Dateiverschlüsselung im Webbrowser:
Entwicklung und Evaluation

Bachelorarbeit-Kolloquium

Wintersemester 16/17

Referent: Carlo Rissmöller

Datum: 21.03.17

Gliederung

- Aufgabenstellung
- Stand der Technik
- Entwurf
- Implementierung
- Vorführung des Prototyps
- Evaluation
- Ausblick
- Literatur

Aufgabenstellung

- Entwicklung und Evaluation eines Software-Prototyps
- Plattformübergreifender Prototyp
- Hybride Verschlüsselung der Nutzdaten
- Integrität der Nutzdaten
- Persistente Speicherung durch Speicher-Server
- Entschlüsselung durch vertrauenswürdige Instanz

Stand der Technik

Whisply:

- Ende 2015
- Webanwendung
- Web Cryptography API als Kryptoschnittstelle
- Doppelte AES-Verschlüsselung
- Zugang zur Datei über Link und Passwort
- File-Hoster sind Dropbox, Google Drive und One Drive

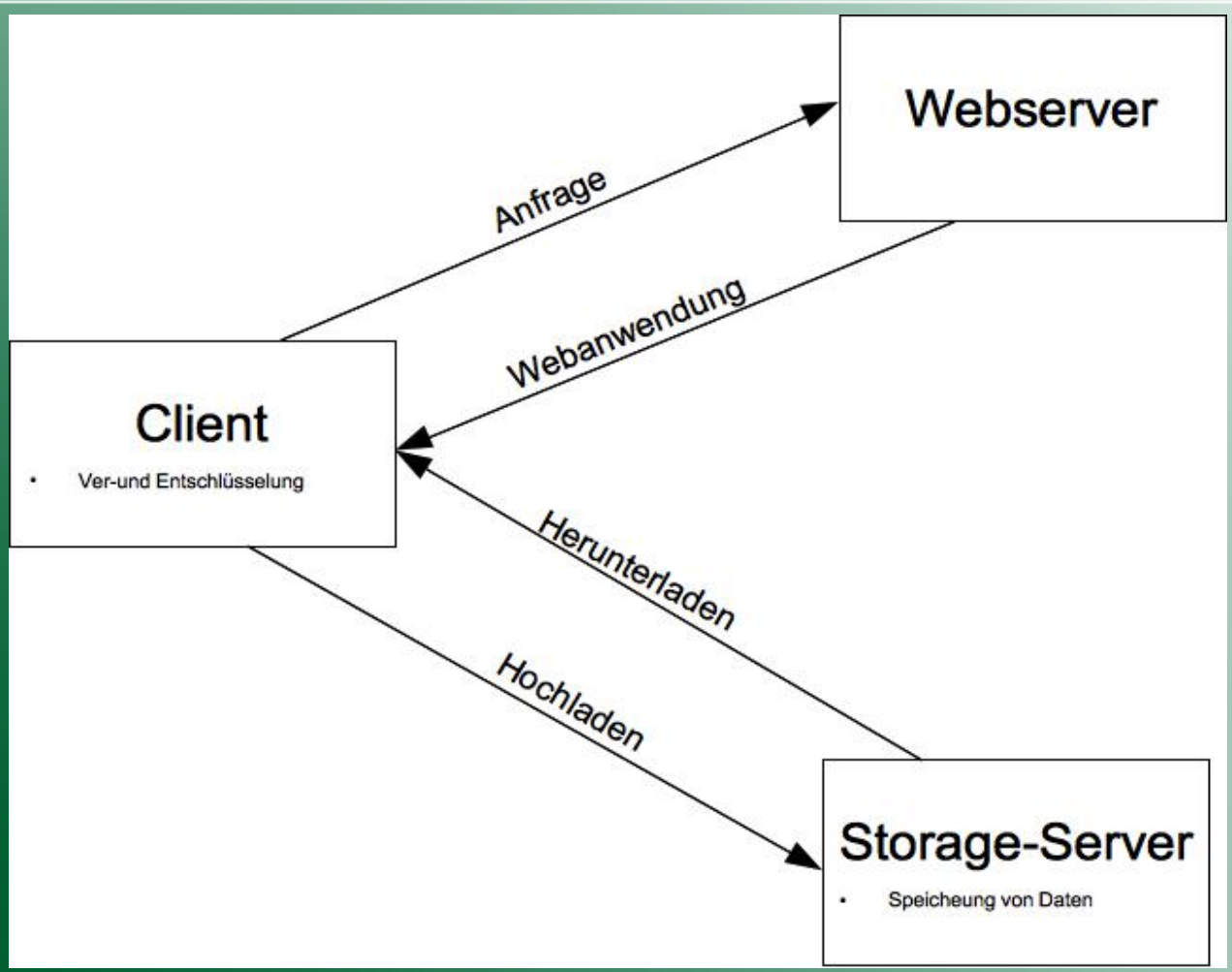
Stand der Technik

Cryptomator:

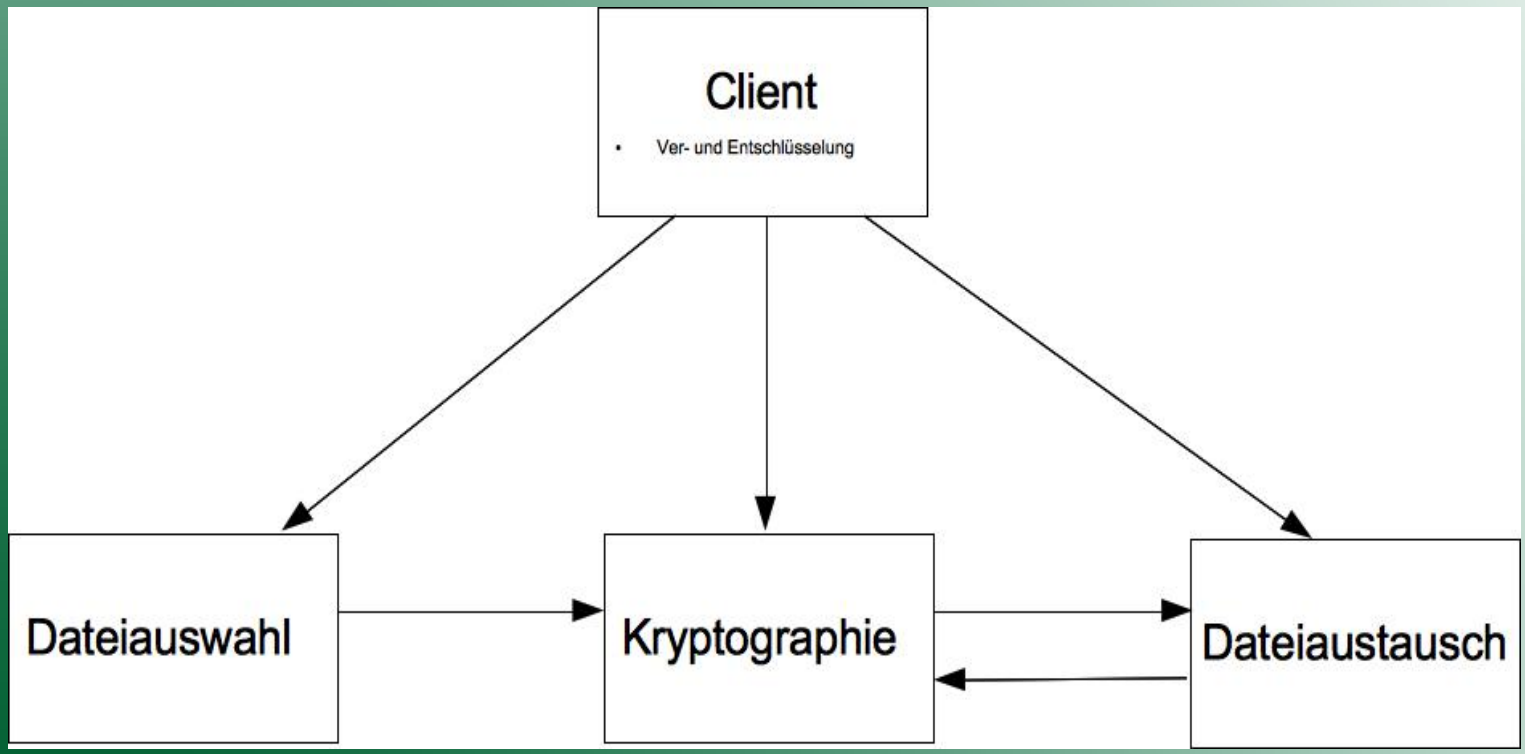
Cryptomator

- Anfang 2016
- Eigenständige Software
- Open-Source
- Schlüsselableitung über Scrypt
- Verschlüsselung über AES mit 256 Bit
- Entschlüsselung über WebDAV
- Unterstützung von Dropbox, Google Drive, One Drive und weitere

Entwurf


Webserver
Anfrage
Webanwendung
Client
Ver-und Entschlüsselung
Herunterladen
Hochladen
Storage-Server
Speicheung von Daten

Entwurf:
Client-Bibliothek



Entwurf: Client-Bibliothek

Komponente zur Dateiauswahl:

- Auf lokale Daten über Dateischnittstelle zugreifen
- Einlesen der Datei durch Chunking
- Eingelesene Chunks werden an die nächste Komponente weitergereicht

Entwurf:
Client-Bibliothek

Komponente für kryptographische Verfahren:

- Zweistufige Kryptoimplementierung
- AES-GCM mit 128 Bit für Verschlüsselung/ Integrität der Nutzdaten
- RSA-OAEP mit 2048 Bit Modul für Verschlüsselung des Dateischlüssels
- Packen der verschlüsselten Daten in eine binäre Datenstruktur

Entwurf:
Client-Bibliothek

Binäre Datenstruktur:

- Dateiname
- Dateigröße
- Dateityp
- Chunknummer
- Verschlüsselter Dateischlüssel
- Initialisierungsvektor
- Authentifizierungs-Tag
- Verschlüsselter Chunk

Entwurf: Client-Bibliothek

Komponente für kryptographische Verfahren:

- Schwachstellen der Datenstruktur
- Entschlüsselung und Packen der Klartextdatei

Komponente zum Dateiaustausch:

- Eingabe ist binäre Datenstruktur aus Komponente für kryptographische Verfahren
- Generische Schnittstelle
- Server kann auf beliebigem Verzeichnis gehostet werden
- Grundfunktionen zum Hoch- und Herunterladen

Implementierung

- Funktionalität der Webapp über JavaScript
- GUI mit HTML und CSS
- Web Cryptography API
- Forge-Framework
- Unterstützung der Verschlüsselung durch Safari, Firefox, Chrome, iOS Safari und Android Chrome
- Unterstützung der Entschlüsselung durch Firefox
- Unterstützung der Schlüsselgenerierung durch Firefox

Implementierung: Verwendete Technologien

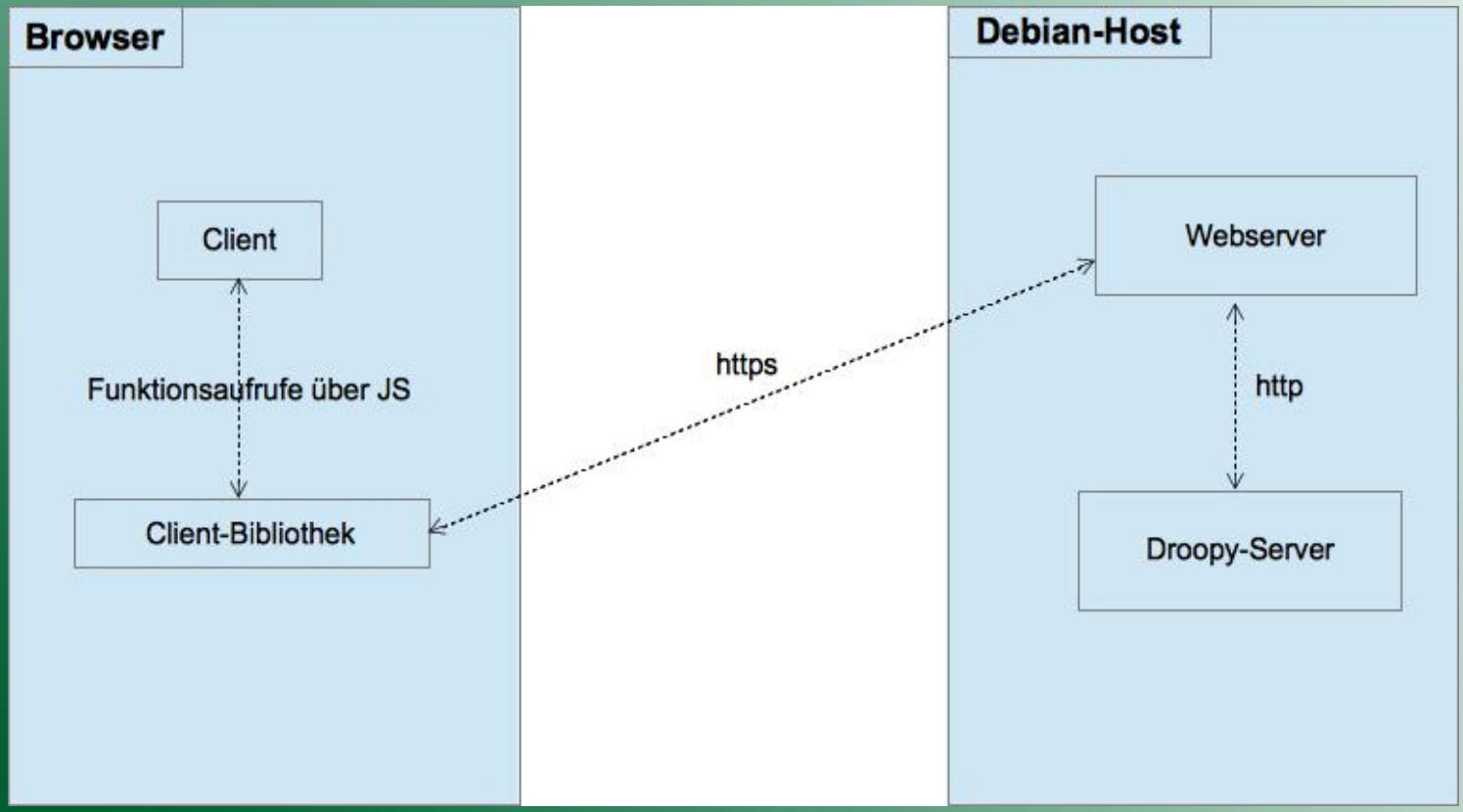
Droopy:

- Einfacher HTTP-Server mit POST und GET
- In Python geschrieben
- Wird auf dem Debian-Host betrieben

XMLHttpRequest:

- Programmierschnittstelle für JavaScript
- Datenaustausch über das HTTP-Protokoll
- Synchron oder asynchron

Implementierung: Architektur des Systems



Implementierung: Hybride Verschlüsselung durch Closure

Hier geht es zum Code!

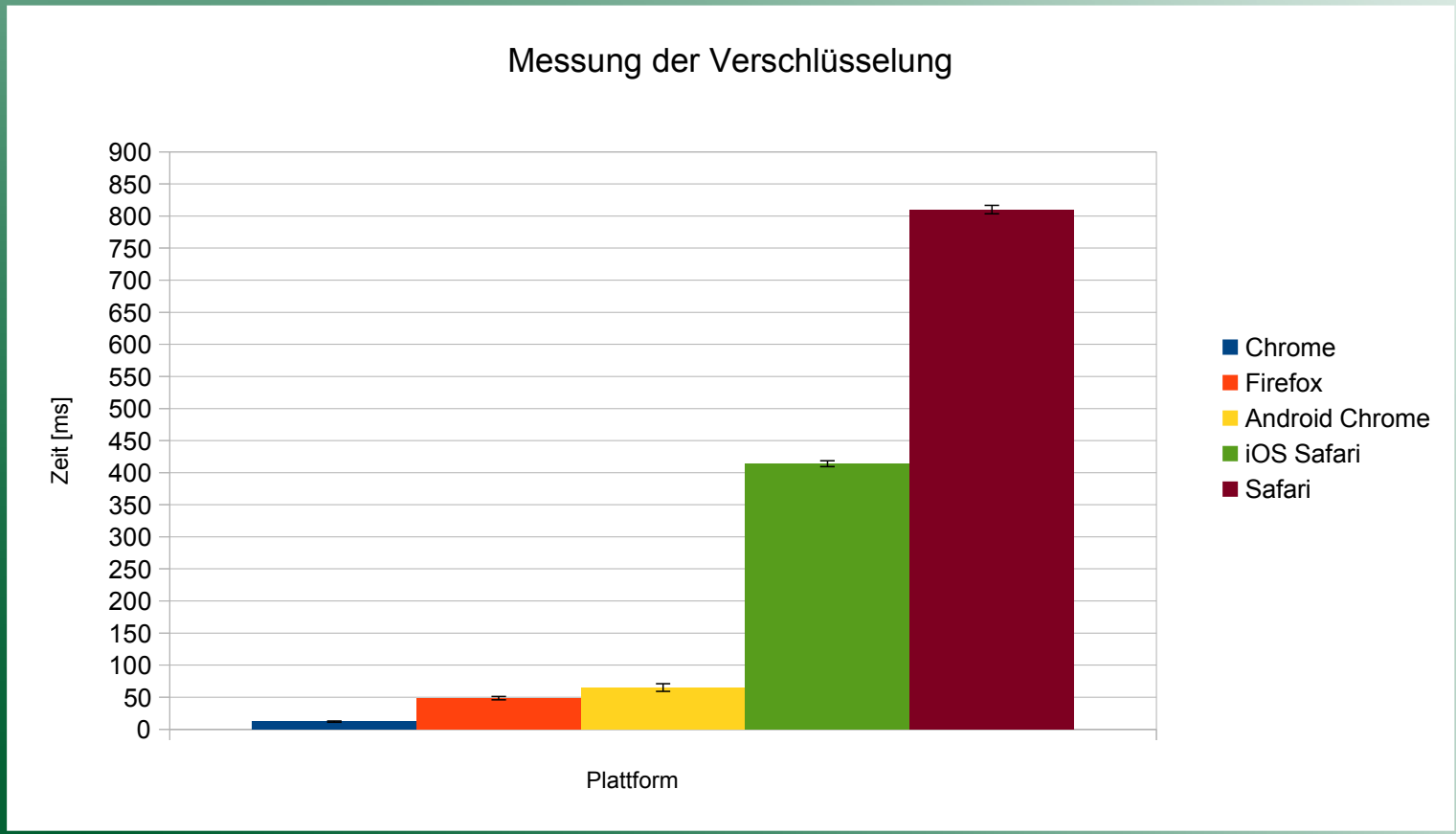
Vorführung des Prototyp

Evaluation: Laufzeitanalyse

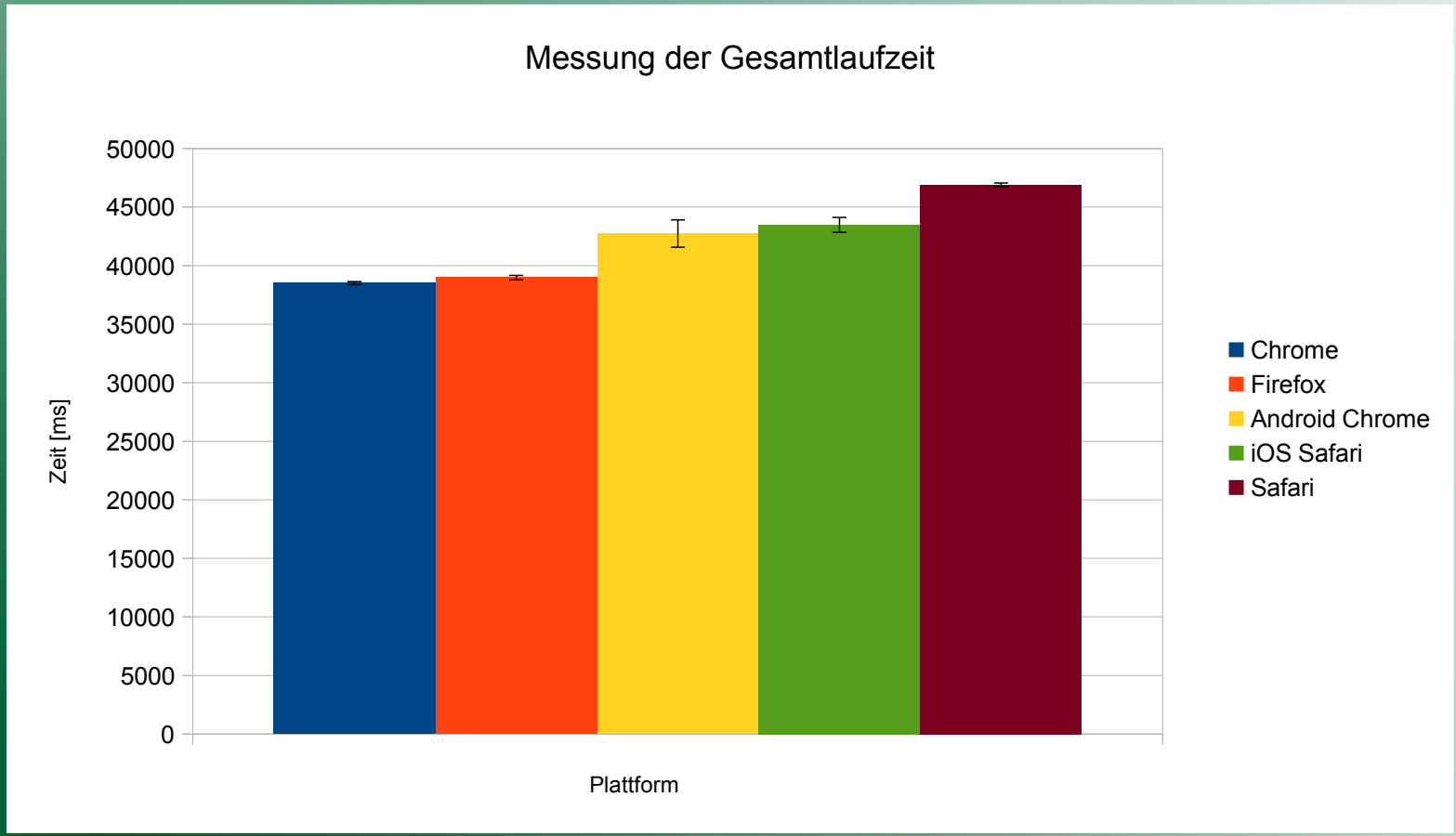
Aufbau des Tests:

- Messungen wurden in unterschiedliche Kategorien unterteilt
- 50 Messwerte pro Kategorie
- Unter den Testplattformen sind *Firefox, Chrome, Safari, iOS Safari und Android Chrome*
- Test wurde in JavaScript geschrieben
- 10 MB Eingabedatei und eine Chunkgröße von 1 MB
- Median und Median der absoluten Abweichungen in Säulendiagramm

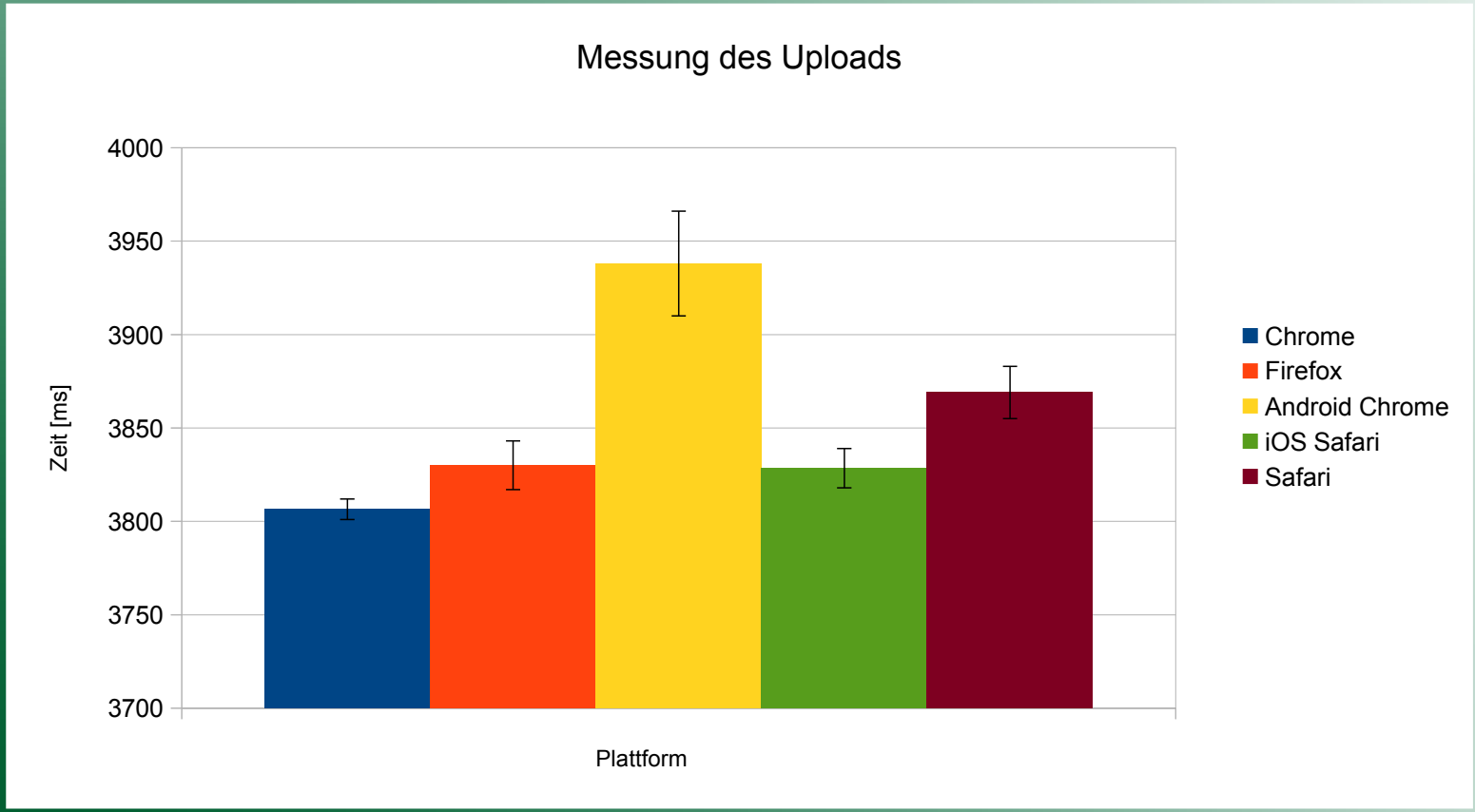
Evaluation: Laufzeitanalyse



Evaluation: Laufzeitanalyse



Evaluation: Laufzeitanalyse



Evaluation: Laufzeitanalyse

Zusammenfassung der Bewertung:

- Verschlüsselung mit Web Cryptography API schneller als mit forge
- Tauglichkeit des Prototypen auf mehreren Plattformen mit akzeptabler Performanz
- Anteil der Verschlüsselung an Gesamtlaufzeit gering

Evaluation: Sicherheitsanalyse

Vertraulichkeit:

- Hybride Verschlüsselung
- Kryptoimplementierungen durch Funktionsabschluss
- Keine Verschlüsselung und Integrität für Metainformationen
- HTTPS übernimmt sichere Kommunikation beim Hochladen

Integrität:

- Integrität des Dateischlüssels und der Nutzdaten
- Keine Integrität für restliche Felder
- Secure-Hosting

Evaluation: Sicherheitsanalyse

Verfügbarkeit:

- Verschlüsselung ist statisch
- Spiegeln der Daten für Backups
- Storage-Backend

Ausblick

Performanz:

- Zwei verschiedene Formate für hochgeladene Dateien
- Nutzung von Worker-Threads

Benutzerfreundlichkeit:

- GUI überarbeiten
- Entschlüsselung auch für den Standardnutzer
- Externes Tool für Entschlüsselung
- Schlüsselverteilung bei gemeinsam genutzten Daten

Literatur

- Thomas Hartmann: Whisply: Dateien im Browser mit Ende-zu-Ende Verschlüsselung versenden. Version: 2015. [Online; Stand 26 Oktober 2016]
- Cryptomator: Cryptomator. [Online; Stand 26. Oktober 2016]
- Pierre Stackp: Droopy- easy file receiving. [Online; Stand 20. Februar 2017]
- Wikipedia: XMLHttpRequest- Wikipedia, Die freie Enzyklopädie. Version: 2017. [Online; Stand 13. März 2017]
- Digital Bazaar: Forge. [Online; Stand 20. Februar 2017]
- Michele Rosica: JavaScript Cryptography: Speed Test. [Online; Stand 20. Februar 2017]
- BSI: Kryptographische Verfahren: Empfehlungen und Schlüssellängen. Version: 2016. [Online; Stand 1. Dezember 2016]

Danke für Ihre Aufmerksamkeit !

Gibt es noch Fragen ?