58:05:2e:86:ef:5c:0f:b7:cb:78
          19:8d:fb:cc:16:53:34:d2:54:79:cb:27:95:36:37:c1:d5:59:ff:29:a9:5a:19:fc:70:94:3c:ac:dc:26:3c:d6
          f9:c1:70:a1:e0:29:c1:33:33:26:a0:c3:05:10:7e:65:bb:cf:de:0f:9a:37:18:b1:ff:32:75:7b:1a:27:d3:fd
          ab:58:c1:be:95:6e:81:4a:8b:8b:47:e8:11:34:56:5e:ec:91:71:c9:74:82:ba:3b:0e:a1:5b:45:b6:93:20:fe
          fd:6b:b8:9e:09:5f:69:34:27:bf:ea:89:ee:19:68:db:4d:a3:1f:ab:c6:e9:55:14:32:b4:db:1c:4e:b2:fd:73
          d8:9c:fb:1d:11:36:9d:64:0c:68:5c:df:7c:7f:86:86:3f:aa:f6:bb:ef:eb:27:fc:7a:17:8b:45:95:50:63:48
          2d:96:a5:18:80:7a:39:3a:59:7b:d9:2b:5f:b5:59:43:61:a3:5a:c7:3c:b9:d9:d1:8a:44:9a:fc:b1:1d:54:88
          3a:18:27:94:be:2f:d6:9c:80:0d:10:16:e8:62:59:af:4d:c1:42:8c:e4:e3:16:44:78:97:dd:e9:8d:e9:30:32
          dc:70:f8:92:89:61:ae:e7:4d:bf:65:b6:88:15:30:08:9b:77:e5:1a:13:a8:89:ae:e4:f3:b5:e4:60:c8:cb:73
          18:be:15:40:7f:1f:c8:d6:9f:9a:a2:23:da:5e:05:5e:63:f5:d4:12:61:4b:e7:d9:f9:b7:3f:6a:d1:01:d3:bb

Commitment:
a[0]   =  56:7c:80:f6:a4:dd:0a:d6:b0:bc:5d:ca:5c:b5:4d:fc:6e:73:01:ce:ad:9e:26:41:b2:a9:64:8e:c6:53:99:3e
          74:4a:a2:9b:bd:f8:52:e7:be:2b:83:94:61:00:07:c1:33:fe:4b:68:0f:b4:c9:26:a7:1b:81:34:51:5d:55:91
          83:91:e1:4b:81:f3:b9:8e:36:f8:1d:b1:e3:89:b6:26:5a:32:03:d3:f7:8d:6c:d9:3f:e8:b4:70:8b:d4:31:e8
          10:fe:82:e1:95:20:20:93:12:5a:2a:a6:9d:d6:bf:1b:9c:66:9b:1d:f2:a5:65:ce:37:5b:b3:9e:ac:6c:09:60
          0a:05:02:e2:fa:97:43:75:82:46:c8:39:08:df:0d:6c:1c:66:d8:da:48:4c:ea:65:cd:86:c7:c1:45:35:08:86
          e6:df:37:d7:61:3b:2a:15:4c:5a:c2:48:9b:5b:a4:00:5a:cb:04:b1:d4:db:86:a4:17:8a:49:fb:bf:27:5b:43
          56:b4:81:de:21:92:58:f9:94:81:ee:3d:85:dc:91:3a:38:06:59:75:63:ce:dc:c5:33:42:d5:18:fc:be:ef:8b
          d7:7a:64:2f:c2:24:77:a7:d6:e9:d3:79:a2:74:e7:e2:e0:5b:66:ac:5f:f5:b0:0f:e3:a5:bf:5b:50:15:bb:bd
          da:cc:7e:7d:fc:54:5c:b1:59:de:13:2f:47:39:8b:b5:1d:1e:ca:5c:b7:1c:a9:1b:f0:3d:48:58:76:cd:e1:98
          a2:27:71:df:0a:09:c4:a7:c5:64:35:24:9c:51:77:2b:b3:1e:30:54:b8:48:67:38:db:3f:17:64:6a:24:70:ef
          54:aa:20:e9:95:63:66:67:2b:0c:08:27:0e:7d:60:42:70:00:24:4c:9a:f5:9a:bc:f0:29:c6:c5:33:92:cb:65
          16:0f:1d:d9:e4:d8:2a:7e:6f:a5:19:bc:5e:aa:35:33:ca:0f:81:0e:29:8a:d8:d6:c3:35:f5:e6:ad:21:e6:b6
a[1]   =  22:ad:bd:91:cb:25:1c:1d:46:bd:aa:38:47:82:66:80:5b:a8:6c:44:77:8f:9d:64:99:ae:09:cd:21:0b:08:db
          96:0d:96:d9:bf:d3:b2:99:c0:f2:df:d9:a8:6f:4c:a0:d8:60:31:43:4d:e6:ba:24:ba:d9:08:27:9e:0d:28:43
          eb:b6:6e:1b:6f:63:48:54:32:e8:6b:99:d9:32:a4:a4:e3:9a:67:20:91:d3:2e:82:ca:7c:9d:64:67:31:0c:03
          99:68:48:fc:cf:21:0b:6e:31:f9:68:c3:0d:9e:55:cc:a5:ee:66:f2:c4:45:37:8b:e9:f6:c6:b8:60:ee:72:ea
          9c:a9:80:5e:92:45:ee:a9:61:a6:a6:f3:46:db:c5:47:5b:60:7d:1c:19:ec:b2:55:0f:f8:ac:b8:41:62:60:0c
          56:24:4d:e2:bf:cd:0c:31:d8:97:f8:b7:29:2d:4d:e5:5f:05:45:73:a4:d4:2b:0b:1f:3f:5f:6e:91:09:a4:7c
          d2:5d:c0:2f:3d:ef:61:96:ba:7b:12:8d:59:b5:05:cb:02:42:ba:3a:47:8f:6b:d6:44:66:1c:63:c3:64:87:ba
          a5:e3:d0:da:26:5d:33:35:07:f2:30:35:ee:0d:ac:7b:11:b8:f4:9b:d8:4e:f5:ce:f2:42:38:23:dc:1d:74:11
          7f:54:0c:ee:c9:9c:df:fa:3e:38:1a:8c:74:9f:ff:ea:7f:0f:ad:16:39:0d:0f:63:1f:10:bd:7e:79:bd:2b:09
```

6 Qualitätssicherung

Die Qualitätssicherung basiert auf Unit Tests, Use Case Test sowie ein Stress Test. Der Code wurde mit dem statistischen Code Analyse Tool JSLint [22] überprüft und entsprechend angepasst. Für die Formatierung wurde ein Code beautifier [23] verwendet.

6.1 Unit Tests und Use Case Test

Ein Unit Test überprüft einzelne Methoden losgelöst vom Gesamtkontext auf Ihre Korrektheit, dabei können pro Methode mehrere Test vorhanden sein, abhängig von den Ausführungspfade der Methode. Ein Use Case Test hingegen testet einen vorgegeben Ablauf in einem Kontext. Die Tests der Bibliothek evotingclient.js sind in der Datei evotingclient_library_unittest.html implementiert. Die Soll-Werte wurden mit der Java Big-Integer Library (J), mit dem Taschenrechner (T) oder mit dem Damgård-Jurik Simulator (DJS) [16] berechnet. Falls nicht anders möglich wurde die Bildschirmausgabe kontrolliert (B) und als Referenzwert eingetragen. Für die get_sha_256-aes_challenge_test wurde Cryptool [17] für die Überprüfung verwendet.

Methode	Anzahl Test	Beschreibung	Gegen-prüfung
get_messages_test()	3	message_base 6^1, 256^1 und 256^8	T
padding_to_message_length_test()	1	message = 00000001, length = 8	B
get_string_with_leading_zero_test()	2	odd und even	B
to_hex_array_with_leading_zero_test()	1	[0x0,**0xA**,0x7F,0xFF] => [0x00,**0x0A**,0x7F,0xFF]	B
encryption_test()	2	s = 1 (Paillier) und s = 2 (Damgård-Jurik)	DJS und J
random_binary_number_test()	5	-1 , 0, 1, 10 und 1536 Bit	B
bulletin_board_entries_test()	2	s = 1 (Paillier) und s = 2 (Damgård-Jurik)	DJS und J
commitment_test()	2	s = 1 (Paillier) und s = 2 (Damgård-Jurik)	DJS und J
response_e_test()	2	s = 1 (Paillier)und s = 2 (Damgård-Jurik)	DJS und J
response_z_test()	2	s = 1 (Paillier)und s = 2 (Damgård-Jurik)	DJS und J

get_sha_256_aes_challenge_test()	2	Bit-Länge und Cryptool Vergleich	Cryptool, DJS und J
usecase_test()	1	Main Success Scenario	DJS

Tabelle 6: Übersicht der Tests

6.2 Stress Test

Die Datei evotingclient_library_demo.html beinhaltet ein Stress Test. Es werden mehrere Durchläufe simuliert. Die Stimme wird zufällig gewählt, sowie ob die Stimmberechtigte Person cheatet oder nicht. Die Parameter n und r werden bei jedem Durchgang neu berechnet. Es wird überprüft, ob ein Cheat der Stimme durch den Verifier richtig erkannt wird. Nach 500 simulierten Wahlvorgängen mit einer Schlüssellänge von 1536 Bit wurde das folgende Ergebnis erzielt. Dabei wurde immer richtig erkannt, ob die Stimmberechtigte Person gecheatet hat oder nicht.

Anzahl	Ohne Cheat	Mit Cheat	Total
Kandidat 1	60	74	**134**
Kandidat 2	56	57	**113**
Kandidat 3	61	67	**128**
Kandidat 4	65	60	**125**
Total	**242**	**258**	**500**

Tabelle 7: Stress Test Ergebnisse

7 Projektmanagement

7.1 Projektplan

Diese Bachelor-Arbeit ist eine Zusammenarbeit von Sonam Samkang und Paolo Dorigo. Der Richtwert für den Aufwand pro Person beruht auf folgende Schätzung, wobei von einer 50 Stunden Arbeitswoche ausgegangen wird.

Gesamten Arbeitsaufwand

= 2 x Anzahl ECTS Punkte x Anzahl Wochen = 2 x 12 x 16 = **384h**

Die Projektdauer beträgt 16 Wochen, davon 14 Wochen studienbegleitend und 2 Wochen Vollzeit.

Arbeitsaufwand Vollzeit

= 2 Arbeitswochen a 40 Stunden = 2 x 50h = **100h**

Arbeitsaufwand pro Woche studienbegleitend

= (Gesamten Arbeitsaufwand – Arbeitsaufwand Vollzeit) / Anzahl Wochen studienbegleitend
= (384h – 100h) / 14 = **20h**

Das gesamte Projekt wird in einzelne Teilaufgaben aufgeteilt, die sich gut in einer überblickbaren Zeitspanne bearbeiten lassen. Diese Strategie des „Teilen und Herrschens“ bedingt, dass die Arbeitspakete klar definiert, strukturiert und einem Verantwortlichen zugeteilt werden. Die folgenden Zeitangaben für die Arbeitspakete wurden aufgrund von Erfahrungswerten geschätzt.

Die Pakete werden in folgenden Klassen eingeteilt:

Management	Verwaltung von Dokumenten, Infrastruktur und Projekt
Einarbeitung	Einarbeitung in asymmetrischen Kryptosystemen und E-Voting
Anforderungsanalyse	Anforderungsanalyse und Machbarkeitsstudien
Design Entwurf	Design Dokumentation
Implementation	Realisierung des Produkts
Tests	Produkt testen
Qualitätssicherung	Qualitätssicherung von Dokumenten und Code
Projektabschluss	Produktabgabe, Präsentation und Auswertung des Projekts

Übersicht

Klasse	Aufwand pdorigo [h]	Aufwand ssamkang [h]
Management	76	56
Einarbeitung	60	60
Anforderungsanalyse	40	50
Design Entwurf	30	30
Implementation	60	60
Tests	30	40
Qualitätssicherung	20	20
Projektabschluss	70	90
Total	**386**	**406**

Tabelle 8: Gesamtübersicht aller Arbeitspakete

Management

Paket	Bezeichnung	Aufwand pdorigo [h]	Aufwand ssamkang [h]
M01	Vision definieren		2
M02	Infrastruktur und Wartung	18	6
M03	Risikoanalyse	4	4
M04	Disaster Recovery	4	2
M05	Arbeitspakete definieren	16	8
M06	Meilensteine definieren	2	2
M07	Sitzungen (inkl. vor- und nacharbeiten)	32	32
Total [h]	**132**	**76**	**56**

Tabelle 9: Übersicht der Management Arbeitspakete

M01	Vision definieren
Inhalt	Motivation und Ziel der Arbeit beschreiben
Vorbedingungen	Aufgabenstellung
Verantwortlicher	ssamkang
Resultat	Vision Dokument
Zeitaufwand [h]	2

M02	Infrastruktur und Wartung
Inhalt	Arbeitsplatz einrichten, Hard- und Software installieren und konfigurieren
Vorbedingungen	Arbeitsplatz Freigabe
Verantwortlicher	pdorigo und ssamkang
Resultat	Übersichtsdokument der verwendeten Hard- und Software
Termin	22.02.2010
Zeitaufwand [h]	16

M03	Risikoanalyse
Inhalt	Risiken identifizieren: Ausfall von Hard- und Software sowie von Projektteilnehmer. Projektbezogene technische Limitationen von Hard- und Software besonders gut bei der Analyse einbeziehen
Vorbedingungen	Aufgabenstellung
Resultat	Risikoanalyse Dokument
Termin	-
Verantwortlicher	pdorigo und ssamkang
Zeitaufwand [h]	8

M04	Disaster Recovery
Inhalt	Verhalten bei Schadensfälle, sowie deren Präventivmassnahmen
Vorbedingungen	Risikoanalyse
Resultat	Erweiterung des Risikoanalyse Dokuments
Termin	-
Verantwortlicher	pdorigo und ssamkang
Zeitaufwand [h]	8

M05	Arbeitspakete definieren
Inhalt	Arbeitspakete identifizieren, beschreiben, priorisieren, Aufwand schätzen sowie die verantwortliche Person für das Paket zuteilen
Vorbedingungen	Interne Sitzung und Besprechung des Projektablaufs (02.03.2010)
Resultat	Arbeitspakete, Glossar
Termin	Dienstag, den 16.02.2010
Verantwortlicher	pdorigo und ssamkang
Zeitaufwand [h]	24

M06	Meilensteine definieren
Inhalt	Meilensteine identifizieren, beschreiben und zeitlich im Projektplan einordnen
Vorbedingungen	Interne Sitzung und Besprechung des Projektplans (02.03.2010)
Resultat	Meilensteine Dokument, Gantt-Diagramm
Termin	Dienstag, den 16.02.2010
Verantwortlicher	pdorigo und ssamkang
Zeitaufwand [h]	4

M07	Sitzungen (inkl. vor- und nacharbeiten)
Beschreibung	Traktanden vorbereiten, Resultat im Team besprechen, vorzeitig einreichen, nach der Sitzung nacharbeiten, Resultat im Team besprechen
Vorbedingungen	Abmachung der Sitzungstermine
Verantwortlicher	pdorigo und ssamkang
Resultat	Sitzungsprotokoll Historie und Sitzungsprotokolle
Termin	Gemäss vereinbarte Sitzungstermine
Zeitaufwand [h]	64

Einarbeitung

Paket	Bezeichnung	Aufwand pdorigo[h]	Aufwand ssamkang [h]
E01	Einarbeitung: E-Voting		30
E02	Einarbeitung: Damgård-Jurik Kryptosystem	30	
E03	Einarbeitung: Kryptographisch sicherer Zufallszahlengenerator		30
E04	Einarbeitung: Zero Knowledge Proofs	30	
Total	**120**	**60**	**60**

Tabelle 10: Übersicht der Einarbeitungs Arbeitspakete

E01	Einarbeitung: E-Voting
Inhalt	Was ist der Grundgedanken der Technologie, wie hat sie sich entwickelt, welche Absicht ist dahinter, welche Vorteile verschafft die Technologie, welche Risiken könnten entstehen
Vorbedingungen	Aufgabenstellung
Verantwortlicher	ssamkang
Resultat	Theoriedokument E-Voting
Termin	
Zeitaufwand [h]	30

E02	Einarbeitung: Damgård-Jurik Kryptosystem
Inhalt	Was ist der Grundgedanken der Technologie, wie hat sie sich entwickelt, welche Absicht ist dahinter, welche Vorteile verschafft die Technologie, welche Risiken könnten entstehen
Vorbedingungen	Aufgabenstellung
Verantwortlicher	pdorigo
Resultat	Theoriedokument Damgård-Jurik Kryptosystem
Termin	
Zeitaufwand [h]	30

E03	Einarbeitung: Kryptographisch sicherer Zufallszahlengenerator
Inhalt	Was ist der Grundgedanken der Technologie, wie hat sie sich entwickelt, welche Absicht ist dahinter, welche Vorteile verschafft die Technologie, welche Risiken könnten entstehen
Vorbedingungen	Aufgabenstellung
Verantwortlicher	ssamkang
Resultat	Theoriedokument Kryptographisch sicherer Zufallszahlengenerator
Termin	
Zeitaufwand [h]	30

E04	Einarbeitung: Zero Knowledge Proofs
Inhalt	Was ist der Grundgedanken der Technologie, wie hat sie sich entwickelt, welche Absicht ist dahinter, welche Vorteile verschafft die Technologie, welche Risiken könnten entstehen
Vorbedingungen	Aufgabenstellung
Verantwortlicher	pdorigo
Resultat	Theoriedokument Zero Knowledge Proofs
Termin	
Zeitaufwand [h]	30

Anforderungsanalyse

Paket	Bezeichnung	Aufwand pdorigo [h]	Aufwand ssamkang [h]
A01	Anforderungsanalyse E-Voting		10
A02	Anforderungsanalyse Damgård-Jurik Kryptosystem	10	
A03	Anforderungsanalyse Kryptographisch sicherer Zufallszahlengenerator		10
A04	Anforderungsanalyse Sicherheitsaspekte	10	
A05	Machbarkeitsstudie E-Voting		10
A06	Machbarkeitsstudie Damgård-Jurik Kryptosystem	20	
A07	Machbarkeitsstudie Kryptographisch sicherer Zufallszahlengenerator		20
Total	**90**	**40**	**50**

Tabelle 11: Übersicht der Anforderungsanalyse Arbeitspakete

A01	Anforderungsanalyse E-Voting
Inhalt	Requirements Dokumentation: Use Cases und Sequenzdiagramme zu „E-Voting“
Vorbedingungen	Aufgabenstellung und Einarbeitung in E-Voting
Verantwortlicher	ssamkang
Resultat	Fully dressed use case, Sequenzdiagramme
Zeitaufwand [h]	10

A02	Anforderungsanalyse Damgård-Jurik Kryptosystem
Inhalt	Welche Aspekte sind für das Damgård Jurik Kryptosystems relevant.
Vorbedingungen	Einarbeitung Damgård-Jurik Kryptosystem
Verantwortlicher	pdorigo
Resultat	Theoriedokumente Damgård Jurik Kryptosystem
Zeitaufwand [h]	10

A03	Anforderungsanalyse Kryptographisch sicherer Zufallszahlengenerator
Inhalt	Welche Aspekte sind für die Sicherheit eines asymmetrischen Kryptosystems relevant, wie könnte die Sicherheit beeinträchtigt werden.
Vorbedingungen	Einarbeitung kryptographisch sicherer Zufallszahlengenerator
Verantwortlicher	pdorigo
Resultat	Theoriedokumente Sicherheitsaspekte
Zeitaufwand [h]	10

A04	Anforderungsanalyse Sicherheitsaspekte
Inhalt	Welche Aspekte sind für die Sicherheit eines asymmetrischen Kryptosystems relevant, wie könnte die Sicherheit beeinträchtigt werden.
Vorbedingungen	Einarbeitung E-Voting
Verantwortlicher	pdorigo
Resultat	Theoriedokumente Sicherheitsaspekte
Zeitaufwand [h]	10

A05	Machbarkeitsstudie E-Voting
Inhalt	Grundsatzfrage einer Machbarkeit einer E-Voting Lösung in JavaScript
Vorbedingungen	Einarbeitung E-Voting, Anforderungsanalyse E-Voting
Verantwortlicher	ssamkang
Resultat	Risikoabschätzung
Zeitaufwand [h]	10

A06	Machbarkeitsstudie Damgård-Jurik Kryptosystem in JavaScript
Inhalt	Testen und eine Empfehlung erarbeiten über die Machbarkeit des Damgård-Jurik Kryptosytems in JavaScript
Vorbedingungen	Einarbeitung E-Voting, Anforderungsanalyse E-Voting, Theorie Sicherheitsaspekte
Verantwortlicher	pdorigo
Resultat	Machbarkeitsanalyse Dokument Damgård Jurik Kryptosystem in JavaScript und Prototypen Demo
Zeitaufwand [h]	30

A07	Machbarkeitsanalyse: Kryptographisch sicherer Zufallszahlengenerator
Inhalt	Testen und eine Empfehlung erarbeiten über die Machbarkeit eines Kryptographisch sicherer Zufallszahlengenerators in JavaScript
Vorbedingungen	Einarbeitung E-Voting, Anforderungsanalyse E-Voting, Theorie Sicherheitsaspekte, Einarbeitung kryptographisch sicherer Zufallszahlengenerator
Verantwortlicher	ssamkang
Resultat	Theoriedokumente Kryptographisch sicherer Zufallszahlengenerator und Prototypen Demo
Zeitaufwand [h]	30

Design Entwurf

Paket	Bezeichnung	Aufwand pdorigo [h]	Aufwand ssamkang [h]
D01	Design Entwurf E-Voting		10
D02	Design Entwurf Damgård Jurik-Kryptosystem	10	
D03	Design Entwurf kryptographisch sicherer Zufallszahlengenerator		10
D04	Design Entwurf Zero Knowledge Proofs Protokoll	20	
D05	Design Entwurf GUI		10
Total	**60**	**30**	**30**

Tabelle 12: Übersicht der Design Entwurf Arbeitspakete

D01	Design Entwurf E-Voting
Inhalt	Design Dokumente für die Entwicklung
Vorbedingungen	Machbarkeitsstudie E-Voting
Verantwortlicher	ssamkang
Resultat	Flussdiagramme, Klassendiagramme, Interaktionsdiagramme
Zeitaufwand [h]	16

D02	Design Entwurf Damgård Jurik Kryptosystem
Inhalt	Design Dokumente für die Entwicklung
Vorbedingungen	Machbarkeitsanalyse
Verantwortlicher	pdorigo
Resultat	Flussdiagramme, Klassendiagramme, Interaktionsdiagramme
Zeitaufwand [h]	16

D03	Design Entwurf Kryptographisch sicherer Zufallszahlengenerator
Inhalt	Design Dokumente für die Entwicklung
Vorbedingungen	Machbarkeitsanalyse
Verantwortlicher	ssamkang
Resultat	Flussdiagramme, Klassendiagramme, Interaktionsdiagramme
Zeitaufwand [h]	16

D05	Design Entwurf Zero Knowledge Proofs
Inhalt	Design Dokumente für die Entwicklung
Vorbedingungen	Machbarkeitsanalyse
Verantwortlicher	pdorigo
Resultat	Flussdiagramme, Klassendiagramme, Interaktionsdiagramme
Zeitaufwand [h]	32

D06	GUI Design Spezifikation
Inhalt	GUI Design Konzept
Vorbedingungen	Theorie E-Voting, Anforderungsanalyse E-Voting
Verantwortlicher	ssamkang
Resultat	GUI Design Dokument
Zeitaufwand [h]	16

Implementation

Paket	Bezeichnung	Aufwand pdorigo[h]	Aufwand ssamkang [h]
I01	Integration der Komponenten		10
I02	Implementation Damgård Jurik	20	
I03	Implementation Kryptographisch sicherer Zufallszahlengenerator		20
I04	Implementation Zero Knowledge Proofs Protokoll	40	
I05	Implementation GUI		20
I06	Implementation. JSON		10
Total	**120**	**60**	**60**

Tabelle 13: Übersicht der Design Arbeitspakete

I01	Integration der Komponenten
Inhalt	Integration von Damgård-Jurik Kryptosystem, Kryptographisch sicherer Zufallszahlengenerator und Zero Knowledge Proofs Protokoll, JSON
Vorbedingungen	Code der einzelnen Komponenten vorhanden
Verantwortlicher	pdorigo
Resultat	Vollständiges Softwarepaket inkl. Unit Tests
Zeitaufwand [h]	10

I02	Implementation Damgård Jurik Kryptosystem
Inhalt	Korrekte Verschlüsselung der Wahl
Vorbedingungen	Design Entwurf Damgård Jurik Kryptosystem
Verantwortlicher	pdorigo
Resultat	JavaScript Code mit Unit Tests
Zeitaufwand [h]	20

I03	Implementation Kryptographisch sicherer Zufallszahlengenerator
Inhalt	Korrekte Verschlüsselung der Wahl mit einer sicheren Zufallsquelle
Vorbedingungen	Design Entwurf Damgård Jurik Kryptosystem
Verantwortlicher	pdorigo
Resultat	JavaScript Code mit Unit Tests
Zeitaufwand [h]	20

I04	Implementation Zero Knowledge Proofs Algorithmus
Inhalt	Korrekte Überprüfung der verschlüsselten Nachricht
Vorbedingungen	Design Entwurf Zero Knowledge Proofs Algorithmus
Verantwortlicher	pdorigo
Resultat	JavaScript Code mit Unit Tests
Zeitaufwand [h]	20

I05	Implementation GUI
Inhalt	GUI Implementation, eventuell extjs
Vorbedingungen	Design Spezifikation GUI
Verantwortlicher	ssamkang
Resultat	JavaScript, HTML (eventuell extjs)
Zeitaufwand [h]	20

I06	Implementation JSON
Inhalt	Kommunikation mittels JSON zu Server Applikation
Vorbedingungen	Design Entwurf E-Voting
Verantwortlicher	ssamkang
Resultat	JavaScript Code mit Unit Tests
Zeitaufwand [h]	20

Test Arbeitspakete

Paket	Bezeichnung	Aufwand pdorigo[h]	Aufwand ssamkang [h]
T01	Integrationstest		10
T02	Damgård Jurik Kryptosystem Test	10	
T03	Kryptographisch sicherer Zufallszahlengenerator Test		10
T04	Zero Knowledge Proofs Protokoll Test	20	
T05	GUI Test		10
T06	JSON Test		10
Total	**70**	**30**	**40**

Tabelle 14: Übersicht der Test Arbeitspakete

T01	Integration Test
Inhalt	Testspezifikation und Test
Vorbedingungen	Alle Komponenten vorhanden
Verantwortlicher	pdorigo
Resultat	Testprotokoll
Zeitaufwand [h]	10

T02	Damgård-Jurik Kryptosystem Test
Inhalt	Testspezifikation und Test
Vorbedingungen	Damgård-Jurik Kryptosystem Code und Test Bedingungen
Verantwortlicher	Pdorigo
Resultat	Testprotokoll zu Damgård-Jurik Kryptosystem
Zeitaufwand [h]	10

T03	Kryptographisch sicherer Zufallszahlengenerator Test
Inhalt	Testspezifikation und Test
Vorbedingungen	Kryptographisch sicherer Zufallszahlengenerator Code und Test Bedingungen
Verantwortlicher	pdorigo
Resultat	Testprotokoll zu kryptographisch sicherer Zufallszahlengenerator
Zeitaufwand [h]	10

T04	Zero Knowledge Proofs Test
Inhalt	Testspezifikation und Test
Vorbedingungen	Zero Knowledge Proofs Code und Test Bedingungen
Verantwortlicher	pdorigo
Resultat	Testprotokoll zu Zero Knowledge Proofs Protokoll
Zeitaufwand [h]	20

T05	GUI Test
Inhalt	Testspezifikation und Test
Vorbedingungen	GUI Code und Usability Test Spezifikation
Verantwortlicher	ssamkang
Resultat	Usability Test
Zeitaufwand [h]	10

T06	JSON Test
Inhalt	Testspezifikation und Test
Vorbedingungen	JSON Code und Test Bedingungen
Verantwortlicher	ssamkang
Resultat	Testprotokoll zu JSON
Zeitaufwand [h]	10

Qualitätssicherung

Paket	Bezeichnung	Aufwand pdorigo[h]	Aufwand ssamkang[h]
Q01	Qualitätssicherung der Dokumente		20
Q02	Qualitätssicherung des Codes	20	
Total	**40**	**20**	**20**

Tabelle 15: Übersicht der Qualitätssicherung Arbeitspakete

Q01	Qualitätssicherung der Dokumente
Inhalt	Dokumente nach Inhalt, Form und sprachliche Gleichstellung prüfen. Einen Gegenleser organisieren.
Vorbedingungen	Dokumente inhaltlich vollständig
Verantwortlicher	ssamkang
Resultat	Revidierte Dokumente
Zeitaufwand [h]	20

Q02	Qualitätssicherung des Codes
Beschreibung	Code Überprüfung mit Test Tools
Vorbedingungen	Test erfolgreich abgeschlossen, Unit Test
Verantwortlicher	pdorigo
Resultat	Revidierter Code
Zeitaufwand [h]	20

Projektabschluss

Paket	Bezeichnung	Aufwand pdorigo[h]	Aufwand ssamkang [h]
P01	Auswertung der Zeiterfassung	2	
P02	Management Summary		10
P03	Poster		8
P04	Abstract	4	
P05	Titelblatt		2
P06	Diplomarbeitsbroschüre		4
P07	Präsentation der Bachelor-Arbeit	8	8
P08	Mündliche Abschlussprüfung	8	8
P09	Dokumentation drucken	4	4
P10	Glossar		4
P11	Erstellen Abgabe-CD	2	
P12	Persönlicher Bericht	2	2
P13	Dokumentation vervollständigen	40	40
Total	**160**	**70**	**90**

Tabelle 16: Übersicht der Projektabschluss Arbeitspakete

P01	Auswertung der Zeiterfassung
Beschreibung	Arbeitsaufwand Soll- zu Ist-Wert vergleich
Vorbedingungen	Zeiterfassung vollständig
Verantwortlicher	pdorigo
Resultat	Analyse, Erkenntnisse und Grafik
Zeitaufwand [h]	2

P02	Management Summary
Beschreibung	Projektzusammenfassung für das Management
Vorbedingungen	Technischer Bericht vollständig
Verantwortlicher	ssamkang
Resultat	Management Summary Dokument
Zeitaufwand [h]	10

P03	Poster
Beschreibung	Poster für die Arbeitsausstellung
Vorbedingungen	Technischer Bericht und Management Summary fertig
Verantwortlicher	ssamkang
Resultat	Poster
Zeitaufwand [h]	8

P04	Abstract
Beschreibung	Abstract der Arbeit
Vorbedingungen	Technischer Bericht und Management Summary fertig
Verantwortlicher	Pdorigo
Resultat	Abstract
Zeitaufwand [h]	4

P05	Titelblatt
Beschreibung	Titelblatt
Vorbedingungen	Technischer Bericht und Management Summary fertig
Verantwortlicher	ssamkang
Resultat	Titelblatt
Zeitaufwand [h]	2

P06	Diplomarbeitsbroschüre
Beschreibung	Broschüre für die Präsentation der Diplomarbeit
Vorbedingungen	Technischer Bericht und Management Summary fertig
Verantwortlicher	ssamkang
Resultat	Broschüre
Zeitaufwand [h]	4

P07	Präsentation der Bachelor-Arbeit
Beschreibung	Präsentation der Bachelor-Arbeit
Vorbedingungen	Arbeit fertig
Verantwortlicher	pdorigo und ssamkang
Resultat	Präsentation
Zeitaufwand [h]	16

P08	Mündliche Abschlussprüfung
Beschreibung	Mündliche Abschlussprüfung
Vorbedingungen	Präsentation der Bachelor Arbeit und Vertiefung des Stoffes
Verantwortlicher	ssamkang und pdorigo
Resultat	Mündliche Abschlussprüfung
Zeitaufwand [h]	16

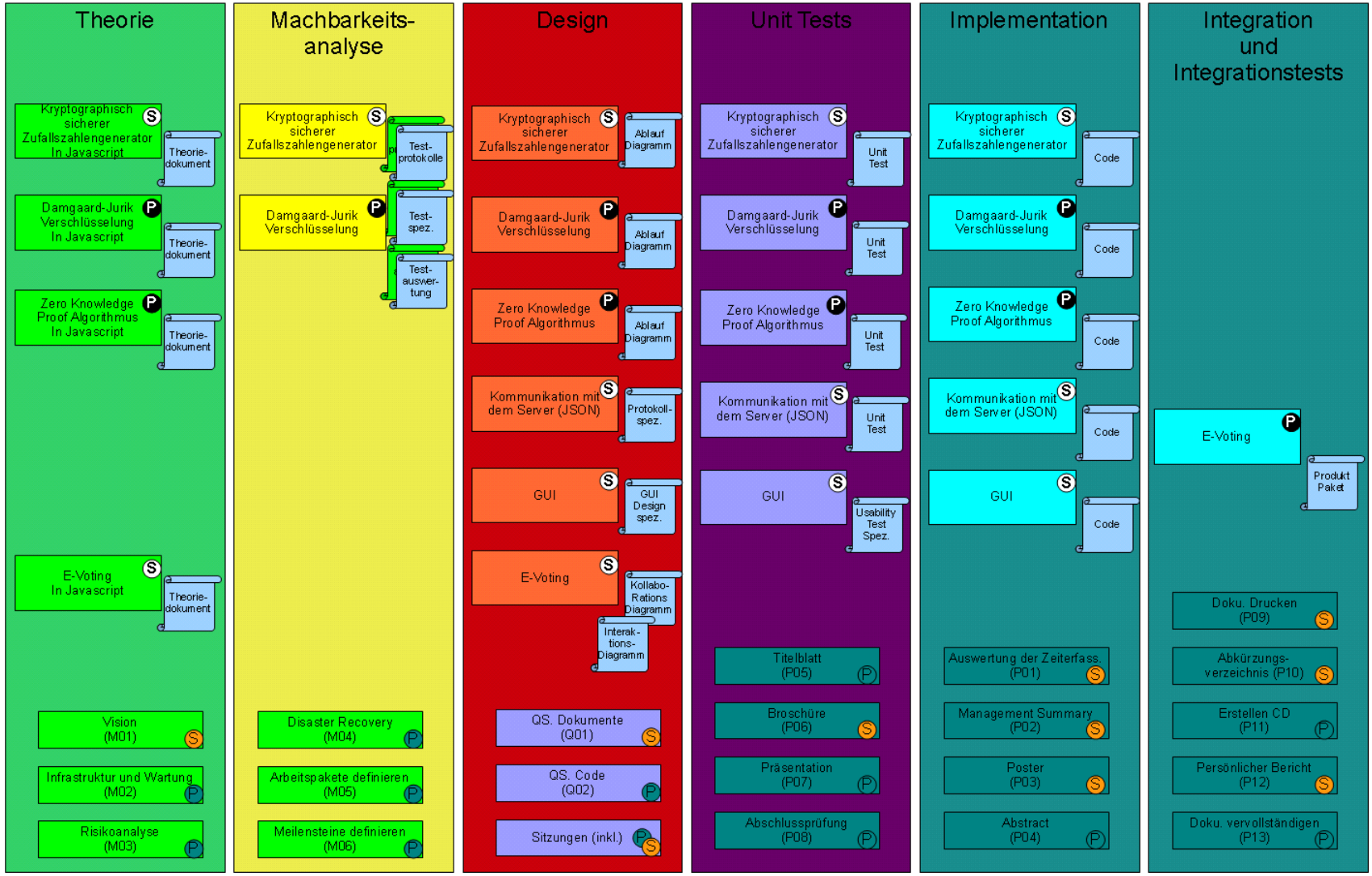
P09	Dokumentation drucken
Beschreibung	3 Exemplare Ausdrucken
Vorbedingungen	Alle Dokumente vollständig und Qualitätssicherung
Verantwortlicher	ssamkang und pdorigo
Resultat	3 facher Ausdruck
Zeitaufwand [h]	8

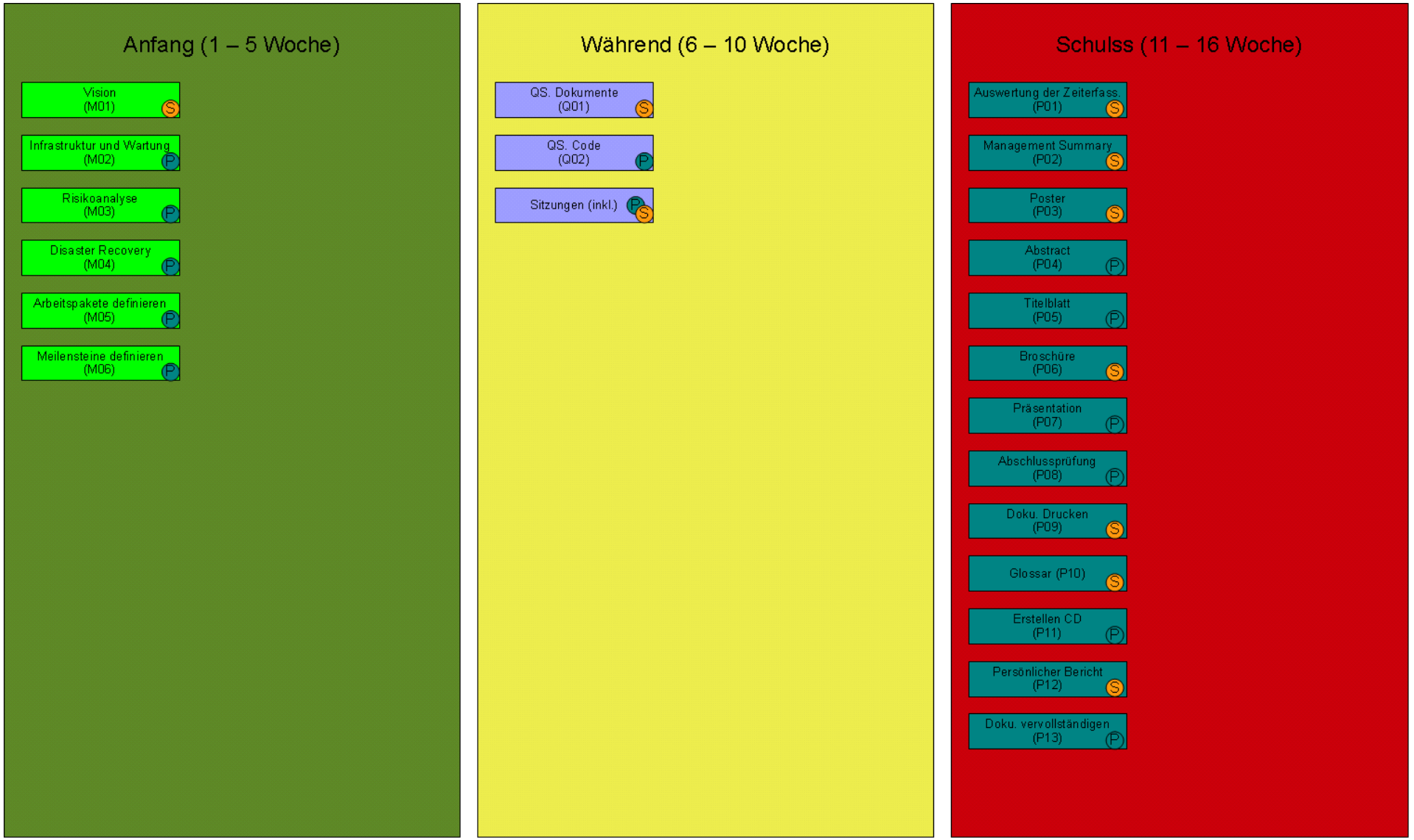
P10	Glossar
Beschreibung	Begriffserklärung
Vorbedingungen	Keine
Verantwortlicher	ssamkang
Resultat	Bergrissindex
Zeitaufwand [h]	4

P11	Erstellen Abgabe-CD
Beschreibung	CD brennen und verfassen
Vorbedingungen	Alle Dokumente fertiggestellt
Verantwortlicher	pdorigo
Resultat	3 x Datenträger
Zeitaufwand [h]	2

P12	Persönlicher Bericht
Beschreibung	Persönlicher Bericht
Vorbedingungen	Arbeit abgeschlossen
Verantwortlicher	ssamkang pdorigo
Resultat	Persönlicher Bericht
Zeitaufwand [h]	4

P13	Dokumentation vervollständigen
Beschreibung	Dokumentation vervollständigen
Vorbedingungen	-
Verantwortlicher	ssamkang pdorigo
Resultat	Vollständige Dokumentation
Zeitaufwand [h]	80


Theorie
Kryptographisch sicherer Zufallszahlengenerator In Javascript
Theorie-dokument
Damgaard-Jurik Verschlüsselung In Javascript
Theorie-dokument
Zero Knowledge Proof Algorithmus In Javascript
Theorie-dokument
E-Voting In Javascript
Theorie-dokument
Vision (M01)
Infrastruktur und Wartung (M02)
Risikoanalyse (M03)
Machbarkeits-analyse
Kryptographisch sicherer Zufallszahlengenerator
Test-protokolle
Damgaard-Jurik Verschlüsselung
Test-spez.
Test-auswer-tung
Disaster Recovery (M04)
Arbeitspakete definieren (M05)
Meilensteine definieren (M06)
Design
Kryptographisch sicherer Zufallszahlengenerator
Ablauf Diagramm
Damgaard-Jurik Verschlüsselung
Ablauf Diagramm
Zero Knowledge Proof Algorithmus
Ablauf Diagramm
Kommunikation mit dem Server (JSON)
Protokoll-spez.
GUI
GUI Design spez.
E-Voting
Kollabo-Rations Diagramm
Interak-tions-Diagramm
QS. Dokumente (Q01)
QS. Code (Q02)
Sitzungen (inkl.)
Unit Tests
Kryptographisch sicherer Zufallszahlengenerator
Unit Test
Damgaard-Jurik Verschlüsselung
Unit Test
Zero Knowledge Proof Algorithmus
Unit Test
Kommunikation mit dem Server (JSON)
Unit Test
GUI
Usability Test Spez.
Titelblatt (P05)
Broschüre (P06)
Präsentation (P07)
Abschlussprüfung (P08)
Implementation
Kryptographisch sicherer Zufallszahlengenerator
Code
Damgaard-Jurik Verschlüsselung
Code
Zero Knowledge Proof Algorithmus
Code
Kommunikation mit dem Server (JSON)
Code
GUI
Code
Auswertung der Zeiterfass. (P01)
Management Summary (P02)
Poster (P03)
Abstract (P04)
Integration und Integrationstests
E-Voting
Produkt Paket
Doku. Drucken (P09)
Abkürzungs-verzeichnis (P10)
Erstellen CD (P11)
Persönlicher Bericht (P12)
Doku. vervollständigen (P13)


Anfang (1 – 5 Woche)
Vision (M01) S
Infrastruktur und Wartung (M02) P
Risikoanalyse (M03) P
Disaster Recovery (M04) P
Arbeitspakete definieren (M05) P
Meilensteine definieren (M06) P
Während (6 – 10 Woche)
QS. Dokumente (Q01) S
QS. Code (Q02) P
Sitzungen (inkl.) P S
Schulss (11 – 16 Woche)
Auswertung der Zeiterfass. (P01) S
Management Summary (P02) S
Poster (P03) S
Abstract (P04) P
Titelblatt (P05) P
Broschüre (P06) S
Präsentation (P07) P
Abschlussprüfung (P08) P
Doku. Drucken (P09) S
Glossar (P10) S
Erstellen CD (P11) P
Persönlicher Bericht (P12) S
Doku. vervollständigen (P13) P

Woche	Arbeitspakete Paolo Dorigo	Arbeitspakete Sonam Samkang	Meilensteine
1	Infrastruktur und Wartung	Infrastruktur und Wartung	1: Kickoff
2	Theorie Dokument Damgård-Jurik Verschlüsselung Risiko Analyse / Disaster Recovery	Theorie Dokument kryptografisch sicherer Zufallsgenerator	
3	Damgård-Jurik Big-Integer Testspezifikation	Testspezifikation kryptografisch sicherer Zufallsgenerator	
4	Arbeitspakete definieren Meilensteine definieren	Arbeitspakete definieren Meilensteine definieren	
5	Zero Knowledge Proof Algorithmus Theorie Dokument Damgård-Jurik Verschlüsselung Testspezifikation	Testspezifikation kryptografisch sicherer Zufallsgenerator	2: Machbarkeitsentscheid
6	Testdurchführung	Testdurchführung	
7	Damgård-Verschlüsselung Sequenzdiagramm Zero Knowledge Proof Protokoll Sequenzdiagramm	Kryptografisch sicherer Zufallsgenerator Sequenzdiagramm Theorie Dokument Kommunikation mit dem Server (JSON)	
8	Damgård-Verschlüsselung Unit Test Zero Knowledge Proof Protokoll Unit Test	GUI Design Spezifikation	
9	Damgård-Verschlüsselung Code	GUI Design Test Spezifikation	
10	Damgård-Verschlüsselung Code Zero Knowledge Proof Protokoll Code	GUI Design Implementation	3. Zero-Knowledge Proof
11	Zero Knowledge Proof Protokoll Code	GUI Design Implementation	
12	Qualitätssicherung des Codes	Qualitätssicherung des Codes	
13	Abstract, Titelblatt, Präsentation, Mündliche Prüfung vorbereiten	Abstract, Titelblatt, Präsentation, Mündliche Prüfung vorbereiten	
14	Abkürzungsverzeichnis Qualitätsprüfung der Dokumentation Auswertung der Zeiterfassung	Abkürzungsverzeichnis Qualitätsprüfung der Dokumentation Auswertung der Zeiterfassung	
15	Persönlicher Bericht Erstellung der CD	Persönlicher Bericht Erstellung der CD	4: Abgabe

7.2 Meilensteine

Woche Nr.	Datum	Bezeichnung	Ziel
1	22.02.2010	Kickoff Meeting	Präsentation der Aufgabenstellung
5	23.03.2010	*Go* or *no Go* ?	Machbarkeitsentscheid
10	20.04.2010	Zero Knowledge Proofs lösen	Entscheid für die Implementierung des Zero Knowledge Proofs Algorithmus
16	16.06.2010	Abgabe	Abgabe der Arbeit

7.3 Entwicklungswerkzeuge

Kategorie	Name	Version	Lizenz
Betriebssysteme	Ubuntu	9.04	GNU GPL
Office	OpenOffice.org	3.0	GNU
	Sunbird	0.9	MPL
	Lightning	0.9	MPL
Programmierung	Eclipse	3.2	EPL
	Java Development Kit	5.0	Sun License
	Aptana Studio	2.0.3	APL
	JSCoverage	0.4	GNUs
	Subversive	0.7.8	EPL
Webbrowser	Firebug	1.4.0	BSD
	Firefox	3.0.17	MPL
	Chrome	5.0.307.7	Google License
	Opera	10.10	Opera License
Internet	Skype	2.1.0.81	Skype License
	Thunderbird	2.0.0.23	MPL
E-Books	SelfHtml	8.1.2	Franzi's GmbH

7.4 Risikomanagement

Personalausfall

ID	Name	Ausfall-zeit in Tage	Eintrittswahr-scheinlichkeit	Auswirkungen	Massnahmen
R1	Leichter Krankheitsfall	1 bis 3	hoch	tief	M1
R2	Schwerer Krankheitsfall	ab 7	tief	hoch	M2
R3	Verspätung	1	hoch	tief	M1
R4	Abwesenheit des Betreuers	1	tief	mittel	MX

Arbeitsplatz oder Geräteausfall

ID	Name	Ausfall-zeit in Tage	Eintrittswahr-scheinlichkeit	Auswirkungen	Massnahmen
R10	Datenverlust	1	hoch	hoch	P1, P2, P3, M4 oder M5
R11	Systemverlust	bis 3	mittel	hoch	P1, P3, M5
R12	Arbeitsplatz-verlust	ab 1	tief	tief	M6
R13	Serververlust	1	tief	mittel	M1 und M3 an Service Desk

Präventive Massnahmen

ID	Name	Aufwand [h]	Betroffene Personen	Vorgehen
P1	System Checkup	2	Projekt-mitarbeiter	• System auf Viren prüfen • Harddisk auf Fehler prüfen • Arbeitsspeicher testen
P2	Daten sichern	1	Projekt-mitarbeiter	• Automatisiert Projektdaten auf Memory Stick sichern
P3	System sichern	1	Projekt-mitarbeiter	• Harddisk Spiegelung durchführen

Massnahmen

ID	Name	Aufwand [h]	Betroffene Personen	Vorgehen
M1	Telefonische Meldung	1	Projektmitarbeiter	• Datenbackup und SVN Synchronisation vornehmen • Sofort alle Beteiligte anrufen • Die Nummern sind aus der Personalien Liste zu entnehmen • Grund und Ausfalldauer melden
M2	Notfallzettel den Angehörigen geben	1	Projektmitarbeiter	• Eine Stellvertretende Person soll die Meldung gemäss M1 vornehmen
M3	E-Mail Meldung	1	Projektmitarbeiter	• Durch E-Mail benachrichtigen
M4	Daten Recovery	1	Projektmitarbeiter	• Lokale Projektdaten von Memorystick laden
M5	System Recovery	1	Projektmitarbeiter	• Acronis [24] Images zurückspielen
M6	Arbeitsplatz wechseln	Bis 3	Projektmitarbeiter	• Nach Absprache einen neuen provisorischen Arbeitsplatz abmachen • Zusätzlich M1 oder M3
M7	Memorystick austauschen	Bis 2	Projektmitarbeiter	• Daten durch Memorystick austauschen

MX	Massnahmen durch den Betreuer	-	Projektmitarbeiter und Betreuer	• -

7.5 Wichtige Termine

Gemäss der Agenda der Fachhochschule Rapperswil sind folgend Termine zu berücksichtigen:

Von	Bis	Betroffen	Beschreibung
12.07.09	20.12.09	Betreuer	Die Betreuer erfassen ihre Arbeiten im AVT-Tool.
	21.12.09	Betreuer	Die Arbeiten werden unter https://avt.hsr.ch um 9 Uhr veröffentlicht.
	20.01.2010	Studenten	Die Studierenden können sich um 1 Arbeit mit Priorität 1, um 3 Arbeiten mit Priorität 2 und um 3 Arbeiten mit Priorität 3 bewerben.
	02.05.10	Abteilungsvorsteher	Zuteilung durch den Abteilungsvorsteher
	21.02.2010	Studenten	Einrichten der Arbeitsplätze in den Labors
	22.02.2010	Studenten	Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch die Betreuer.
	Mai 2010	Studenten	Fotoshooting. Genauere Angaben erteilt die Rektoratsassistentin rechtzeitig.
	11.06.2010	Studenten	Die Studierenden geben die Kurzbeschreibung für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Die Studierenden erhalten vorgängig vom Studiengangsekretariat die Aufforderung zur Online-Erfassung der Kurzbeschreibung. Die Studierenden senden per Email das A0-Poster zur Prüfung an ihren Examinator/Betreuer.

			Vorlagen stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
	16.06.2010	Betreuer	Der Betreuer/Examinator gibt das Dokument mit der korrekten und vollständigen Kurzbeschreibung zur Weiterverarbeitung an das Studiengangsekretariat frei.
	18.06.2010	Studenten	Abgabe des Berichtes an den Betreuer bis 12.00 Uhr. Fertigstellung des A0-Posters bis 12.00 Uhr. Abgabe des Posters im Studiengangsekretariat 6.113.
21.06.10	31.06.2010	Studenten	Mündliche BA-Prüfung
	25.06.2010	Studenten	HSR-Forum, Vorträge und Präsentation der Bachelor- und Diplomarbeiten, 13 bis 18 Uhr
	01.10.2010	Studenten	Bachelorfeier und Ausstellung der Bachelorarbeiten (am Nachmittag)

7.6 Auswertung der Zeiterfassung

Die geplante Arbeitszeit konnte mit einer Abweichung von 5% - 10% eingehalten werden.

Woche	Paolo Dorigo		Sonam Samkang	
	Ist [h]	Soll-Ist-Abweichung [h]	Ist [h]	Soll-Ist-Abweichung [h]
1	21	+1	23	+2
2	21	+1	20	0
3	32	+12	21.5	+1.5
4	22	+2	31.5	+11.5
5	30	+10	23.5	+3.5
6	29	+9	23	+3
7	6	-14	23.5	+3.5
8	13	-7	23	+3
9	26	+6	25	+5
10	17	-3	27	+7
11	17	-3	22	+2
12	32	+12	32.5	+12.5
13	30	+10	25	+5
14	23	+3	23	+3
15	28	+8	25	+5
16	46	-4	36	-14
17	30	-20	37.5	-12.5
Total	**423**	**+23**	**442**	**+42**

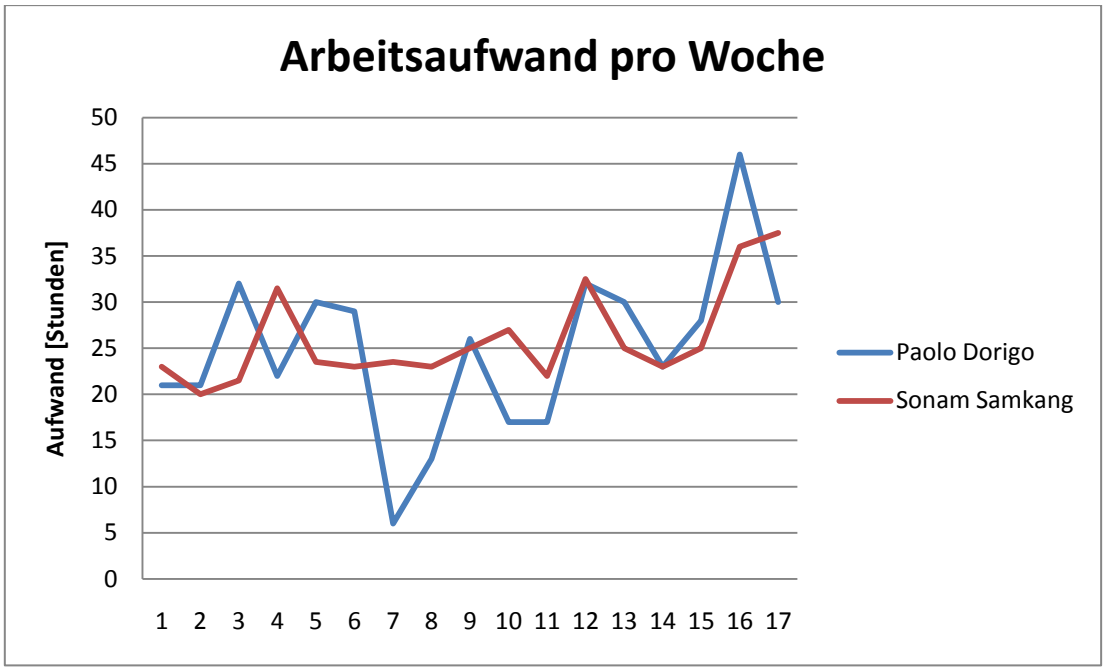


Abbildung 33: Arbeitsaufwand proWoche

7.7 Erfahrungsbericht

7.7.1 Paolo Dorigo

Das Interesse für kryptografische Algorithmen wurde bei mir bei den Modulen Internet Sicherheit 1

und Internet Sicherheit 2 geweckt, sowie die Übungen mit dem E-Learning-Programm Cryptool fasziniert mich sehr.

Erfahrungen in JavaScript und HTML konnte ich in einem vorherigen Praktikum sammeln, das ich zwischen 2009 und 2010 absolviert habe. Aus diesem Grund fiel meine erste Wahl, bei der Vergabe der Bachelor Arbeit, auf die Arbeit "E-Voting Web Client in JavaScript" von Herr Dr. Prof. Andreas Steffen.

Der Einstieg in die komplexe Thematik sowie das Einarbeiten in die wissenschaftlichen Publikationen war aufwendig, JavaScript und HTML Erfahrungen konnte ich schon früher sammeln, so dass der Einstieg in diesen Technologien unproblematisch war.

Nach den ersten funktionierenden Prototypen und die Produkt Analyse konnten die Fortschritte bei der Zwischenpräsentation gezeigt werden.

Diverse Anpassungen wurden vorgenommen, einerseits die Erweiterung des Zero Knowledge Proofs Protokolls zum Non Interactive Zero Knowledge Proofs Protokolls und anderseits die Kommunikation mit dem Server mittels JSON.

Aus dieser Arbeit habe ich gelernt, dass eine gute Planung entscheiden ist für das Gelingen eines Projekts. Auch das Einplanen von Zeitreserven ist für die Einhaltung des Zeitplans entscheidend. Ich möchte für die Zukunft mitnehmen, dass bei der Risiko Analyse die waren Risiken zu berücksichtigen sind, nämlich jene, die die Machbarkeit gefährden. Eine angemessene Strategie, die falls technische Limitationen vorkommen das Projekt weiterbringen kann.

7.7.2 Sonam Samkang

Diese Bachelorarbeit hat mich von Anfang an mein Interesse geweckt. Einerseits ist diese Arbeit ein sehr aktuelles Thema, andererseits bietet sie mir die Gelegenheit mein theoretisches Wissen in den Modulen Internettechnologie sowie auch Internet Sicherheit in Praxis umzusetzen.

Zu Beginn des Semesters habe ich mich intensiv mit JavaScript auseinandergesetzt da das Verständnis und die Anwendung dieser Skriptsprache essenziell für die Arbeit sind. Im Nachhinein bin ich froh darüber dass ich diese Vorarbeit geleistet habe. Denn die Komplexität des Damgård -Jurik Kryptosystems und Zero Knowledge Proofs Theorie war aus meiner Sicht grösser als wir es erahnen konnten. Dasselbe mussten wir bei der Generierung von echten Zufallszahlen feststellen.

Durch Recherchen im Internet gelang uns aber etwas Licht im Dunkeln zu bringen. Wir konnten mit guter Zusammenarbeit viele Punkte gemeinsam lösen. Bei der Zwischenpräsentation waren wir mit der vom Damgård-Jurik Kryptosystems fast fertig. Um bessere Noten zu erhalten, haben wir beschlossen auch den optionalen Teil der Arbeit zu realisieren was uns trotz mehrmaligen Hürden am Ende gelang. Bei der Gesamt-Implementierung haben wir mehr viel Zeit gebraucht als geplant. Immer wieder hatten wir neue Ideen, die wir unbedingt umsetzen wollten, was zu mehr Aufwand führte. Ich bin auch hier froh darüber wir ohne grosse Komplikationen alle unsere Ideen und natürliche alle Aufgaben inkl. den optionalen Teil realisieren konnten. Nebst der Erfüllung der Aufgabenstellung konnten wir viele Erfahrungen mit JavaScript sowie auch ihre Anwendung im Bereich E-Voting Sicherheit sammeln.

Die Wahl des Dozenten, Herrn Steffen, war eine gute Entscheidung. Die wöchentliche Sitzung mit ihm fand ich sehr gut. Wir konnten so sicherstellen dass wir allfällige Fehler bereits frühzeitig bereinigen konnten. Aus meiner Sicht war er für uns eine grosse Unterstützung. Nebst Betreuung gab er uns wertvolle Inputs und Verbesserungsvorschläge. An dieser Stelle möchte ich mich bei Herrn Steffen und sein Assistent Tobias Brunner für die gute Zusammenarbeit bedanken.

Die Zusammenarbeit mit Paolo Dorigo war von Anfang an sehr angenehm. Wir konnten uns sehr gut ergänzen und haben uns wo möglich gegenseitige Unterstützung geleistet. Zu keiner Zeit habe ich das Gefühl gehabt dass ich alleine die Arbeit schreibe. Eine weitere Zusammenarbeit mit ihm würde ich auf jeden Fall ohne zu zucken akzeptieren.

8 Management Summary

Ausgangslage

Ein E- Voting System muss speziellen Anforderungen genügen. Einerseits muss der Staat sicherstellen, dass jede stimmberechtigte Person nur eine einzige Stimme abgibt, andererseits muss sich der Stimmbürger darauf verlassen können, dass seine Wahl berücksichtigt und anonym ausgewertet wird. Dieses Projekt befasst sich mit der Aufgabe, ein solch sicheres, stabiles und transparentes Wahlsystem anzubieten. Das Ziel der Arbeit war es, einen Web Client zu entwickeln, der mit Hilfe einer Big- Integer Library und JavaScript Funktionen das für E- Voting geeignete Damgård- Jurik Kryptosystem implementiert. Weiter galt es, die Verfügbarkeit von echten Zufallszahlen in JavaScript abzuklären. Die generierten Datensätze waren in der JavaScript Object Notation (JSON) an den Server weiterzuleiten. In einem optionalen Teil der Arbeit sollte die Wohlgeformtheit der verschlüsselten Stimmzettel mittels eines Zero-Knowledge Proofs nachgewiesen werden können.

Vorgehen/ Technologien

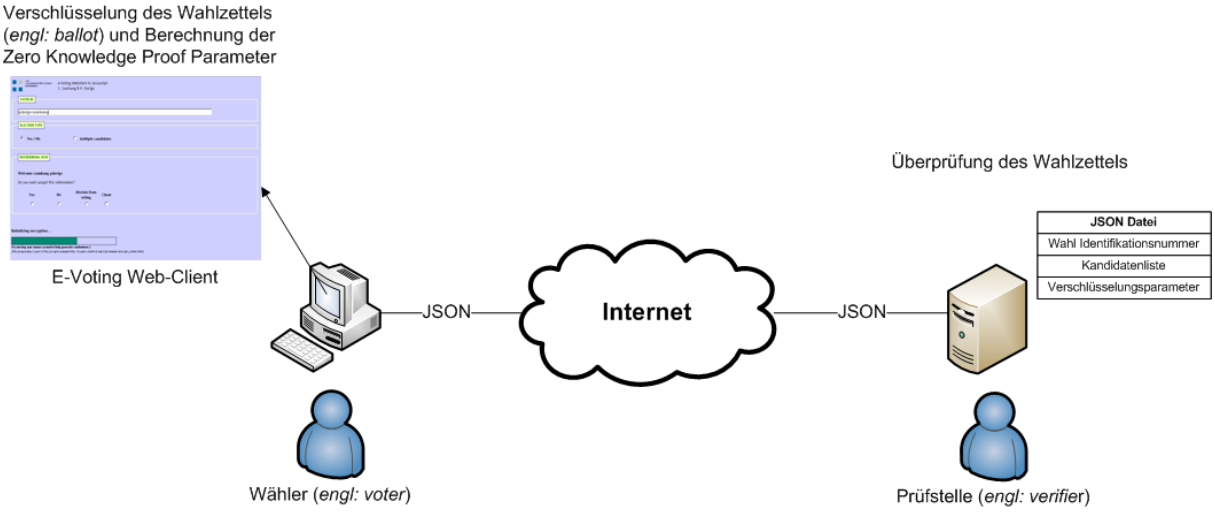


Abbildung 34: Kommunikation zwischen Client und Server

Die clientseitige Verschlüsselung der Wahlzettel mit JavaScript erlaubt es jedermann, den lesbaren JavaScript Code auf Korrektheit zu überprüfen. Der verwendete asymmetrische Damgård- Jurik Verschlüsselungsalgorithmus, hat die homomorphe Eigenschaft, dass die Multiplikation chiffrierter Werte gleich der verschlüsselten Summe der entsprechenden

Klartexte entspricht. Da nur das aufkumulierte Schlussresultat der Abstimmung entschlüsselt wird, ist dadurch die Anonymität der einzelnen Stimmen gewährleistet. Um die Gültigkeit eines verschlüsselten Wahlzettels zu überprüfen, wird ein Zero-Knowledge Proofs Protokoll verwendet. Da JavaScript nur mit Zahlen kleiner als 53 Bit rechnen kann, ist für die Berechnungen eine JavaScript Big- Integer Library notwendig. Es wurden verschiedene Bibliotheken getestet und im Detail ausgewertet. Ebenfalls wurde die Möglichkeit echte Zufallszahlen plattformunabhängig zu generieren abgeklärt.

Die angewendeten Technologien sind HTML, JavaScript und CSS. Desweiteren wurden diverse externe Libraries angewendet. Diese sind jsbn, jsbn2, jQuery, jsCrypto, jssha256, jsaes und json.

Ergebnis

Die Wahl der Big-Integer Library ist auf *jsbn.js* der Stanford University gefallen und für die Generierung von Zufallszahlen auf *jscrypto.js*, welche durch die Auswertung von Mausbewegungen Entropie gewinnt. Der implementierte JavaScript Client verschlüsselt erfolgreich Stimmzettel mittels des Damgård- Jurik Verschlüsselungsalgorithmus und leitet das Chiffrat mit zusätzlichen, für den Zero-Knowledge Proofs Protokoll notwendigen Parametern im JSON Format an den Server weiter, der die empfangenen Daten auf ihre Wohlgeformtheit prüft.

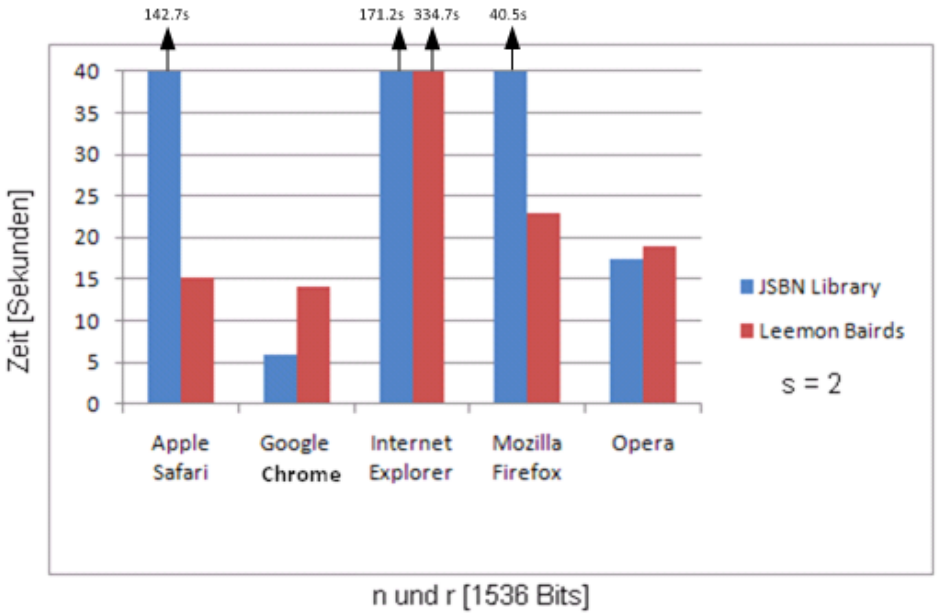


Abbildung 35: Browser Performance Test

Die Performance Tests haben ergeben dass der Web-Client am optimalsten mit dem Google Chrome Browser läuft. Die Benutzung von anderen Browsern dauert zu lange und ist somit ungeeignet.

Bei der Verwendung eines 1024 Bit RSA Modulus benötigt der schnelle Google Chrome Browser nur 1.7 Sekunden für die Damgård- Jurik Verschlüsselung und zusätzlich 8.8 Sekunden für die Generierung der Zero-Knowledge Proofs Protokoll notwendigen Parametern.

9 Anhang

9.1 Abkürzungsverzeichnis

AES	Advanced Encryption Standard
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator
JSON	JavaScript Object Notation
PRN	Pseudo Random Number
PRNG	Pseudo-Random Number Generator
SHA	Secure Hash Algorithm
TRN	True Random Number

9.2 Tabellenverzeichnis

9.3 Abbildungsverzeichnis

9.4 Literatur-/Quellenverzeichnis

[1] Bundesamt für Statistik. Von BFS Registerharmonisierung Amtlicher Katalog der Merkmale: http://www.bfs.admin.ch/bfs/portal/de/index/news/00/00/05/08.parsys.23948.downloadList.83598.DownloadFile.tmp/72dstimmundwahlrecht.pdf

[2] Reusser, H. (2009), Zero-knowledge Proofs für n-te Potenzen. Rapperswil: Hochschule Rapperswil.

[3] International, E. (2009). ECMAScript Language Specification.

[4] Thorsten Kleinjung, K. A., & Dag Arne Osvik, H. t. (2010). Factorization of a 768-bit RSA Modulus.

[5] Lenstra, A. K. (2004). Key Lengths - Contribution to The Handbook of Information Security.

[6] Wikipedia.Von Binäre modulo-Exponentiation: http://de.wikipedia.org/wiki/Binäre_Exponentiation

[7] Montgomery, "Modular Multiplication Without Trial Division. Math. Computation.

[8] Jurik, I. B. (2000). A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System. BRICS Report Series.

[9] Jens Groth, Non-interactive Zero-Knowledge Arguments for Voting

[10] WebHits

http://www.webhits.de

[11] Applying UML and Patterns von Craig Larman (ISBN: 0-13-148906-2).

[12] Wu, T. BigIntegers and RSA in JavaScript. jsbn.js and jsbn2.js.

[13] Baird, L. (kein Datum). BigInt.js.

[14] Schneidermann. (2010). 04-UI2-2010-WahrnehmenErinnernDenkenHandlen.pdf. Hochschule für Technik Rapperswil.

[15] Google. (kein Datum). V8 Benchmark Suite - v3. http://v8.googlecode.com/svn/data/benchmarks/v3/run.html

[16] Steffen, P. D.

http://security.hsr.ch/msevote/damgardjurik

[17] Cryptool

http://www.cryptool.de

[18] Stanford University jsCrypto

http://crypto.stanford.edu/sjsc/jscrypto.pdf

[19] Theorie Zufallszahlengenerator

http://de.wikipedia.org/wiki/Zufallszahlengenerator

[20] Theorie dev/random

http://de.wikipedia.org/wiki//dev/random

[21] Miller-Rabin-Test

http://de.wikipedia.org/wiki/Miller-Rabin-Test

[22] JSLint (static code analysis tool)

http://www.jslint.com/lint.html

[23] Online JavaScript beautifier

http://jsbeautifier.org/

[24] Acronis® True Image

http://www.acronis.de/homecomputing/products/trueimage/

9.5 Messprotokolle

9.5.1 Korrektheit der Algorithmen

jsbn.js / jsbn2.js

[Addition]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa +
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
15e20e8d3890c945d8f4f1df515f2e38f8ec0f18329d5f1f3a634ded9dbd5cb6a7ea8100e91424ed6d0d976b188d6891ff2f50bdf032dee591ec
5303f9c77968a0468c8d18621fc0b18e9d3c460a4ac0704d228cdf04d4990e1d1249fb4954e8667f7ac628f9db9d0ce8ac8e48a9f1b8e2313e7d
6d456aeffedd641de9b36f94f2684c3854a660573f4fa91ca1e0c99512aab3ca368a725d9eb67eb4ca35bcf77b057762e659e98605e160db6fb5
7b4684d7c19e4e07d8cc604cbd10fcb2a340f

MD5 Hash: d00c1884081f9d91167dc11501873989 / Ausfuehrungszeit: 1 ms

[Subtraktion]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa -
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
5f8425e8c03d4a6f8e2b06974b7cd81bd1ea9a8484c6b1163f99421da24705aed47682b6bf9bf5fcef075a0c6ac83943561d9e5511a01835b8d
3643313872bb0b9285b6db48f3a95cae8537d43d04fb62378cfea1c5606f5dcf86c1cfc86c28d1beb5a7cfa4bd69a843fd79dedfa4b5fe908200d
e22bda90080e50d22cade2ff72928223da4d48698194918e73cbd2ec04c39196203559282c71fcfe8c8849f20022447c384c50cffc8bf780d09b
4b0422a41c007dbf0d64e24846719ad15545

MD5 Hash: 5d15d57804f77bb6323b6d579b42d6d4 / Ausfuehrungszeit: 0 ms

[Multiplikation]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa *
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
6eceaa20e5c3fa1929afc50d2489a06ffacfae3c6f3cd88cda4582ef08a8d66cf2121087e4fad153be84892b386c0e10d1b14b2a915c4d0efcc1b4
c32ad2d274504f2e838629498a5548abaa4bd468e8df3b2241fc7dd0b9e88519baa6c8096db38aa2462bf47f1028805adc8130437032a9dc66
605900af942f6d57deb19475b267abc181eb047de245b8ee5623c113666a4f9028654fbf0775ff270c1d2051fb03f85c7a5f8d44323375714221
1f726634d8f312b91f0754f361cac5492d32e971250af0a8d1a99222134795b5b59f172fc2e2a3e80fe3332dc62e5cc2f09f997f278fce4d9f5b8e
0135c15367db226b0cd5c00a8c1be25aae80d363cfb6b6de229526cccc950eca23628cd48a07cb98c47025207945bd72883c77d4547ba9465e

0b16a440dc5f1c5e7a1b925f199961bac718c2d142f50684ce25bdacb0597f4eb11e2acac10160c54586d425678df6d91d0b16dc213893dcf42
ddbf73f5e6660b57d5815c100b5703ce891fa0450595db26587cca74b44464126639c4d12

MD5 Hash: ba503ca81b1e8967fea94e4574740942 / Ausfuehrungszeit: 1 ms

[Division]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa *
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 = 1

MD5 Hash: c4ca4238a0b923820dcc509a6f75849b / Ausfuehrungszeit: 2 ms

[Potenzieren]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa ^ 3 =
a8cf31370273f3a7de72f9bd2449d4664c71ee19b7b160c0f7250e50c2cb77ceb97c337f845bbf30a26ffe7674edcaf5c9a1ec35d5b425f253da6
77e7156734e928239b56bd2e861f9ecd96258123920f702cea4b969ff8b2f8c7eded6ab595673183a1f54fcfad09cb4bef8347c728865e4e35d2
0dd7309d6b3a427c721dbb82bd516bb9b9b5e241bc380da2d97af74be9e336dbe26e5fcb99aba690e81a93611c089651452a737f59df2c4c13
2fa04f0546d6d40d37c658c89525ba184e641539c0d0d3cfa2c75720987b8440791847e3695c290f1ccd5df094e90dd424de7c5014ad5dfda72
434a67f8258ecea4db850a627e218e9e5ad7378c6ea19928523780464a485417d9d1051bc41f35d11cf5e901f22804c7480b3b6cd2edf47e20
6541b71331f1fcd525973fdbfd46da1a11cfb0ece586f69e8342a30f67217326fc234761d5794ab12a34fa43cabf453a355556d791403f56d53f7
5ebc622db754ca3f697e565084bcc91a1ddb793a338b49a51bce7a062bb86f9f7f1dec72152bc3cc06b4e19ecf0413d5c9102ef1470bb1d00cb
34458c98d82ca02647c435c4cc62f0c742e882d9e97f90111233bfcc10b963ebc852129f7271bffef99d49d23d7e6c7f6277f131cc1e81f48d897
e2514990a4d5eb775ebc791f867123fb32a918d8a4cf88544d2d4843268e32d320a7072bdb3de24c132406d113dd54d13c2baf09c5f7c695f4
22281e862702d7e750f886fc01e831e01add152e4484feaae7df8ac7d94835b0991560954846a34256c37db8e41eaa8da53a133e1b372a768

MD5 Hash: e4aec1b032a8f81a224cd67799686b1b / Ausfuehrungszeit: 1 ms

[Modulo]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa mod
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
5f8425e8c03d4a6f8e2b06974b7cd81bd1ea9a8484c6b1163f99421da24705aed47682b6bf9bf5fcef075a0c6ac83943561d9e5511a01835b8d
3643313872bb0b9285b6db48f3a95cae8537d43d04fb62378cfea1c5606f5dcf86c1cfc86c28d1beb5a7cfa4bd69a843fd79dedfa4b5fe908200d
e22bda90080e50d22cade2ff72928223da4d48698194918e73cbd2ec04c39196203559282c71fcfe8c8849f20022447c384c50cffc8bf780d09b
4b0422a41c007dbf0d64e24846719ad15545

MD5 Hash: 5d15d57804f77bb6323b6d579b42d6d4 / Ausfuehrungszeit: 0 ms

[Modulo-Potenzieren]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa ^

7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 mod
7cf7f30a4151b791960f178dad365aa5efba491ef2045b85f40c31d9a1ea00410361742f5e617de517100d51c03143c0c79ed73399b4b680357
3d4eb9d4abdb93f6fd6af8cb5a332e06f9f9d88bb4f878cc980455be92018fdb219c886ed9bcb5c3acda200ea5a1ccce28c4075aee536909d164
be17edb10983c0228c6af4f10d70a1272a4a84ab1679201de987c7536ed10a3ec7382402f45aa3bc96676f73c525cdac32690cb1f6545dfa411
57ae9725ed807f70843bea008fc6b9efaa736d =
36d87009377bf54602a7e5b3ffdfa754c26a6d002642cf7d5e002568bc51bf8d5d07d9e9068a8a8922672bc148bde4c16dff5205b7c056e4bd1
353a0c4e3c09349a5a2f3597ec3a226597d7f86856e25929d26d82956c022369b409fc1552a661ffc765262794059f27de703c462c1ed81f38a4
8a81302fd6ab28e5be0f7efba3d8ad7d8719e4e59d49b44184e21347077515e8575f7adb5b82459d95403effc35a805e0bbcfff37e3caa7dcef7
9193fc60673f2271b74c77cb36908fed524a0

MD5 Hash: a2c906914021b87bd9ae0dd8e0717718 / Ausfuehrungszeit: 315 ms

[Grösster gemeinsamer Teiler]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa gcd
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 = 5

MD5 Hash: e4da3b7fbbce2345d7772b0674a318d5 / Ausfuehrungszeit: 4 ms

Leemon Baird

[Addition]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa +
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
15e20e8d3890c945d8f4f1df515f2e38f8ec0f18329d5f1f3a634ded9dbd5cb6a7ea8100e91424ed6d0d976b188d6891ff2f50bdf032dee591ec
5303f9c77968a0468c8d18621fc0b18e9d3c460a4ac0704d228cdf04d4990e1d1249fb4954e8667f7ac628f9db9d0ce8ac8e48a9f1b8e2313e7d
6d456aeffedd641de9b36f94f2684c3854a660573f4fa91ca1e0c99512aab3ca368a725d9eb67eb4ca35bcf77b057762e659e98605e160db6fb5
7b4684d7c19e4e07d8cc604cbd10fcb2a340f

MD5 Hash: d00c1884081f9d91167dc11501873989 / Ausfuehrungszeit: 1 ms

[Subtraktion]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa -
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
5f8425e8c03d4a6f8e2b06974b7cd81bd1ea9a8484c6b1163f99421da24705aed47682b6bf9bf5fcef075a0c6ac83943561d9e5511a01835b8d
3643313872bb0b9285b6db48f3a95cae8537d43d04fb62378cfea1c5606f5dcf86c1cfc86c28d1beb5a7cfa4bd69a843fd79dedfa4b5fe908200d
e22bda90080e50d22cade2ff72928223da4d48698194918e73cbd2ec04c39196203559282c71fcfe8c8849f20022447c384c50cffc8bf780d09b
4b0422a41c007dbf0d64e24846719ad15545

MD5 Hash: 5d15d57804f77bb6323b6d579b42d6d4 / Ausfuehrungszeit: 0 ms

[Multiplikation]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa *
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
6eceaa20e5c3fa1929afc50d2489a06ffacfae3c6f3cd88cda4582ef08a8d66cf2121087e4fad153be84892b386c0e10d1b14b2a915c4d0efcc1b4
c32ad2d274504f2e838629498a5548abaa4bd468e8df3b2241fc7dd0b9e88519baa6c8096db38aa2462bf47f1028805adc8130437032a9dc66
605900af942f6d57deb19475b267abc181eb047de245b8ee5623c113666a4f9028654fbf0775ff270c1d2051fb03f85c7a5f8d44323375714221
1f726634d8f312b91f0754f361cac5492d32e971250af0a8d1a99222134795b5b59f172fc2e2a3e80fe3332dc62e5cc2f09f997f278fce4d9f5b8e
0135c15367db226b0cd5c00a8c1be25aae80d363cfb6b6de229526cccc950eca23628cd48a07cb98c47025207945bd72883c77d4547ba9465e
0b16a440dc5f1c5e7a1b925f199961bac718c2d142f50684ce25bdacb0597f4eb11e2acac10160c54586d425678df6d91d0b16dc213893dcf42
ddbf73f5e6660b57d5815c100b5703ce891fa0450595db26587cca74b44464126639c4d12

MD5 Hash: ba503ca81b1e8967fea94e4574740942 / Ausfuehrungszeit: 1 ms

[Division - Wird nicht unterstuetzt!]

[Potenzieren - Wird nicht unterstuetzt!]

[Modulo]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa mod
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 =
5f8425e8c03d4a6f8e2b06974b7cd81bd1ea9a8484c6b1163f99421da24705aed47682b6bf9bf5fcef075a0c6ac83943561d9e5511a01835b8d
3643313872bb0b9285b6db48f3a95cae8537d43d04fb62378cfea1c5606f5dcf86c1cfc86c28d1beb5a7cfa4bd69a843fd79dedfa4b5fe908200d
e22bda90080e50d22cade2ff72928223da4d48698194918e73cbd2ec04c39196203559282c71fcfe8c8849f20022447c384c50cffc8bf780d09b
4b0422a41c007dbf0d64e24846719ad15545

MD5 Hash: 5d15d57804f77bb6323b6d579b42d6d4 / Ausfuehrungszeit: 1 ms

[Modulo-Potenzieren]
ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa ^
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 mod
7cf7f30a4151b791960f178dad365aa5efba491ef2045b85f40c31d9a1ea00410361742f5e617de517100d51c03143c0c79ed73399b4b680357
3d4eb9d4abdb93f6fd6af8cb5a332e06f9f9d88bb4f878cc980455be92018fdb219c886ed9bcb5c3acda200ea5a1ccce28c4075aee536909d164
be17edb10983c0228c6af4f10d70a1272a4a84ab1679201de987c7536ed10a3ec7382402f45aa3bc96676f73c525cdac32690cb1f6545dfa411
57ae9725ed807f70843bea008fc6b9efaa736d =
36d87009377bf54602a7e5b3ffdfa754c26a6d002642cf7d5e002568bc51bf8d5d07d9e9068a8a8922672bc148bde4c16dff5205b7c056e4bd1
353a0c4e3c09349a5a2f3597ec3a226597d7f86856e25929d26d82956c022369b409fc1552a661ffc765262794059f27de703c462c1ed81f38a4
8a81302fd6ab28e5be0f7efba3d8ad7d8719e4e59d49b44184e21347077515e8575f7adb5b82459d95403effc35a805e0bbcfff37e3caa7dcef7
9193fc60673f2271b74c77cb36908fed524a0

MD5 Hash: a2c906914021b87bd9ae0dd8e0717718 / Ausfuehrungszeit: 712 ms

[Grösster gemeinsamer Teiler]

ded2875e24a4ef668ebd124630b7ddd5b055c603d74e5184f2e7107bbf0e688ca98f4962a86f2269dff0685ef9cf6131a489551a0a6703476bcc
4a3957ff611d5ec8921f9d589b5071e913a0d23a7dde94257c5c0651a8435f64c85e588e0889c1f1836fc4f4c835a96550413c4cb377060e03f2
5b4144c7faf2495863f26e274c8ba2d49259a6eebb4791ac48ec361e97b7671cc46e3f810becf42597f20cb4d83cdd554ef574982d51029be5f9
7fb638101af2af1e4d15738a0bc0b2fdc4aa gcd
7f4e61756467a4f700920baee53b05b9de6b2b7f5287a06eb34dce5e1cc762ddd518c6abe8d32c6cf0e90e528f0727ee4e6bb6c4f8c6eb11b2f
8e6064478356ca5a036b1e8c960baa700c0238e6a2e2870acac71e9fba14d826c5c415c0745fca60628f2caa8f19b252578a34e5268171d05e3
e479156a37f2e3f88637448b27d9f920b0b80c5e8539b3001dd520633292f3d586a438e658df7af7270b69c2c2d81a98d916a923c830c50b1b
155e34b2156bfef2315f3fb09141c54f182c6f65 = 5

MD5 Hash: e4da3b7fbbce2345d7772b0674a318d5 / Ausfuehrungszeit: 10 ms

9.5.2 Tauglichkeit einer Damgård-Jurik Verschlüsselung

Google Chrome	JSBN Library Zeit	Leemon Library Zeit
Grösse von n, r in Bit	Sekunden	Sekunden
100	0.0	0.0
200	0.0	0.0
300	0.1	0.1
400	0.2	0.3
500	0.3	0.7
600	0.5	1.2
700	0.8	1.8
800	1.2	2.8
900	1.7	3.9
1000	2.3	5.3
1100	3.0	7.0
1200	3.9	8.9
1300	5.0	11.6
1400	6.2	14.5
1500	7.6	17.5
1600	9.3	21.4
1700	11.1	25.8
1800	13.0	30.4
1900	15.3	36.1
2000	17.9	42.1
2100	20.6	48.1