PEER-TO-PEER DATEISYNCHRONISATION

sicher

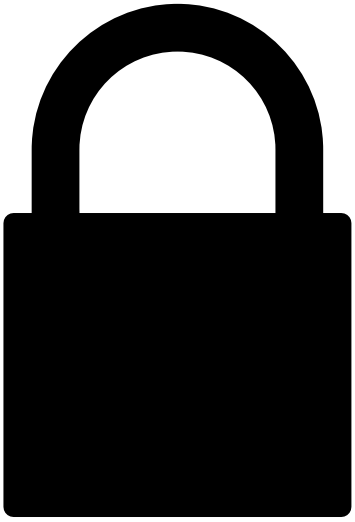
dezentral

ausfallsfrei

open source

ÜBERBLICK

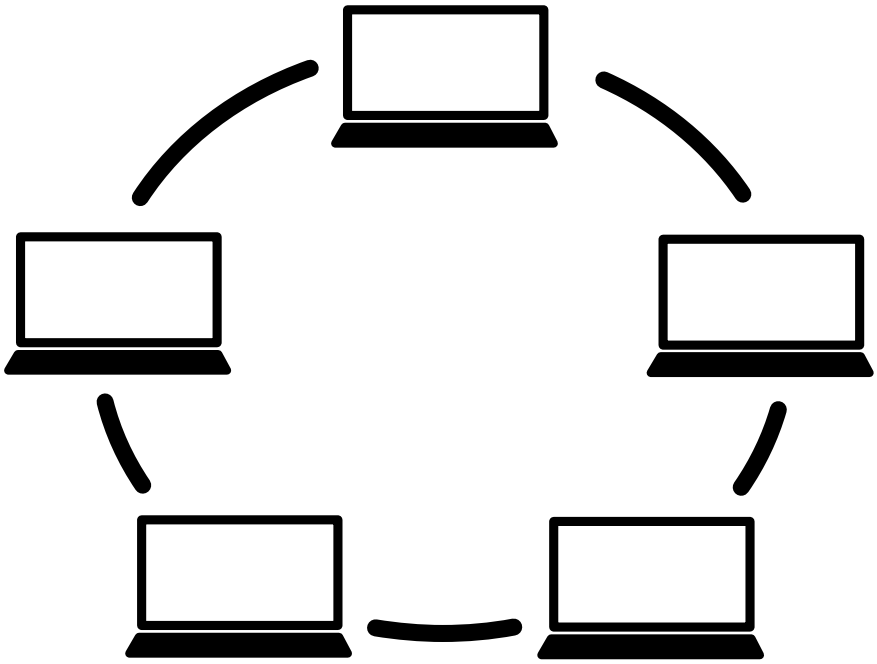
sicher



Starke Verschlüsselung

Alle Daten und Kommunikationskanäle werden mit AES im Modus GCM verschlüsselt. Knacken unmöglich.

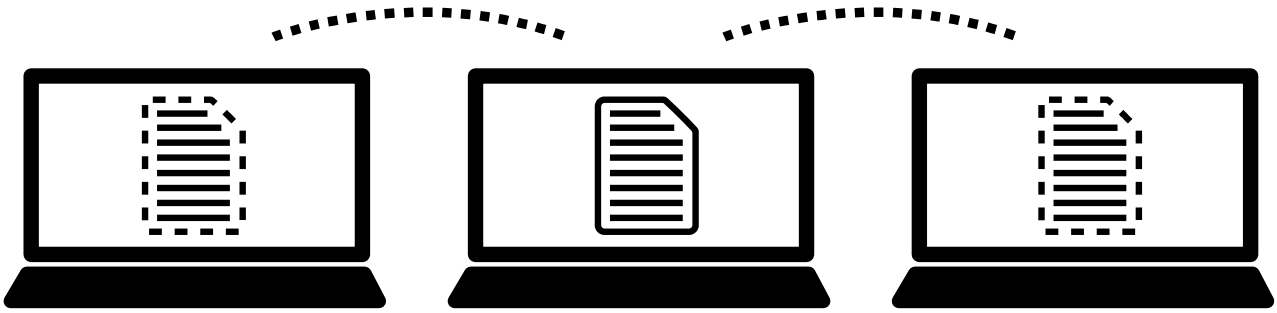
dezentral



Keine zentralen Stellen

Die Geräte kommunizieren über ein Peer-to-Peer-Netzwerk direkt miteinander. Es gibt keine Server oder andere zentrale Stellen, die abgehört werden könnten.

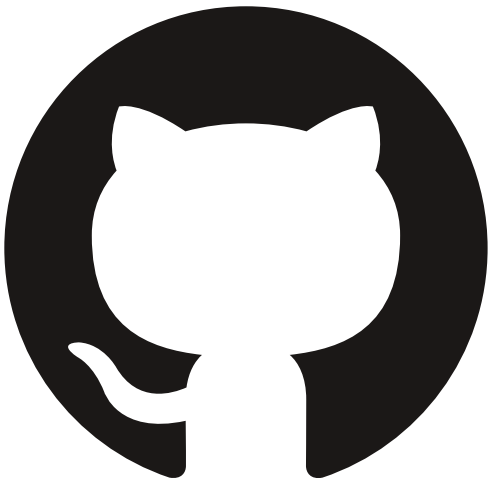
ausfallsfrei



Redundanz

Alle Dateien werden mehrmals auf verschiedenen Geräten gespeichert. Die Nicht-Erreichbarkeit der Dateien aufgrund des Ausfalls des dezentralen Systems ist praktisch unmöglich.

open source

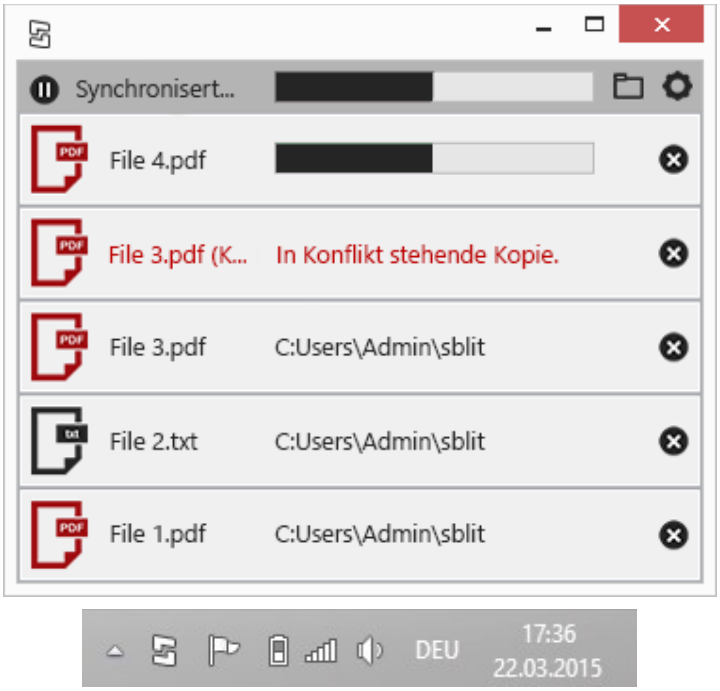


GitHub

Der Quellcode ist unter der GNU Public License in Version 3 lizenziert und auf **github.com/sblit** *öffentlich zugänglich.*

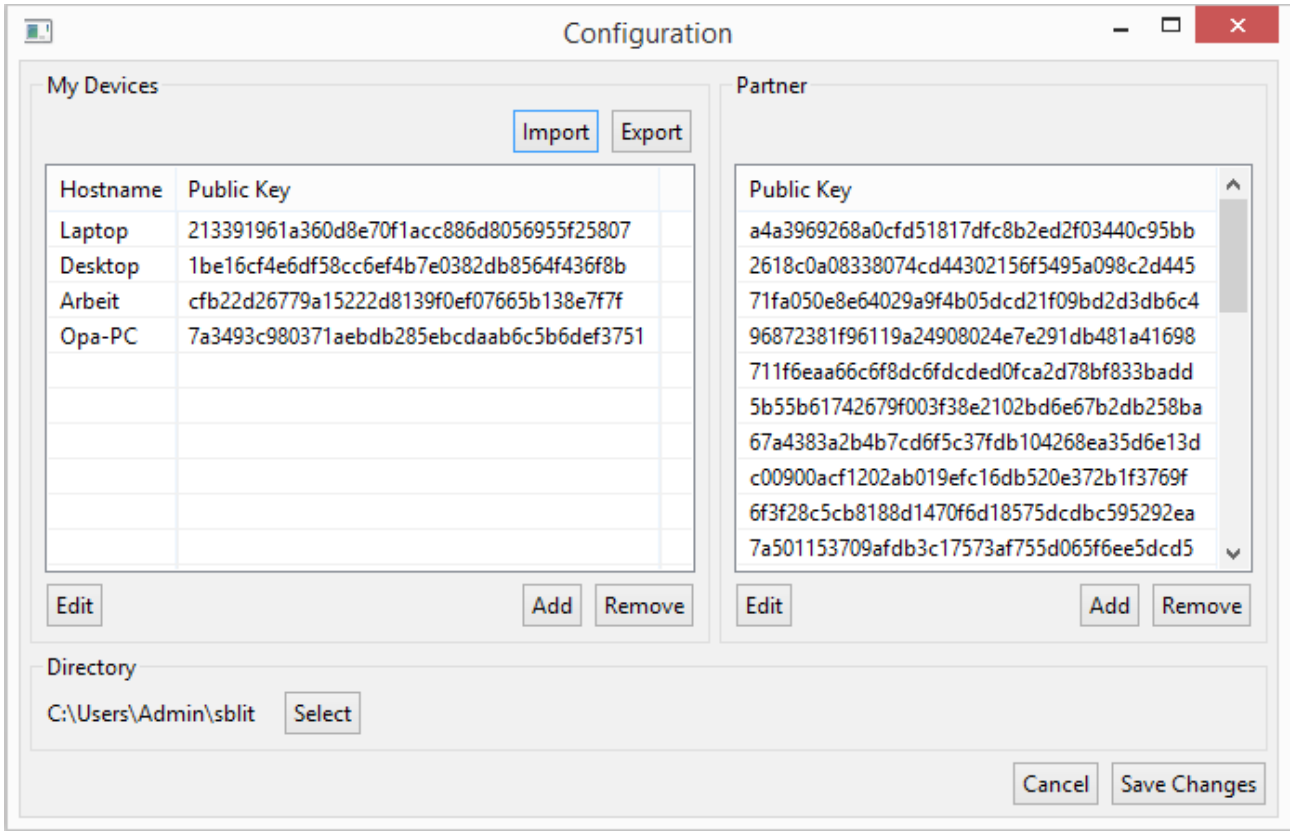
GRAPHISCHE OBERFLÄCHE

Statusübersicht



Das Hauptfenster des zurückhaltenden GUIs lässt sich mit einem Klick auf das Icon im System-Tray aufrufen. Hier werden die letzten Änderungen innerhalb des sblit-Ordners, aufgetretene Konflikte und der Fortschritt der laufenden Synchonisation angezeigt.

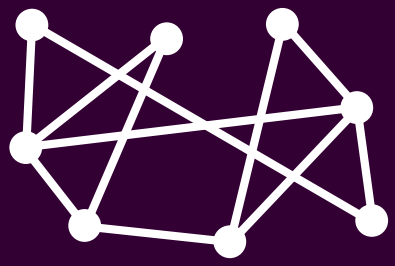
Konfiguration



Mit einem Klick auf das Zahnrad im Hauptfenster öffnet sich das Konfigurationsmenü, in dem der User die Liste der Synchronisationsgeräte und die sonst automatisch verwaltete Liste der Partnergeräte bearbeiten kann. Außerdem lässt sich hier der Pfad des zu synchronisierenden Ordners verändern.

TECHNIK

Peer-to-Peer-Netzwerk



OFFENES PEER-TO-PEER-NETZWERK

Sblit kommuniziert über ein eigens entwickeltes, offenes Peer-to-Peer-Netzwerk, den sogenannten Decentralized Communication Layer (DCL). Dieser stellt neben sblit auch Anwendungen von Fremdentwicklern flexible und vor allem sichere Kommunikationskanäle zur Verfügung.

NETZWERK AUS NETZWERKEN

DCL beherbergt dabei nicht nur ein Peer-to-Peer-Netzwerk, sondern ermöglicht es, praktisch unendlich viele Netzwerke, genauestens abgestimmt auf die Anforderungen der Anwendungen, aufzuziehen.

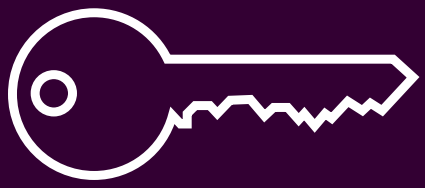
VERBINDUNGSVERSCHLÜSSELUNG

Die Kommunikation zwischen entfernten DCL-Instanzen erfolgt über UDP auf zufälligen Ports. Dabei wird die gesamte Payload der UDP-Pakete mit AES im Modus GCM verschlüsselt, was es unmöglich macht, auf den Inhalt des Datenstroms zu schließen.

APPLICATION LIBRARY

Eine Application Library zur Einbindung von DCL in Drittanwendungen bietet deren Entwicklern eine einfache Möglichkeit, Peer-to-Peer-Kommunikation umzusetzen.
Statt erst ein eigenes Netzwerk aufbauen zu müssen, wird auf die bestehdende Menge an Nodes im DCL zurückgegriffen.

Adressierung mit RSA



DEZENTRALE ADRESSIERUNG

Durch die vollständige Dezentralisierung und die damit einhergehende Gleichberechtigung aller Nodes im DCL können Adressen weder von zentralen Stellen vergeben noch validiert werden. Deshalb kommt im DCL zur Adressierung das asymmetrische Verschlüsselungsverfahren RSA zum Einsatz. Jeder Node generiert beim Start ein RSA-Schlüsselpaar, dessen öffentlicher Schlüssel als Adresse verwendet wird.

VALIDIERUNG VON ADRESSEN

Solche Adressen können gänzlich ohne zentrales Organ auf ihre Echtheit überprüft werden, indem der zu überprüfende Node mit seinem privaten Schlüssel eine Signatur von Daten anfertigt, die vom überprüfenden Node generiert wurden. Der überprüfende Node kann diese Signatur anschließend mit der Adresse, also dem öffentlichen Schlüssel des zu überprüfenden Nodes validieren und somit sichergehen, dass seine Adresse echt ist.

ROUTING

Zum Finden von Wegen durch das DCL-Netzwerk kommt ein Algorithmus zum Einsatz, der dem des Kademlia-Modells sehr ähnlich ist. Eine Nachricht wird dabei schrittweise von einem Node zu dem Node weitergeleitet, dessen Adresse die geringste Differenz zur Zieladresse aufweist, bis die Nachricht schließlich ihr Ziel erreicht. Um die Länge von Adressen zu verringern und dadurch das Routing und die Kommunikation effizienter zu machen, werden die eigentlichen Adressen durch Hashen der öffentlichen Schlüssel erzeugt. Als Hashalgorithmus wird dabei SHA-1 verwendet, wodurch sich Adressen mit einer Länge von 20 Bytes ergeben.

Synchronisation

VERFÜGBARKEIT

Damit die Änderungen auch synchronisiert werden können, wenn immer nur eines der eigenen Geräte gleichzeitig online ist, werden die Änderungen auf Geräten anderer Nutzer zwischengespeichert. Diese Geräte werden Partnergeräte genannt.

DELTA-DATEN

Sblit synchronisiert Delta-Daten. Unter Delta-Daten kann man sich die Änderungen an einer Datei vorstellen. Bei einer Änderung werden diese auf den Partnergeräten gespeichert und wieder gelöscht, sobald die Änderung auf alle eigenen Geräten übertragen wurde.

VERSCHLÜSSELUNG

Sblit verschlüsselt die Dateien mithilfe eines zufällig generierten AES-256-Schlüssels. Beim Hinzufügen eines neuen Gerätes kann dieser exportiert und z.B. über einen USB-Stick auf das neue Gerät kopiert werden. Angenommen, man verwendet den stärksten Supercomputer der Welt, so benötigt man durchschnittlich 10^{56} Jahre, um den Schlüssel durch ausprobieren zu knacken. Das ist in etwa das 10^{46}-fache Alter des Universums.

ERKENNEN VON ÄNDERUNGEN

Damit Änderungen sofort erkannt und synchronisiert werden können, verwendet sblit das Watchservice. Dieses ist eine Schnittstelle zum Dateisystem, welche benachrichtigt wird, sobald sich eine Datei in einem bestimmten Ordner ändert.

Synchronisationskonflikte

KONFLIKT

Konflikte treten auf, wenn ein und dieselbe Datei gleichzeitig auf zwei verschiedenen Geräten geändert wird. Dabei entstehen zwei unterschiedliche Versionen dieser Datei.

ERKENNUNG

Dieses Problem erkennt sblit mit einer internen, Hash-basierten Versionierung. Dabei wird zu jeder Datei ein Versionsverlauf gespeichert. Bei einer Änderung wird immer der zur Datei passende Versionsverlauf mitgeschickt. Stimmt der letzte Hash im lokalen Versionsverlauf mit einem beliebigen Hash im mitgeschickten Versionsverlauf überein, tritt kein Konflikt auf. Enthält der empfangene Versionsverlauf jedoch nicht den aktuellsten Hash im lokalen Versionsverlauf, wurde die Datei geändert, bevor diese synchronisiert werden konnte. Anders gesagt: Ein Konflikt ist aufgetreten.

LÖSUNG

Um einen Konflikt zu lösen, wird dem Namen der Datei auf dem aktuellen Gerät das Suffix „**(Konflikt 1)**“ hinzugefügt und die empfangene Datei unter dem Originalnamen gespeichert. Falls bereits eine gleichnamige Datei mit dem Suffix „**(Konflikt 1)**“ existiert, wird die Nummer entsprechend erhöht. Diese Konfliktdatei wird dann ebenfalls auf das andere Gerät synchronisiert.

Plattformunabhängigkeit

JAVA

Sowohl sblit als auch die Unterkomponenten DCL und die Application Library sind in Java geschrieben, um den reibungslosen Einsatz auf allen Plattformen gewährleisten zu können.

BOUNCY CASTLE

Um die notwendigen kryptographischen Funktionen auf den unterschiedlichen Plattformen uneingeschränkt verwenden zu können, benutzt sblit die freie Bibliothek Bouncy Castle, die auch bei Googles mobilem Betriebssystem Android zum Einsatz kommt.

SWT

Damit der Plattform-spezifiische Look auf den verschiedenen Betriebssystemen nicht verloren geht, verwendet sblit SWT, eine Programmbibliothek für graphische Oberflächen. SWT benutzt die OS-nativen Widgets. Dadurch haben die GUIs auf allen Plattformen das für das jeweilige Betriebssystem typische Aussehen, obwohl überall der gleiche Programmcode ausgeführt wird.

TEAM

Andreas Novak

Graphische Oberfläche, Website

Nikola Szucsich

Synchronisationsanwendung

Martin Exner

Kommunikationsschicht, Projektleitung

FORTSCHRITT

Fortschritt
per 22.03.2015

UMGESETZT

Online-Synchronisation

Benutzeroberfläche

Dateiverschlüsselung

Verbindungsverschlüsselung

Adressierung

Routing

Application Library

AUSSTEHEND

Partnerschaften (Offline-Synchronisation)

Effizienter Netzwerkeinstieg

Verbinden der einzelnen Komponenten