19. August 2016 Studie · Digitale Identität · Web & Cloud · 22 Min. Lesezeit

Flexible Zweifaktor-Authentifizierung mit FIDO

Archiviert. Dieser Inhalt wird nicht mehr aktualisiert. Er bleibt zur Information verfügbar.

Quelle https://technology2.a-sit.at/veroeffentlichungen/flexible-zweifaktor-authentifizierung-mit-fido/

FIDO Universal Second Factor (U2F) ist ein Industriestandard für eine generell verwendbare Zweifaktor-Authentifizierung. Unter Verwendung eines USB-Security-Tokens können sich BenutzerInnen gegenüber einer Vielzahl von Webdiensten authentifizieren. Ein wesentliches Merkmal des U2F-Konzepts besteht darin, dass das entsprechende Hardware-Element zum Zeitpunkt eines Anmeldeverfahrens physisch mit dem Computer verbunden ist, sodass der Web-Browser über eine geeignete Schnittstelle damit unmittelbar interagieren kann.

Der weiten Anwendbarkeit von FIDO U2F steht entgegen, dass für die Verwendung bislang zwingend ein zertifiziertes Hardware-Element benötigt wird. Dies tritt beispielsweise bei einer U2F-Anmeldung über Smartphones in den Vordergrund, mit welchen sich typischerweise keine USB-Tokens verbinden lassen. Oft ist mangels Unterstützung auch ein Zugriff über NFC keine probate Alternative.

Im Zuge dieses Projekts wurde eine Lösung gesucht, um den U2F-Anmeldeprozess auch dann zu ermöglichen, wenn software- oder hardwareseitig keine Verfügbarkeit gegeben ist. Aufbauend auf der bestehenden Architektur der zentralen, als Open-Source-Software verfügbaren, Schlüsselspeicherlösung CrySIL wurde ein Konzept entwickelt um FIDO für beliebige Plattformen anbieten zu können. Die Tauglichkeit der propagierten Lösung konnte im Rahmen einer Implementierung untermauert werden. Konkret wurde für den Webbrowser Firefox eine Erweiterung umgesetzt, welche den Browser um die nativ nicht gegebene FIDO-Unterstützung ergänzt und die Kommunikation mit einem emulierten U2F-Token ermöglicht, welcher auf der Seite von CrySIL realisiert wurde.

Das vorgeschlagene Konzept sowie die praktische Demonstration zeigen neue Anwendungsmöglichkeiten des FIDO U2F-Standards auf. Indem die bisherige Notwendigkeit eines Hardware-Elements zur Authentifizierung entfällt und die Abwicklung des Anmeldeprozesses ein zentraler Schlüsselspeicher durchführt, ermöglicht sich ein hoher Grad an Flexibilität bei Verwendung von FIDO U2F über verschiedene Plattformen hinweg.

Link

[CrySIL Source-Code

](https://github.com/IAIK/CrySIL)

1. Einleitung

Einfache Formen der Authentifizierung, etwa über Benutzername und Passwort, sind vielen Herausforderungen unterworfen und wurden daher in den letzten Jahren vermehrt von mehrfaktorbasierten Mechanismen abgelöst. Die österreichische Bürgerkarte bzw. Handy-Signatur ist nur eines von vielen möglichen Beispielen, anhand derer bei der Remote-Authentifizierung von BenutzerInnen ein hohes Sicherheitsniveau erreicht werden kann. Insbesondere bei sicherheitskritischen Anwendungsszenarien wie der Autorisierung einer E-Banking-Transaktion oder serverbasierten Signaturlösungen trägt der Einsatz weiterer Authentifizierungsfaktoren dazu bei, sicherzustellen, dass nur sensible Aktionen nur von legitimierten Akteuren ausgelöst werden.

Die Sicherheit von Lösungen zur Zweifaktorauthentifizierung basieren traditionellerweise darauf, dass BenutzerInnen sich zunächst mithilfe eines Benutzernamens und eines Passworts ausweisen und anschließend ein weiterer Faktor der Authentifizierung zur Anwendung kommt. Ein Beispiel aus dem Alltag wäre etwa die Verwendung einer physisch vorzuweisenden Bankomatkarte („proof of possession“) sowie die notwendige Eingabe eines PIN-Codes („proof of knowledge“) um Geld vom Bankomaten abzuheben zu können. Analog dazu wird in der klassischen Variante des SMS-TAN-Verfahrens eine Kurznachricht bzw. ein Einmalpasswort (OTP) an ein zweites Endnutzergerät gesendet, deren Inhalt schließlich im anderen Gerät (zumeist einem Web-Browser) eingegeben werden muss. Die Sicherheit dieser Lösungen fundiert auf dem Einsatz zweier voneinander unabhängige Faktoren, die jedoch in Kombination verwendet werden müssen.

Aufgrund des rasant angestiegenen Funktionsumfangs von mobilen Geräten wurde jenen zunehmend ermöglicht, im Web-Browser etwa die gleiche Funktionalität anzubieten, wie bei traditionellen Desktop-Geräten. In weiterer Folge ist es somit theoretisch möglich, dass das SMS-TAN-Verfahren, das ursprünglich für zwei Endgeräte konzipiert wurde, auf nur einem ausgeführt wird. Die ausschließliche Verwendung eines Mobilgeräts für den Zugriff auf sicherheitskritische Anwendungen ist unter Einsatz dieses Verfahrens somit nicht anwendbar. Wenngleich der zweite Faktor das Sicherheitsniveau in jedem Fall erhöht, zeigt dieser Umstand implizit auch auf, welchen Limitierungen Mechanismen zur Zweifaktorauthentifizierung unterworfen sind. Ein Einmalpasswort, das an ein Mobilgerät gesendet wird, benötigt beispielsweise zwangsläufig die Verfügbarkeit eines Mobilfunknetzes. Methoden, die auf einem Challenge-Response-Verfahren basieren, etwa SmartCard-Lösungen wie die österreichische Bürgerkarte, benötigen wiederum ein geeignetes Lesegerät sowie die Möglichkeit, es aus der lokalen Umgebung heraus anzusprechen.

FIDO [1] U2F [2] ist ein offener Standard zur Authentifizierung, der den Einschränkungen traditioneller Mechanismen zur Zweifaktorauthentifizierung entgegen wirken möchte. Die dahinter stehende FIDO Alliance ist ein Industriekonsortium, welches sich der Entwicklung der technischen Aspekte sowie entsprechender Unterstützung bei den jeweils angebotenen Diensten verschrieben hat. Im Wesentlichen ermöglicht es U2F beliebigen Diensten, starke Zweifaktorauthentifizierung zu bestehenden Anmeldeverfahren hinzuzufügen. Im Vordergrund des offenen Protokolls steht ein einfacher Ablauf der Registrierung sowie des Anmeldeprocedere von BenutzerInnen. Hierfür vorgesehen ist die Verwendung eines kryptographischen Hardware-Elements („U2F-Token“), welches für eine beliebige Anzahl von Diensten verwendet werden kann, da für jeden von ihnen ein individueller Token generiert wird. Je nach

Element ist ein Zugriff neben USB auch via NFC oder Bluetooth möglich. Der industriell geprägte Hintergrund von FIDO ermöglicht eine Verwendung bei populären Diensten wie Google, Dropbox und Github. Native Unterstützung, um mit einem FIDO-Token kommunizieren zu können, findet sich derzeit nur im Browser Google Chrome.

1 https://fidoalliance.org 2 https://developers.yubico.com/U2F/

Device Browser Server Username and Password Verify Password Generate Challenge Challenge Challenge User touches button Response Response	**Col2**	**Col3**	**Col4**
Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Verify P	assword
Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Verify P	
Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Generate	Challenge
Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Browser Device Server Challenge Username and Password Challenge Response Response User touches button Verify Password Generate Challenge	Generate	
User touc	hes button	hes button	hes button

Abbildung 1.Schematische Darstellung einer Anmeldung über U2F.

1.1. Motivation

Bestehende U2F-Token implementieren ein Secure Element auf der Größe eines USB-Datensticks. Die Registrierung sowie der Anmeldeprozess von BenutzerInnen basieren auf einem „Challenge-Response"-Verfahren zwischen einem Dienst („relying party") und dem Hardware-Element. Typischerweise dient eine Software, welche lokal auf dem Endgerät läuft, als Mediator zwischen Token und Service. Dem Web-Browser kommt dabei die Aufgabe zu, eine sichere Umgebung für den Anmeldeprozess zu schaffen und durch die Browser-Sandbox etwa Man-in-the-Middle-Angriffe zu verhindern. AnwenderInnen obliegt es schließlich nur noch, das Hardware-Element zu verbinden und nach Aufforderung den Knopf am Element als „proof of possession" zu betätigen bzw. um eine Operation zu bestätigen (siehe Abbildung 1). Dieser

Vorgang ist vergleichsweise einfach auszuführen, sofern eine Desktop-Umgebung mit Google Chrome verwendet wird und BenutzerInnen das Hardware-Element lokal verfügbar halten. Ein analoger Einsatz mit weiteren Web-Browsern ist zum gegenwertigen Zeitpunkt mangels Unterstützung nicht durchführbar.

Das beschriebene Anwendungsszenario ist abseits der Modellbedingungen jedoch nicht ohne weiteres anwendbar. Einer der Hauptgründe dafür ist bei Mobilgeräten etwa die mangelnde Verfügbarkeit einer USB-Schnittstelle, welche den Anschluss eines U2F-Token ermöglichen würde. Wenngleich manche Token alternativ über NFC oder Bluetooth angesprochen werden können, gibt es bis dato keine mobilen Browser, die diese Methoden unterstützen würden. Sollte Zweifaktorauthentifizierung über U2F bei einem Dienst für bereits registrierte AnwenderInnen dennoch aktiviert sein, muss notwendigerweise auf einen Authentifizierungsmechanismus umgeschaltet werden, um eine Anmeldung dennoch zu ermöglichen. Dadurch erhöht sich letztlich auch die Komplexität eines Einsatzes von U2F über mehrere Plattformen hinweg. Trotz der grundsätzlichen Offenlegung des U2F-Protokolls, können zurzeit nur zertifizierte Token (z.B. von Yubico [3]) eingesetzt werden. Bestehende „Challenge-Response"-Verfahren von SmartCards sind mangels Protokollverständnis nicht kompatibel mit U2F, auch wenn die darunter liegenden kryptographischen Operationen ausgeführt werden könnten.

Abgesehen von den technischen Einschränkungen belastet das stets notwendige Mitführen eines U2F-Tokens die Flexibilität und den Komfort eines Einsatzes auf Seite der AnwenderInnen doch in erheblichem Ausmaße.

3 https://developers.yubico.com/U2F/

1.2. Ziele dieses Projekts

Dieses Projekt begegnet den im vorigen Abschnitt beschriebenen Einschränkungen und versucht eine Annäherung an eine Umgebung zur Zweifaktorauthentifizierung, bei der BenutzerInnen von der Einfachheit des U2F-Standards profitieren können, ohne an ein Hardware-Element oder einen spezifischen Web-Browser gebunden zu sein. Dazu führt die nachstehende Auflistung mehrere Anforderungen auf und stellt zugleich einen Leitfaden zur Umsetzung dar.

- Eine Abstrahierung der technischen Verbindung weg von einer konkreten Umsetzung (NFC, USB, Bluetooth) um mit einem Token zu interagieren, kann den Einsatz über mehrere Plattformen hinweg gewährleisten. Für BenutzerInnen würde es somit einerseits entfallen, ein Hardware-Element für den Anmeldeprozess physisch mitzuführen und andererseits wäre es möglich, die gleichen benutzerspezifischen Kennungen an mehreren Geräten einzusetzen.

Aufgrund der thematischen Nähe wird hierfür eine zentralisierte Lösung vorgeschlagen, welche den Token für AnwenderInnen verwaltet und von verschiedenen Plattformen aus gleichermaßen erreichbar wäre. Die einzige Abhängigkeit wäre eine funktionsfähige Internetverbindung. Da die Open-Source-Software CrySIL [4] bereits zentralisierte Lösungsansätze zum Management von kryptographischen Schlüssel und Operationen implementiert, wird eine Integration von FIDO U2F in diese Plattform ins Auge gefasst. Prinzipiell konzipiert für den Einsatz mit heterogenen Plattformen, kann die modular aufgebaute CrySIL-Lösung um Unterstützung für das U2F-Protokoll erweitert werden.

Die zentralisierte Plattform könnte fortan die Rolle des U2F-Token übernehmen. In dieser Funktionsausübung gelingt es zudem, die bisher notwendige physische Verfügbarkeit eines Token soweit zu abstrahieren, dass es für BenutzerInnen unerheblich ist, wo die beim U2F-Prozess generierten Schlüssel und Zertifikate abgelegt sind.

- Google Chrome ist zurzeit der einzige Desktop-Browser, welcher Unterstützung für FIDO U2F implementiert. Eine Integration in Mozilla Firefox wird diskutiert [5], wobei sich jedoch im aktuellen Verlauf der Diskussion kein Konsens abzeichnet, der auf eine baldige Integration der FIDO U2F-API [6] in diesem Web-Browser hinweisen würde.

Um eine breitere Einsetzbarkeit des Standards zu ermöglichen, wird im Rahmen dieses Projekts eine eigenständige Integration in Mozilla Firefox angestrebt. Für dieses Vorhaben lassen sich auch Resultate aus einem früheren Projekt verwerten, bei dem erhoben wurde, wie CrySIL transparent in beliebige Webseiten integriert werden könnte. Im konkreten Fall ließe sich die U2F-API etwa als Browser-Erweiterung bereitstellen, die die andernfalls nicht existierende API nachrüsten und FIDO U2F über CrySIL verwendbar machen würde. Da Mozilla Firefox mittlerweile auch für Android- und iOS-Mobilgeräte verfügbar ist und mit Erweiterungen ausgestattet werden kann, könnte diese Lösung neben Desktop-Umgebungen auch Mobilplattformen versorgen.

Das angestrebte Konzept zur Flexibilisierung des FIDO-Mechanismus anhand der Plattform CrySIL wird im folgenden Abschnitt genauer erörtert. Um die Praxistauglichkeit zu untermauern, wird in einer prototypischen Implementierung die Realisierung folgender Komponenten vorgenommen:

- **Emulation eines U2F-Token als Modul von CrySIL** Durch die Ergänzung von CrySIL um die Unterstützung des FIDO U2F-Protokolls wird die Verwendbarkeit von FIDO ohne zertifiziertes Hardware-Element bestätigt. Für Demonstrationszwecke wird auf einen Schlüsselspeicher zurückgegriffen, der in Software abgelegt ist. Für einen produktiven Einsatz könnten die entsprechenden Schlüssel und Zertifikate seitens CrySIL auf einem beliebigen Hardware Security Module (HSM) abgelegt

4 https://github.com/IAIK/CrySIL 5 https://bugzilla.mozilla.org/show_bug.cgi?id=1065729 6 https://fidoalliance.org/specs/fido-u2f-javascript-api-ps-20150514.pdf sein. In der Praxis bedeutet das, dass das CrySIL-Modul die entsprechenden kryptographischen Operationen etwa auch mithilfe der österreichischen Bürgerkarte durchführen könnte, ohne dass diese dazu spezielle Voraussetzungen für einen Einsatz mit U2F erfüllen müsste.

- **Entwicklung eine Firefox-Erweiterung zur Nachrüstung der U2F-API über CrySIL** Die Bereitstellung der FIDO U2F-API über eine Erweiterung, die mit der Desktop- sowie Mobilversion von Mozilla Firefox kompatibel ist, soll dazu dienen, die Praxistauglichkeit der zentralisierten Lösung zu veranschaulichen. Das Ziel des Demonstrators ist es, das FIDO U2F-Protokoll mit Firefox auf beliebigen Seiten ohne ein explizites Hardware-Element (bzw. über ein von CrySIL emuliertes) einsetzen zu können.

Eine analoge Implementierung wäre auch für Google Chrome realisierbar. Da die U2F-API bei Chrome jedoch bereits nativ mitgeliefert wird, wäre es nur möglich, sie durch eine eigene Implementierung zu überschreiben, die daraufhin zwar mit CrySIL über das U2F-Protokoll kommunizieren könnte, jedoch nicht

mehr mit physisch angeschlossenen Hardware-Elementen. Im Interesse einer breiteren Einsetzbarkeit von U2F wurde darauf verzichtet und stattdessen eine Lösung mit Mozilla Firefox avisiert.

2. Konzept eines zentralisierten FIDO-Token

Um den bestehenden Einschränkungen von FIDO U2F zu begegnen und den Authentifizierungsvorgang über mehrere Plattformen hinweg zu ermöglichen, wird eine Integration des U2F-Protokolls in CrySIL angestrebt. Bei CrySIL handelt es sich um eine unter der EUPL-Lizenz verfügbaren, zentralisierten Plattform zur Verwaltung kryptographischer Schlüssel sowie zur Anwendung derselben. Die angestrebte Flexibilität wird vor allem dadurch gewährleistet, da CrySIL plattformunabhängig eingesetzt und anhand von Modulen auch dahingehend erweitert werden kann, U2F-Nachrichten zu verarbeiten und entsprechende kryptographische Operationen auszuführen.

2.1. Vorgeschlagenes Konzept

Wenngleich der FIDO-Standard auf eine offene Plattform zur Zweifaktorauthentifizierung abzielt, die Webseitenbetreibern eine vergleichsweise einfache Integration ermöglicht, ist die Anwendbarkeit auf Seite der AnwenderInnen davon gezeichnet, stets ein entsprechendes Hardware-Element physisch mit einem weiteren zu verbinden, um eine Authentifizierung durchzuführen. Wie anhand zuvor angeführter Beispiele beschrieben, ist ein derartiger Einsatz in der Praxis oft nicht möglich.

Zur Flexibilisierung der Authentifizierung mit FIDO wird daher eine Integration als CrySIL-Modul vorgeschlagen. In der bestehenden Infrastruktur ist die Möglichkeit bereits geschaffen, Daten zu verschlüsseln, zu signieren, kryptographische Schlüssel zu generieren und mittels „key wrapping“ Schlüssel zu exportieren. Diese Funktionalität ähnelt jener eines U2F-Token als Hardware-Element, welcher asymmetrische Schlüsselpaare generiert und „Challenges“ damit signiert. Angesichts der funktionalen Anforderungen an einen U2F-Token erscheint die Integration als CrySIL-Modul daher zu favorisieren.

2.2. Umsetzung der U2F-Operationen

Ein für das FIDO U2F-Protokoll zertifiziertes Hardware-Element ist zur Erfüllung zweier wesentlicher Operationen ausgelegt: die Bindung (Registrierung) eines U2F-Token an ein Benutzerkonto sowie die Bestätigung des Authentifizierungsprozesses durch Signatur einer gegebenen „Challenge“. Für die vorgeschlagene Umsetzung in CrySIL gilt es, die kryptographischen Vorgänge gemäß der funktionalen Spezifikation [7] zu integrieren, die Werte als U2F-Nachrichten im vorgesehen Format [8] zurückzugeben und die Interaktion über eine entsprechende API zu ermöglichen.

Beim Registrieren eines neuen Benutzerzugangs auf einer Webseite wird bei U2F klassischerweise ein Befehl an das Hardware-Element weitergegeben, damit dort infolge ein neues asymmetrisches 7 https://fidoalliance.org/specs/fido-u2f-overview-ps-20150514.pdf 8 https://fidoalliance.org/specs/fido-u2f-raw-message-formats-ps-20150514.pdf Schlüsselpaar erstellt, ein gerätespezifisches Zertifikat („Attestation Certificate“) sowie die berechnete Signatur über eine gegebene „Challenge“ zurückgegeben werden. Für das Generieren eines „Wrapped Key“ und das Berechnen einer Signatur über Daten können bei Verwendung von CrySIL die bereits bestehenden Funktionen verwendet werden. Möchten BenutzerInnen sich daraufhin bei Diensten über U2F anmelden, sendet der Browser als „Relying Party“ eine Anmeldeanfrage an den Token. Das Hardware-Element hat daraufhin eine Signatur über eine gegebene „Challenge“ zu berechnen und teilt den Wert eines monoton steigenden Zählwertes („Counter Value“) mit.

Der Browser inkludiert in seiner Anfrage einen sog. „Key Handle“ welcher jenes Schlüsselpaar identifiziert, das im Zuge der vorangegangenen Registrierung generiert wurde. Bei Substitution des Hardware-Elements durch CrySIL wird zunächst der als „Key Handle“ aus dem verwendeten Schlüsselspeicher abgerufen. Die ebenfalls gegebene „Challenge“ wird daraufhin mit dem privaten Schlüssel signiert. Gleich wie beim zertifizierten Hardware-Element umfasst die Signatur außerdem den Zähler, der intern vorgehalten und mit jeder Anmeldung erhöht wird.

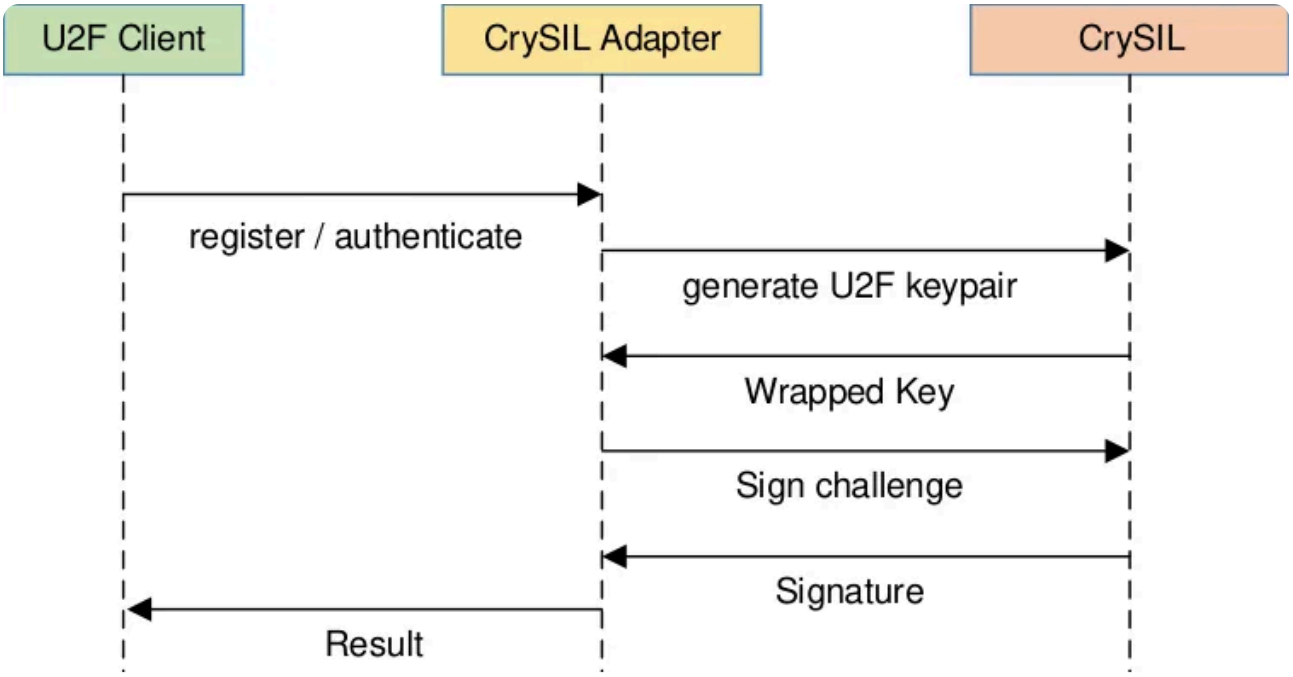
Die konkrete Umsetzung des Konzepts erschießt sich aus dem Sequenzdiagramm in Abbildung 3. Anstelle des Hardware-Elements leitet der U2F-Client die Nachrichten im Format des U2F-Protokolls an eine CrySIL-Schnittstelle weiter (hier als „CrySIL Adapter“ bezeichnet), in welcher die Nachrichten verarbeitet und die entsprechenden Anfragen an CrySIL gestellt werden. Sollte beispielsweise eine Registrierung auf einer Seite vorgenommen werden, teilt der U2F-Client die zugehörigen Parameter anstelle eines Hardware-Elements an den CrySIL-Adapter mit, welcher sie wiederum in CrySIL-Befehlen umwandelt und die Ausführung veranlasst. Der Adapter implementiert somit die Logik des U2F-Protokolls und abstrahiert es von den kryptographischen Operationen bzw. der Schlüsselverwaltung, die hier von CrySIL übernommen wird.

Abbildung 2. Ablaufdiagramm eines Austauschs von Nachrichten zwischen U2F-Client, Protokolladapter für CrySIL und der CrySIL-Instanz zur Ausführung kryptographischer Operationen.

2.3. Gegenüberstellung

Zum besseren Verständnis wird im Folgenden der traditionelle Ansatz von FIDO U2F mit einem auf CrySIL angewandten gegenüber gestellt. Abbildung 3 veranschaulicht die Interaktion über das U2F-Protokoll im traditionellen Anwendungsszenario (linke Darstellung) sowie eine über CrySIL realisierte (rechte Darstellung). In beiden Fällen behalten BenutzerInnen die Kontrolle über den Client und Authenticator, wenn die Vorannahme getroffen wird, dass CrySIL in der lokalen Domäne bereitgestellt wird. Die Wolken in der Darstellung symbolisieren entfernte Vebindungen (z.B. via Internet) zwischen einer „relying party“ und einem Client, der U2F-Anfragen verarbeiten kann.

Aus der Darstellung ist ersichtlich, dass beim ursprünglichen Ansatz eine „relying party“, wie etwa der Web-Browser, eine U2F-Anfrage an ein Hardware-Element schickt, welches die Anfrage an entsprechend verarbeitet und einen als „Authenticator“ bezeichneten Dienst mit den



kryptographischen Operationen befasst. Die Antworten dieses Dienstes werden schließlich wieder im entsprechenden U2F-Format an den Anfragesteller zurückgegeben.

U2F Relying Party U2F Relying Party

Col1	Col2
U2F Client (Protocol Endpoint) Authenticator User Control	U2F Client (Protocol Endpoint) Authenticator User Control

Col1	Col2
U2F Client (Protocol Endpoint) Authenticator User Control CrySIL Instance	U2F Client (Protocol Endpoint) Authenticator User Control CrySIL Instance

Abbildung 3. Links: Allgemeiner U2F-Ansatz. Rechts: U2F über CrySIL. Im Vergleich dazu können bei der Realisierung über CrySIL die U2F-Nachrichten über einen beliebigen Endpunkt entgegen genommen und zur Operationsausführung weitergeleitet werden. Ist beim allgemeinen U2F-Ansatz etwa typischerweise das Hardware-Element dafür zuständig, die U2F-Nachrichten zu verarbeiten, kann das bei einem Einsatz mit CrySIL auch der Web-Browser sein, welcher als Endpunkt die für U2F-Nachrichten notwendige Logik implementiert, an CrySIL aber schließlich nur jene Daten weiterleitet, die zur Ausführung der kryptographischen Operationen relevant sind. Der dadurch erzielbare Effekt ist eine Entkoppelung von der Abhängigkeit einer monolithischen Implementierung über ein Hardware-Element.

Während im generellen Ansatz die Verbindung zwischen „relying party“ und Hardware-Element üblicherweise über eine direkte Verbindung z.B. über Bluetooth, NFC oder USB hergestellt ist, findet die Kommunikation mit CrySIL etwa über HTTPS gesichert statt.

3. Prototypische Implementierung

Zur Veranschaulichung des praktischen Nutzens des im vorherigen Abschnitt vorgestellten Konzepts einer Verwendung von U2F über CrySIL werden im Folgenden die Spezifika der umgesetzten prototypischen Implementierung erläutert.

Wie bereits in Abschnitt 1.2 vorgeschlagen sowie in der ursprünglichen Zielsetzung festgelegt, wird eine erweiterte Verfügbarkeit von FIDO U2F angestrebt, die über die bisherigen Plattformen bzw. Endgeräte hinausgeht. Konkret soll die Umsetzung demonstrieren, dass U2F sowohl auf Desktopwie auch auf Mobilgeräten gleichermaßen eingesetzt werden kann, auch wenn kein physisches Hardware-Element verfügbar ist oder der verwendete Web-Browser keine native Unterstützung bietet. Hierfür wurde eine Browser-Erweiterung für Mozilla Firefox umgesetzt, deren Aufgabe darin besteht, die U2F-Javascript API anzubieten (da sie nativ nicht existiert), U2F-Befehle entgegen zu nehmen und in geeigneter Weise an CrySIL weiterzuleiten bzw. die Antworten wieder zurückzugeben. Auf der Seite von CrySIL wurde ein Modul geschaffen, das die Registrierung und Authentifizierung nach U2F übernimmt bzw. die passenden Attribute zurückliefert.

Im Folgenden wird detaillierter auf Implementierungsaspekte der Firefox-Erweiterung und des CrySIL-Moduls eingegangen.

3.1. Realisierung einer Browser-Erweiterung

FIDO U2F ist ausgerichtet auf die Benutzerauthentifizierung im Web. Obwohl bereits mehrere populäre Seiten wie Dropbox, Google und Github U2F implementieren, kann diese Funktionalität bisher nur von NutzerInnen von Google Chrome verwendet werden. In Mozilla Firefox gibt es bis dato keine Implementierung der JavaScript API.

Infolgedessen wurde im Zuge dieses Projekts eine Erweiterung entwickelt, die dafür sorgt, dass die FIDO U2F Javascript API [9] in das Document Object Model (DOM) einer jeden aufgerufenen Seite zu injizieren. Erkenntnisse aus einem früheren Projekt würden diese Vorgehensweise auch bei Google Chrome zu, wenngleich dies die dort nativ ohnehin bereitgestellte API überschreiben und eine Kommunikation mit physischen Hardware-Elementen unmöglich machen würde. In Mozilla Firefox ist eine vergleichbare Unterstützung bis dato nicht verfügbar, weshalb die Erweiterung nicht nur die Vorteile des propagierten Konzepts mitbringt sondern erstmals überhaupt U2F-Unterstützung in Firefox einbringt.

Die Erweiterung implementiert die JavaScript-API gemäß Spezifikation und kann somit von beliebigen Seiten, die U2F bereits bisher anbieten, verwendet werden. Für Testzwecke im Zuge dieses Projekts wurde auf den Referenzdienst [10] des Herstellers von U2F-Token Yubico zurückgegriffen.

3.2. Browser-Integration

Die Implementierung der JavaScript-API wird über die Erweiterung beim Aufruf beliebiger Webseiten in den DOM der aufgerufenen Seite als „Content Script“ [11] injiziert. Wurde von einer Webseite ein entsprechender Aufruf ausgelöst, werden sämtliche zugehörigen Parameter vom lokalen DOM in jene der Browser-Erweiterung [12] übergeleitet. Einerseits wird auf diese Weise verhindert, dass irgendein Script, das ebenfalls auf der geladenen Webseite ausgeführt wird, mit der Erweiterung unter Umgehung der API kommuniziert oder gar den Kommunikationsfluss in schädlicher Weise beeinflusst (z.B. durch Überschreiben von Funktionen). Andererseits wird es dadurch möglich, die Same Origin Policy (SOP) des

Browsers zu umgehen und mit CrySIL direkt zu kommunizieren ohne auf den Scope der geladenen Seite beschränkt zu sein. Wäre eine derartige Umgehung nicht möglich, würde die Browser-Erweiterung an CrySIL keine Anfragen senden können, sollte CrySIL auf einem entfernten Dienst bereitgestellt werden. Für die Aufbereitung der von CrySIL zu signierenden Daten wird auf externe Libraries [1314] zurückgegriffen.

Wie in Abbildung 4 illustriert, werden von der Browser-Erweiterung empfangene Nachrichten im U2F-Protokollformat schließlich an einen Protokoll-Adapter weitergereicht, welcher die relevanten Attribute extrahiert. Anschließend findet eine Konvertierung der gegebenen Befehle auf das Format von CrySIL hin statt. Der CrySIL-Adapter übernimmt somit die Kontrolle über die dahinterliegende CrySIL-Instanz und instruiert sie, Schlüsselpaare zu generieren, „Challenges“ zu signieren oder das sog. „Attestion Certificate“ zurückzugeben. Die anschließend in das CrySIL-Format übertragenen 9 https://fidoalliance.org/specs/fido-u2f-javascript-api-ps-20150514.pdf 10 https://demo.yubico.com/u2f 11 https://developer.mozilla.org/de/Add-ons/SDK/Guides/Content_Scripts 12 https://developer.mozilla.org/en-US/Add-ons/SDK/Guides/Content_Scripts/using_port 13 https://kjur.github.io/jsrsasign/ 14 https://github.com/brillout/forge-sha256 Anfragen werden schließlich an eine definierte CrySIL-Instanz via HTTPS übertragen. Die erhaltenen Antworten werden schließlich wieder rückgewandelt, in das U2F-Format gebracht und in den DOM der geladenen Webseite zurückgegeben.

Injected API **:** CrySIL Instance

u2f.register( ) u2f.sign( ) Browser Addon Message Listener U2F Client Forwarder (Protocol Endpoint) CrySIL Adapter Ext. Library Adapter	**Col2**
U2F Client (Protocol Endpoint) `u2f.register( )` `u2f.sign( )` Message Listener Ext. Library Adapter CrySIL Adapter Forwarder Browser Addon	Browser Addon
U2F Client (Protocol Endpoint) Message Listener Ext. Library Adapter CrySIL Adapter Forwarder	

Abbildung 4. Darstellung der Komponenten und Kommunikationspfade der Browser-Erweiterung.

3.3. Ablauf der U2F-Operationen

Im Sinne der Veranschaulichung der Funktionsweise der Browser-Erweiterung in Verbindung mit CrySIL werden im Folgenden ein Registrierungs- und ein Anmeldeprozess vorgeführt.

Registrierung:

1. Möchten sich AnwenderInnen bei Webdiensten registrieren, generiert der Server zunächst randomisiert einen aus 32 Bytes bestehenden „Challenge“-Wert.
2. Der Server sendet diesen Wert, eine FIDO-Versionskennung und eine Identifikationsnummer der Login-Applikation („appId“) an den Browser.
3. Unter Verwendung der U2F JavaScript API leitet der Browser diese Parameter sowie die Herkunft („Origin“) der „Challenge“ an die Browser-Erweiterung weiter, welche die Werte weiterverarbeitet, für CrySIL aufbereitet und dort hin weiterleitet.

4. Im klassischen Einsatzszenario würden AnwenderInnen nun den Zugriff über ihr Hardware-Element bestätigen müssen. Im Falle von CrySIL entfällt dieser Schritt – kann aber durch eine beliebige andere Hürde ersetzt werden. Denkbar wäre etwa eine zusätzliche Anmeldung von BenutzerInnen bei CrySIL oder die Erfüllung einer hinreichend sichereren „Challenge".

5. Die CrySIL-Instanz retourniert schließlich den öffentlichen Schlüssel des generierten Schlüsselpaars, ein „Attestation Certificate" die Kennung des „Key Handle" sowie eine Signatur. Letztere umfasst die von der Webapplikation mitgeteilte „appId", einen SHA256-Hashwert der gegebenen „Challenge", den „Origin" der Webseite, den öffentl. Schlüssel sowie den „Key Handle".

6. Die von CrySIL bereitgestellten Daten werden von der Browser-Erweiterung schließlich ausgewertet. Konkret wird das Zertifikat geprüft, die Signatur validiert und der gelieferte öffentliche Schlüssel sowie der „Key Handle" vermerkt.

7. CrySIL ist nun registriert für die Verwendung mit diesem „Origin" und der konkreten Applikation (gemäß „appId").

Die visuelle Wahrnehmung dieses Prozesses unterscheidet sich für AnwenderInnen dabei in keiner Weise von der Verwendung eines Hardware-Elements, wie Abbildung 5 und 6 veranschaulichen. Einzig der Hinweis, dass das Gerät „unverified" sei, indiziert hier, dass der Registrierungsprozess kein „Attestation Certificate" zurückgeliefert hat, das von Yubico ausgestellt wurde. Im Falle der prototypischen Implementierung wurde hier auf ein selbstgeneriertes Zertifikat zurückgegriffen. Unmittelbare Auswirkungen auf den Registrierungs- und Anmeldeprozess gibt es dadurch keine, da FIDO-Token auch grundsätzlich von anderen Herstellern bereitgestellt werden könnten.

Abbildung 5. Beginn einer Registrierung auf https://demo.yubico.com/u2f

Authentifizierung:

1. Der Server des Webdienstes generiert einen aus 32 Byte bestehenden „Challenge"-Wert, der mit jeder möglichen „Key Handle" verwendbar wäre.
2. Der Server sendet diesen Wert sowie die Applikationskennung („appId") an den Browser.
3. Unter Verwendung der U2F JavaScript API nimmt die Browser-Erweiterung die Parameter entgegen, konvertiert sie in ein für CrySIL taugliches Format und leitet sie dorthin weiter.
4. CrySIL berechnet schließlich eine Signatur über den SHA256-Hashwert der „appId", den hinterlegten Zählwert („Counter Value") sowie den SHA256-Hashwert der gegebenen „Challenge" sowie des „Origin". Das Resultat wird an die Browser-Erweiterung retourniert.
5. Die Browser-Erweiterung leitet den Signaturwert weiter an die geladene Webseite.
6. Der Server der U2F-Anwendung verifiziert die Signatur anhand des früher gespeicherten öffentlichen Schlüssels und stellt sicher, dass der gelieferte Zählwert auf jeden Fall größer ist als Werte, die zuvor mit diesem „Key Handle" verwendet wurden. Diese Praxis beabsichtigt, sog. „Replay-Angriffe" abzuhalten.



Abbildung 6. Erfolgreicher Abschluss der U2F-Registrierung mittels CrySIL als FIDO-Token.

Die aus Abbildung 7 ersichtlichen technischen Daten des Authentifizierungsprozesses lassen wiederum keine Unterschiede gegenüber einer klassischen Anmeldung über ein Hardware-Element erkennen.

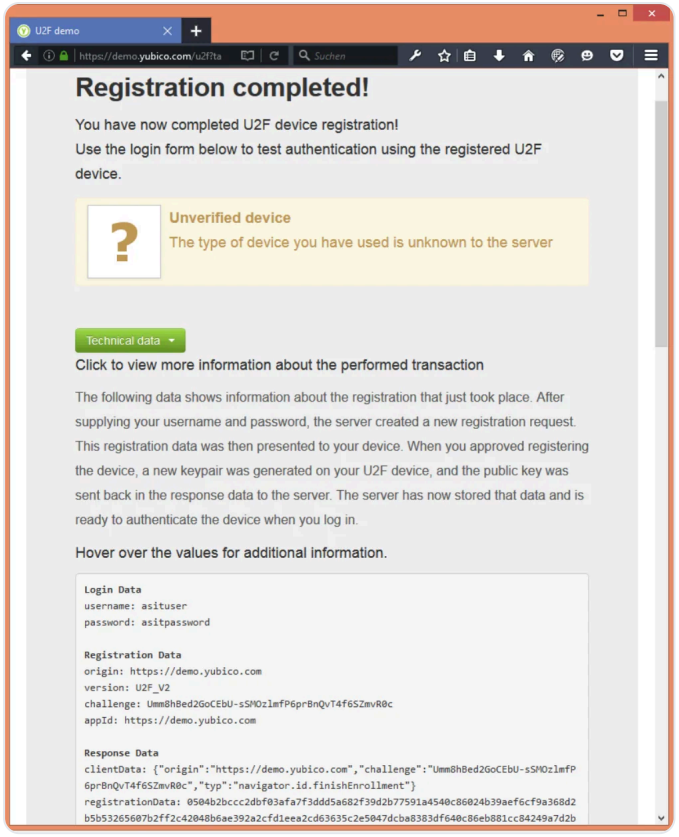
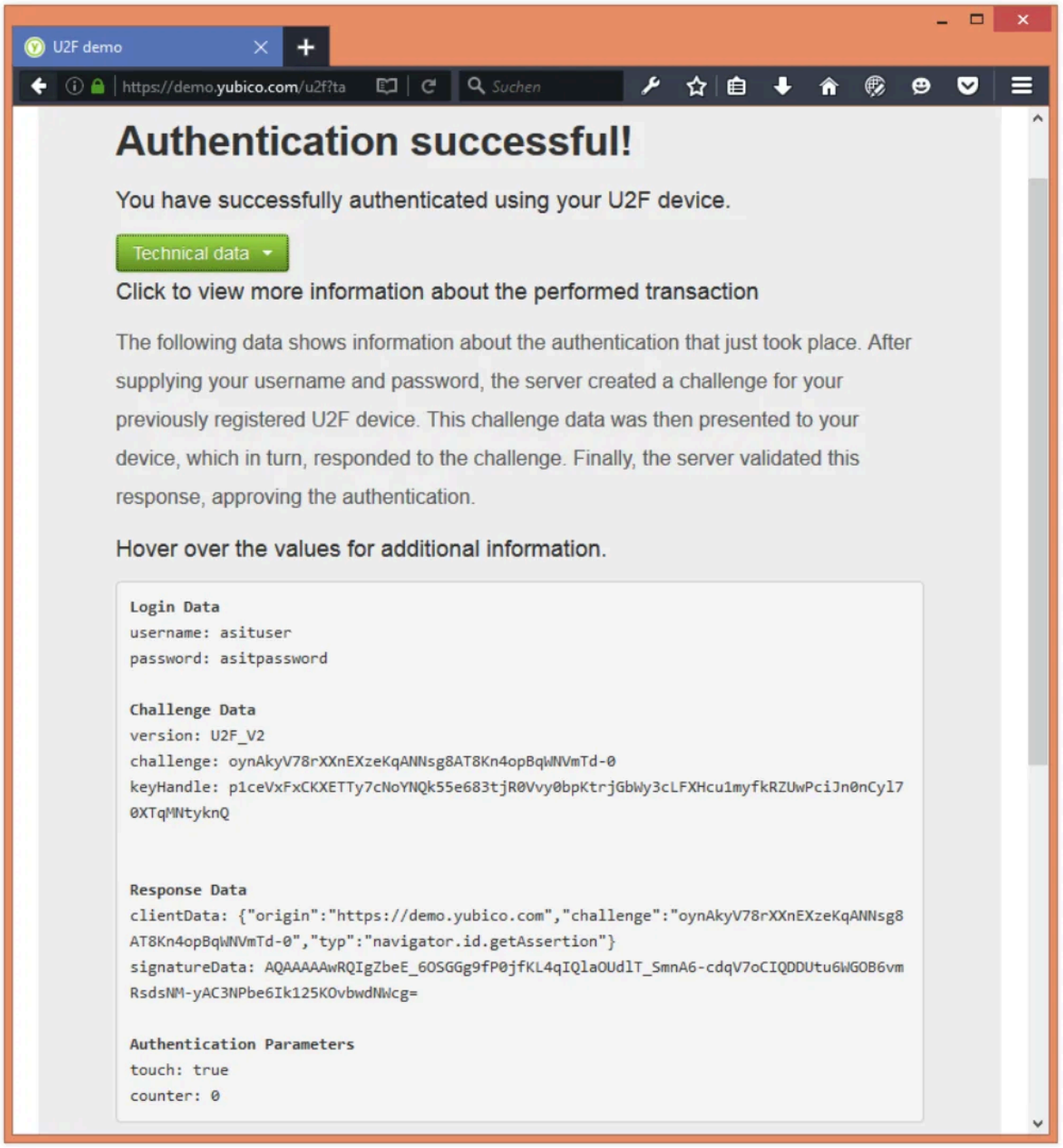


Abbildung 7. Erfolgreiche Anmeldung eines Benutzers mittels eines von CrySIL emulierten Token.



4. Einschränkungen der Lösung

Wenngleich der implementierte Demonstrator die Praxistauglichkeit des Konzepts unterstreicht, gibt es selbstverständlich viele relevante und teils auch sicherheitskritische Aspekte, die für einen produktiven Einsatz der Lösung beachtet werden müssten. Essentiell erscheinen nach Durchführung dieses Projekts folgende Erkenntnisse:

- Bei der hier vorgeschlagenen Lösung findet eine nicht unwesentliche Re-Positionierung einer zentralen Sicherheitseigenschaft von FIDO-Token statt: Die manuelle Bestätigung einer Operation mittels eines Knopfdrucks durch BenutzerInnen entfällt und wird durch CrySIL substituiert. Nichts

desto trotz muss dieser Vorgang nicht zwangsläufig mit einem aktiven Kontrollverlust einhergehen. Sollte die verwendete CrySIL-Instanz in unmittelbarer Einflusssphäre der NutzerInnen sein, etwa durch lokale Bereitstellung, ändert sich durch die

Substitution lediglich der modus operandi. Analog dazu kann natürlich auch die physische Betätigung eins Druckknopfes durch eine qualifizierte Aktion seitens CrySIL ersetzt werden.

Hierfür infrage käme etwa eine zusätzliche Authentifizierung von BenutzerInnen auf Seiten von CrySIL. Würde die U2F-Operation erst nach erfolgreichem Abschluss derselben fortgesetzt, ließe sich so ein effektiver Schutz vor illegitimen Zugriff auf fremde Token realisieren. Da CrySIL prinzipiell modular aufgebaut ist, wäre es sogar denkbar, die vorgeschaltete Authentifizierung an die jeweilige Webseite anzupassen, für die eine U2F-Operation durchgeführt werden soll. Konkret hieße das beispielsweise, dass für den Login bei einer E-Banking-Lösung, die U2F anbietet, der Zugriff auf CrySIL anders geregelt sein könnte als für den U2F-Loginversich bei einer weniger sensiblen Webseite.

- In der prototypischen Implementierung wurde der Einfachheit halber auf ein Benutzermanagement verzichtet. Der als CrySIL-Modul realisierte Authenticator liefert somit allen Anfragenden den gleichen öffentlichen Schlüssel bzw. das gleiche „Attestation Certificate“ zurück. Wenngleich dies für Demonstrationszwecke nicht weiter hinderlich ist, wären für den produktiven Einsatz entsprechende Erweiterungen bei CrySIL nötig.

- Die Browser-Erweiterung übernimmt die Funktion eines U2F-Clients bzw. Protokollendpunkts. Um vorzubeugen, dass potentielle Angreifer z.B. die U2F JavaScript-API überschreiben und Parameter des Registrierungs- bzw. Login abschöpfen, wurde die Clientfunktionalität nicht als „Content Script“ realisiert sondern wird im DOM der Erweiterung ausgeführt. Die Sicherheit dieser Implementierung ist im Wesentlichen abhängig von der eingesetzten Browser-Sandbox. Die Browser-Erweiterung bedient sich somit der Sicherheitsfunktionen von Mozilla Firefox. Würde beabsichtigt, die U2F JavaScript-API in Form von Erweiterungen z.B. für Microsoft Edge oder Apple Safari bereit zu stellen, würden hier selbstverständlich wieder unterschiedliche Sicherheitsaspekte zum Tragen kommen.

5. Fazit

Im Zuge dieses Projekts wurde ein Konzept erarbeitet und praktisch umgesetzt, anhand dessen eine Browser-Erweiterung die FIDO U2F JavaScript-API für Mozilla Firefox bereitstellt. Beliebige Webseiten, die die Registrierung und die Anmeldung über U2F anbieten, können sie verwenden, um mit einem emulierten FIDO-Token zu interagieren.

Angesichts der Einschränkung von FIDO U2F auf die Gegebenheit einer Bluetooth-, NFC- oder USB-Verbindung mit einem Hardware-Element kann FIDO auf anderweitigen Plattformen bislang nicht eingesetzt werden. In diesem Projekt wurde eine praxistaugliche Lösung vorgestellt um beliebige Plattformen mit Web-Browser über U2F kommunizieren zu lassen. Anhand einer Erweiterung für Mozilla Firefox wurde dem Browser erstmalig überhaupt Unterstützung für FIDO U2F verliehen. Durch die Entkoppelung von FIDO U2F von einem zertifizierten Hardware-Element hin zu einer zentralisierten Serverlösung wird eine maßgebliche Flexibilisierung in den Anwendungsmöglichkeit sowie eine Plattformunabhängigkeit auf Clientseite erreicht.

Durch die Wahl eines modularen Systems zum Schlüsselmanagement und der Ausführung von kryptographischen Operationen erhöht sich darüber hinaus die Attraktivität für BenutzerInnen. Es entfällt die Notwendigkeit ein Hardware-Element mitzuführen, CrySIL begünstigt die Verwendung unterschiedlicher Identitäten oder Pseudonyme bei verschiedenen Diensten und ist zur Kommunikation lediglich auf eine funktionierende Internetverbindung angewiesen.