30. Dezember 2022 Studie · Mobile Sicherheit · Web & Cloud · 13 Min. Lesezeit

Verwendung eines Android-Smartphones als Schlüsselspeicher für Web Authentication

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

Stand 2022-12-30
Quelle https://technology2.a-sit.at/veroeffentlichungen/android-smartphone-als-schluesselspeicher-fuer-webauthn/

In diesem Bericht wird gezeigt, wie ein Android-Smartphone am Rechner als sicherer Schlüsselspeicher für Web Authentication verwendet werden kann. Dazu wird ein sicherer Kanal zwischen Smartphone und Rechner aufgebaut. Eine Applikation am Android-Smartphone verwaltet das Schlüsselmaterial. Bei der Registrierung bei einem Webdienst wird das Schlüsselmaterial in der sicheren Hardware des Smartphones angelegt. Dadurch kann das Schlüsselmaterial selbst auf kompromittierten Smartphones nicht extrahiert werden. Die Verwendung des Schlüsselmaterials zur Anmeldung wird von der Benutzerin bzw. von dem Benutzer durch biometrische Authentifizierung bestätigt.

1. Einleitung - 1 2. Vorwissen - 2 2.1. FIDO2 - 2 2.2. Web Authentication (WebAuthn) - 2 2.3. Android Keystore - 2 2.4. Verwandte Arbeiten - 3 3. Konzept - 3 3.1. Verbindungsaufbau - 3 3.2. Registrierung - 5 3.3. Authentifizierung - 6 4. Evaluierung - 7 5. Fazit - 8 Literaturverzeichnis - 8 1. Einleitung Der Einsatz von passwortbasierten Authentifizierungsmethoden ist allgegenwärtig. Die Sicherheit dieser Authentifizierungsmethode hängt jedoch stark von der Auswahl des Passwortes ab. So kann ein zu kurzes oder simples Passwort durch automatisiertes Ausprobieren erraten werden. Auch die Wiederverwendung derselben Passwörter über verschiedene Dienste hinweg ist ein Problem. Um die Sicherheit zu erhöhen, schwenken deswegen viele Betreiber dazu um einen zweiten Authentifizierungsfaktor anzubieten bzw. vorauszusetzen. Als solch ein zweiter Faktor kann ein Hardwaretoken eingesetzt werden. Dieser speichert Schlüsselmaterial sicher im Token, welches bei der erstmaligen Anmeldung bei einem Dienst erstellt wird. Dabei kann es sich um einen externen Token handeln welcher über eine Schnittstelle wie USB angeschlossen wird oder um einen in das Gerät integrierten hardwarebasierten Schlüsselspeicher. Externe Tokens haben jedoch den Nachteil, dass sie extra angeschafft werden müssen, während interne Tokens an das jeweilige Gerät gebunden sind. Um diese Nachteile zu umgehen, wird in diesem Bericht ein Konzept vorgestellt, wie ein bestehendes Android-Smartphone als sicherer Schlüsselspeicher für Web Authentication eingesetzt werden kann.

2. Vorwissen In den folgenden beiden Abschnitten wird auf FIDO 2 und dem aus diesem Projekt entstandenen Web Authentication Standard näher eingegangen. Weiters wird im darauffolgenden Abschnitt der Android Keystore näher beleuchtet, da dieser für die sichere Schlüsselspeicherung unter Android zuständig ist.

2.1. FIDO2 Das FIDO2-Projekt wurde mit dem Ziel gegründet eine sichere Authentifizierungslösung im Browser zu ermöglichen. Im Kern besteht das Projekt aus den beiden Protokollen Web Authentication (WebAuthn) und Client to Authenticator (CTAP). Web Authentication beschreibt hierbei das Protokoll zur Registrierung und Authentifizierung des Clients bei einem Webdienst. CTAP regelt die Kommunikation zwischen dem Client und dem Authentifizierungsgerät auf dem die Schlüssel gespeichert sind.

2.2. Web Authentication (WebAuthn) Web Authentication [1] wurde von der W3C als Standard definiert und ist mittlerweile in allen relevanten Browsern verfügbar. Im Rahmen einer Registrierung über Web Authentication wird ein Schlüsselpaar bestehend aus privatem und öffentlichem Schlüssel erstellt. Der private Teil des Schlüsselpaares verbleibt dabei am Authentifizierungsgerät, während der öffentliche Teil an den Webdienst gesendet wird. Der Server verwendet den öffentlichen Schlüssel zum Nachweis die Identität des Benutzers. Da der öffentliche Teil des Schlüsselpaares ohne den privaten Schlüssel nutzlos für einen Angreifer ist, verringert sich so die Wertigkeit von geleakten Authentifizierungsdatenbanken im Vergleich zu einer reinen passwortbasierten Authentifizierung.

Für jeden Webdienst wird bei der Registrierung mit WebAuthn am Authentifizierungsgerät ein eigenes Schlüsselpaar erstellt. Nur das für den Webdienst erstellte Schlüsselpaar kann auch zur Authentifizierung bei diesem verwendet werden. Durch die vorgeschriebene sichere Speicherung am Authentifizierungsgerät kann auch ein kompromittierter Client die privaten Schlüssel nicht auslesen. Ebenso sendet das Authentifizierungsgerät ein Zertifikat an den Webdienst, mit Hilfe dessen der Dienst feststellen kann, dass es sich um ein vertrauenswürdiges Authentifizierungsgerät handelt.

2.3. Android Keystore Der Keystore [2] ist für die sichere Speicherung sowie Verwendung von Schlüsselmaterial unter Android zuständig. Dazu greift dieser auf die Hardwarefähigkeiten moderner Android-Smartphones zurück. Die Speicherung des durch den Keystore generierten Schlüsselmaterials erfolgt in der sicheren Hardware des Smartphones. Dieses besitzt in der Regel eine Trusted Execution Environment (TEE), welche Teil des System-on-Chips (SoC) des Smartphones ist oder eine Secure Enclave (SE). Zweitere ist physikalisch vom SoC getrennt und besitzt dadurch erhöhte Sicherheitsmerkmale wie einen verbesserten Schutz gegen gewisse Arten von Angriffen (beispielsweise Seitenkanalattacken). Das Ziel des Systems ist, dass das zu schützende Schlüsselmaterial auch von einem kompromittierten Android-System nicht ausgelesen werden kann. Dazu wird sichergestellt, dass das Schlüsselmaterial die sichere Hardware nicht verlässt. Das bedeutet, dass alle Operationen welche das Schlüsselmaterial betreffen in der gesicherten Hardware ausgeführt werden müssen. Die sichere Hardware unterstützt dabei nur eine begrenzte Auswahl an Algorithmen sowie Schlüsselgrößen.

2.4. Verwandte Arbeiten Es gibt bereits einige Lösungen, welche es erlauben das Smartphone als Schlüsselspeicher zur Zwei-Faktor-Authentifizierung einzusetzen. Auf einige der populärsten und die Unterschiede zu unserer Umsetzung wird in diesem Abschnitt eingegangen. Google selbst liefert eine solche Lösung über die Play Services aus. Über einen langen Zeitraum konnte diese Umsetzung jedoch nur für die Anmeldung bei Diensten von Google verwendet werden [3]. Dabei werden die Schlüssel am Android Smartphone gespeichert. Die Verbindung zwischen Smartphone und Rechner erfolgt prinzipiell via

Bluetooth oder USB. Zur Herstellung einer Verbindung via Bluetooth wird jedoch ein proprietäres cloudbasiertes Protokoll verwendet. Dieses wird als „cloud-assisted Bluetooth Low Energy“ oder kurz caBLE beschrieben [4]. Im Jahr 2022 wurde diese Umsetzung auch für weitere Dienste abseits von Google freigeschaltet. Seither ist dieses System unter dem Namen Passkeys bekannt [5]. Zur Verwendung sind jedoch weiterhin die Google Play Services und ein Google-Konto erforderlich. Ebenso wird weiterhin nur der Google Chrome-Browser unterstützt.

Darüber hinaus gibt es noch weitere appbasierte Lösungen, wie beispielsweise Akamai MFA [6] oder Chiff [7]. Ähnlich wie die in diesem Dokument gezeigte Umsetzung bestehen diese Lösungen meist aus zwei Komponenten, einer Smartphone-Applikation und einer Desktopanwendung bzw. einer Browsererweiterung. Anders als unsere Lösung verwenden diese Dienste aber oft eine cloudbasierte Lösung zur Verbindungsherstellung bzw. zur Kommunikation zwischen Smartphone und Rechner. Schlussendlich können auch dedizierte FIDO-zertifizierte Hardware-Tokens zur Schlüsselspeicherung eingesetzt werden.

3. Konzept In diesem Kapitel wird präsentiert, wie ein Android-Smartphones als sicherer, durch biometrische Merkmale geschützter Schlüsselspeicher am Desktop verwendet werden kann. Zuerst wird dabei auf den Verbindungsaufbau zwischen Rechner und Smartphone eingegangen. Danach wird gezeigt, wie die initiale Registrierung bei einem Onlinedienst abläuft. Zum Schluss wird dargestellt, wie die Authentifizierung bei einem bereits registrierten Dienst funktioniert.

3.1. Verbindungsaufbau Zur Verbindung zwischen dem Smartphone und dem Rechner empfiehlt der Standard die Schnittstellen NFC, Bluetooth und USB [8]. Da die meisten Desktop-Rechner jedoch keine NFC-Schnittstelle haben, wird diese Verbindungsmöglichkeit in diesem Bericht nicht weiter eruiert. Eine Verbindung vom Android-Smartphone zum Rechner per USB ist zwar schnittstellentechnisch möglich, jedoch wird diese Möglichkeit von keinem Browser unterstützt. Chrome unterstützt zwar prinzipiell das Installieren von Systemabbildern auf Android-Smartphones, jedoch muss das Android-Gerät dazu im Entwicklungsmodus sein, wodurch ein Großteil der Nutzerinnen und Nutzer bereits ausgeschlossen wird. Weiters hat eine App mit normalen Rechten keinen Zugriff auf diese Verbindung zum Browser, selbst wenn der Entwicklermodus am Gerät aktiviert ist. Dadurch kann auch USB nicht sinnvoll eingesetzt werden.

Eine Verbindung des Smartphones und des Browsers am Rechner über Bluetooth ist möglich. Es gibt jedoch auch hier die Einschränkung, dass (noch) nicht jeder Desktop-Browser auf jedem Betriebssystem eine Verbindung via Bluetooth unterstützt. Als zusätzliche Option kann jedoch auch eine Verbindung zwischen dem Smartphone und dem Rechner über das lokale Netzwerk aufgebaut werden. Da dies von den Browsern jedoch nicht nativ unterstützt wird, ist eine weitere Software am Rechner notwendig, welche die Kommunikation zwischen Browser und Smartphone übernimmt. Da der Datenverkehr im lokalen Netzwerk ggf. von anderen Nutzerinnen und Nutzern mitgeschnitten werden kann, ist eine gesicherte Verbindung zwischen der Software und dem Smartphone unumgänglich. Wie der Verbindungsaufbau und die Kommunikation zwischen Software und Rechner ablaufen kann, wird im folgenden Absatz beschrieben. Diese Sicherheitsmaßnahmen sind jedoch auch bei einer Bluetooth-Verbindung ratsam, da alle Applikationen am Android-Smartphone die die Bluetooth-Verbindung angefordert haben die Möglichkeit besitzen sämtlichen Datenverkehr mitzuschneiden.

In Abbildung 1 wird gezeigt wie der Verbindungsaufbau zwischen Rechner und Smartphone stattfindet. Bei der Verbindung über das lokale Netzwerk wird am Rechner ein Server gestartet, der zwischen Smartphone und Browser vermittelt (im Diagramm Verbindungskomponente genannt). Die Verbindung wird über TLS abgesichert um zu verhindern, dass andere Nutzerinnen und Nutzer im selben Netzwerk Zugriff auf den Authenticator haben. Dazu wird bei der Einrichtung der Verbindungskomponente ein Schlüsselpaar sowie ein selbstsigniertes Zertifikat erstellt. Um eine gesicherte Verbindung aufzubauen, muss dieses Zertifikat über einen sicheren Kanal auf das Smartphone übertragen werden. Diese Übertragung erfolgt über einen QR-Code. Dazu öffnet die Verbindungskomponente eine Webseite, die den QR-Code anzeigt. Dieser QR-Code wird durch den Nutzer bzw. die Nutzerin mit der App am Smartphone gescannt. Die Smartphone-App extrahiert das Zertifikat sowie die

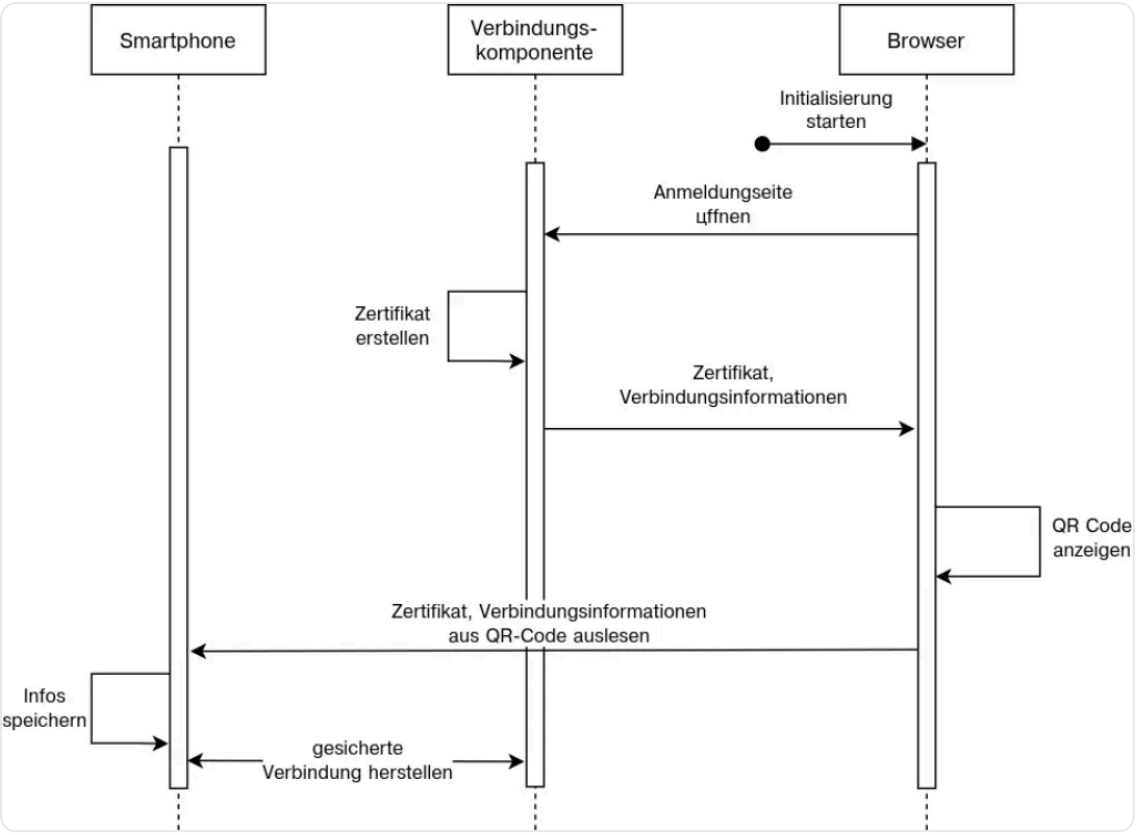


Abbildung 1: Prozess zur Herstellung einer Verbindung zwischen Rechner und Smartphone.

ebenfalls im QR-Code enthaltene URL. Daraufhin verwendet die App diese Informationen, um eine über TLS gesicherte Verbindung zur Verbindungskomponente aufzubauen. Die im QR-Code enthaltenen Informationen werden von der Smartphone-App für alle weiteren Verbindungsherstellungen gespeichert. Das Scannen des QR-Codes ist also nur bei der ersten Verbindungsherstellung notwendig.

3.2. Registrierung In Abbildung 2 wird der Registrierungsprozess gezeigt. Der Nutzer bzw. die Nutzerin startet den Registrierungsprozess im Browser des Rechners. Dieser sendet dann eine Registrierungsanfrage an den Dienst. Diese Anfrage enthält den gewählten Nutzernamen sowie verschiedene Optionen, wie Attestation Type und Authenticator Selection. Der Server überprüft die Anfrage und erstellt nach erfolgreicher Überprüfung ein PublicKeyCredentialCreationOptions Objekt. Dieses enthält Informationen über den Dienst selbst, über den anzulegenden Nutzer, zu verwendende Algorithmen zur Erstellung des Schlüssels, sowie Anweisungen an das Authentifizierungsgerät. Ebenso wird eine durch den

Dienst generierte Challenge hinzugefügt. Dieses Objekt wird an den Browser zurückgeschickt und von diesem an das Smartphone weitergeleitet. Das Smartphone liest die Anfrage aus und authentifiziert die Nutzerin bzw. den Nutzer durch Verifizierung des biometrischen Faktors. Nach erfolgreicher Authentifizierung erstellt und speichert das Smartphone das Schlüsselpaar sicher im Android Keystore. Daraufhin wird als Antwort das PublicKeyCredential-Objekt erstellt. Dieses enthält eine eindeutige Identifizierungsnummer des Credentials sowie das verschachtelte Antwort-Objekt. Dieses wiederum enthält die Felder clientDataJSON sowie das Attestation-Objekt. Das Feld clientDataJSON beinhaltet die vom Webdienst gesendete Challenge, die Herkunft (also in diesem Fall die URL des Webdienstes) und den Typ. Dieser ist für die Registrierung auf webauthn.create festgelegt. Das Attestation-Objekt enthält die Authentifizierungsdaten, das Format und das Attestation Statement. Die Authentifizierungsdaten beinhalten dabei den öffentlichen Teil des erstellten Schlüssels. Das Attestation Statement besteht im Wesentlichen aus einer Signatur. Diese wird über den Authentifizierungsdaten und dem clientDataJSON-Objekt erstellt.

Das vom Smartphone erstellte PublicKeyCredential-Objekt wird an den Browser am Desktop weitergeleitet. Dieser sendet das Objekt an den Webdienst. Der Webdienst verifiziert die Challenge, die im Objekt enthaltenen Informationen über den Webdienst, und die gesendete Signatur. Nach erfolgreicher Verifizierung werden die Identifizierungsnummer des Credentials und der öffentliche Schlüssel gespeichert sowie ein Signaturzähler angelegt. Der Erfolg (bzw. im Fehlerfall der Misserfolg) wird an den Browser übermittelt.

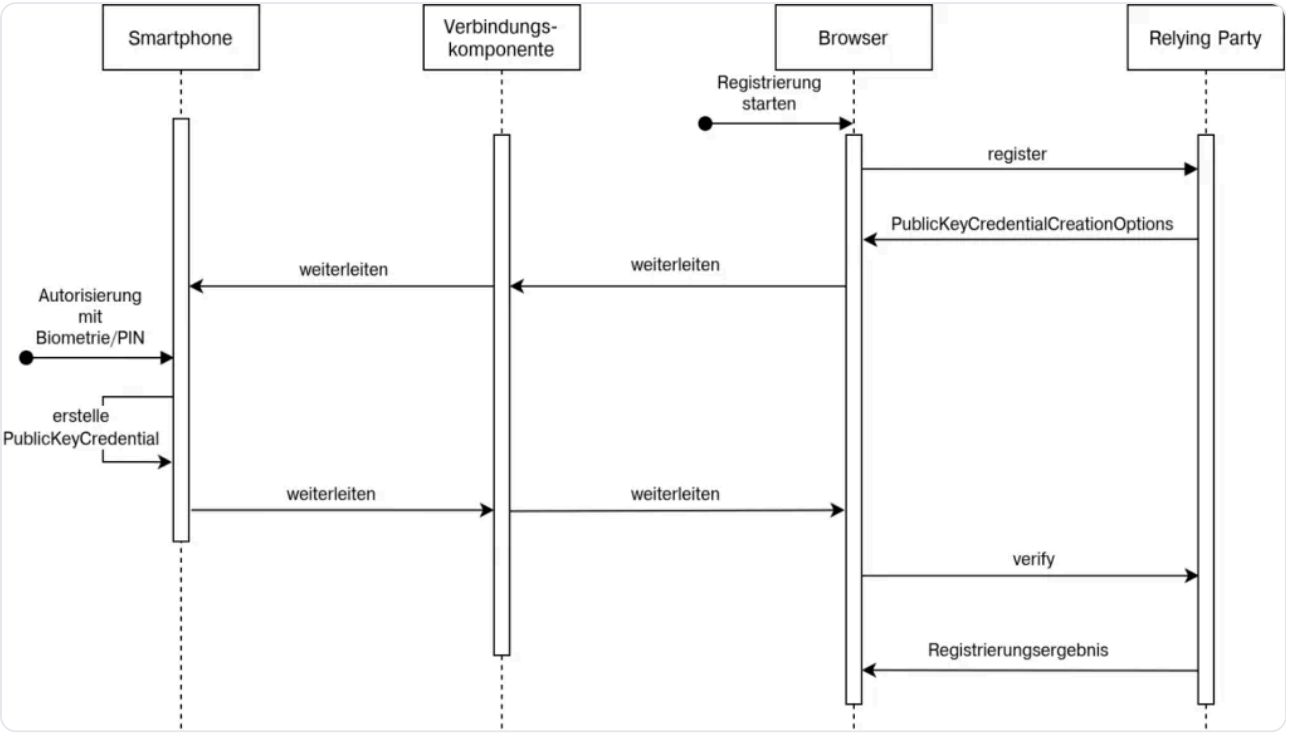


Abbildung 2: Ablauf der Registrierung bei einem Webdienst.

3.3. Authentifizierung Abbildung 3 zeigt den Ablauf einer Authentifizierung bei einem zuvor registrierten Webdienst. Der Prozess wird durch die Nutzerin bzw. den Nutzer durch Aufruf der Anmeldeseite gestartet. Der Webbrowser am Rechner sendet daraufhin eine Anmeldeanfrage an den Webdienst. Dieser antwortet mit dem PublicKeyCredentialRequestOptions-Objekt. Dieses enthält die ID des Webdienstes und eine durch den Webdienst erstellte Challenge. Ebenso sendet der Webdienst eine Liste an erlaubten

Authentifizierungsgeräten mit. Diese enthält alle beim Webdienst für die Nutzerin bzw. den Nutzer registrierten Geräte. Das Objekt wird an das Android-Smartphone weitergeleitet. Wie bei der Registrierung bestätigt die Nutzerin bzw. der Nutzer den Anmeldevorgang durch Scannen der hinterlegten Biometrie. Das Smartphone erstellt daraufhin ein PublicKeyCredential-Objekt. Dieses enthält eine Identifizierungsnummer sowie das Antwort-Objekt. Dieses enthält wiederum das clientDataJSON- und AuthenticatorData-Objekt sowie eine Signatur. Das clientDataJSON-Objekt ist bei der Authentifizierung auf webauthn.get konfiguriert.

Das erstellte PublicKeyCredential-Objekt wird wieder an den Webbrowser und weiter an den Webdienst gesendet. Dieser überprüft die Signatur mit Hilfe des während der Registrierung gespeicherten öffentlichen Schlüssels. Im Erfolgsfall wird der Signaturzähler um eins erhöht und der Nutzer bzw. die Nutzerin über die erfolgreiche Anmeldung informiert.

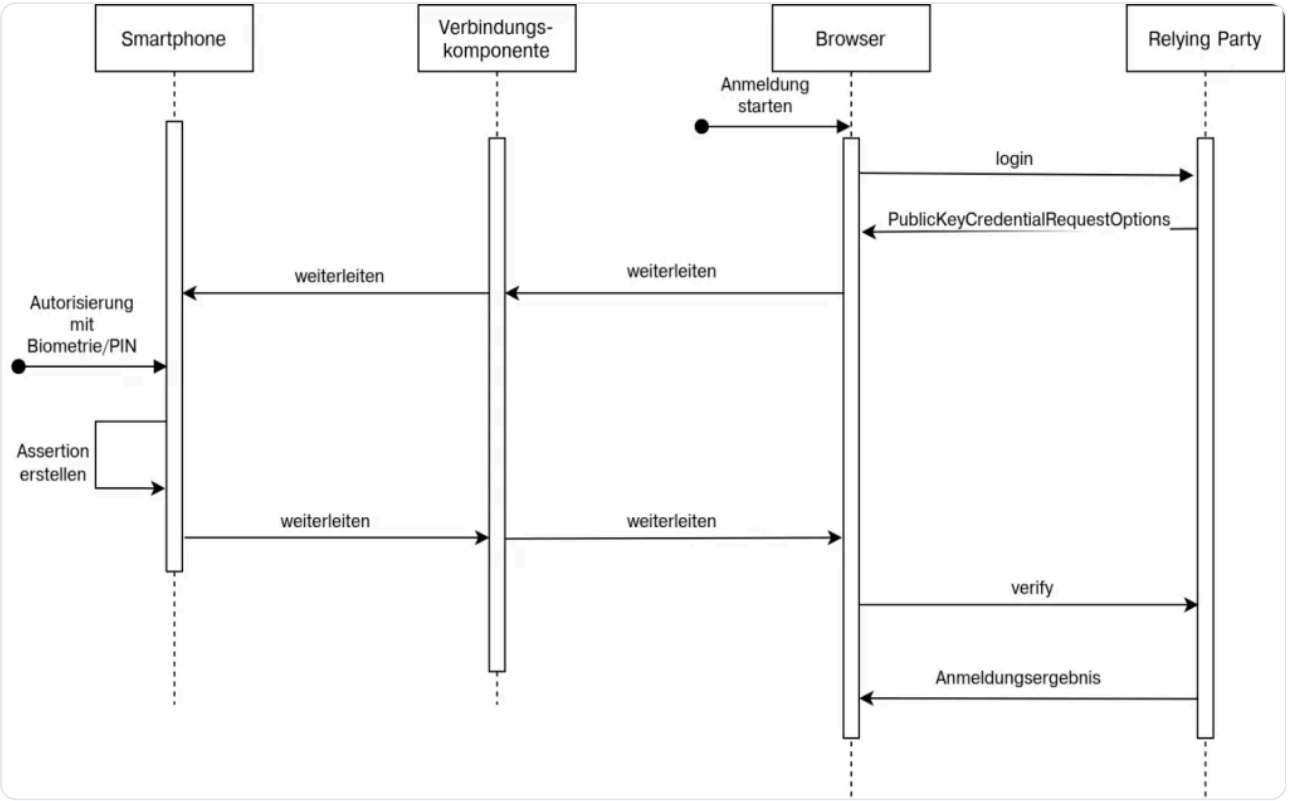


Abbildung 3: Ablauf der Authentifizierung bei einem Webdienst nach erfolgter Registrierung.

4. Evaluierung In den folgenden Absätzen wird die Sicherheit des vorgestellten Konzeptes evaluiert und auf mögliche Bedrohungen eingegangen. Dazu wird zuerst auf die Unterschiede des dargestellten Konzepts im Vergleich zu klassischen FIDO-Tokens eingegangen. Danach wird gezeigt, welche Vorkehrungen getroffen wurden um mögliche Bedrohungen zu kontern.

Im Vergleich zu einem USB- oder NFC-Token hat das vorgestellte Konzept sicherheitstechnisch einige Vorteile, bietet jedoch in gewissen Aspekten eine größere Angriffsfläche. So ist die Verbindung per NFC bzw. USB durch die physische Nähe schwieriger abhörbar als Bluetooth bzw. das lokale Netzwerk. Ebenso können auf dem Android-Smartphone andere, potentiell bösartige, Applikationen laufen. Dafür bieten die Tokens im Vergleich zu Smartphones nur eine sehr eingeschränkte Bedienungsmöglichkeit, d.h. die Bestätigung zur Verwendung einer Anmeldeinformation erfolgt durch den Klick einer Taste oder durch

Lesen des Fingerabdrucks. Im Vergleich dazu können auf dem Android-Smartphone detaillierte Informationen zur Verwendung angezeigt werden. Ebenso unterstützten moderne Android-Smartphones oft verschiedene Arten der biometrischen Authentifizierung, wie Gesichts- oder Irisscan.

Bei der Verbindung über das lokale Netzwerk könnte ein Angreifer im selben Netzwerk als Man-in-the-Middle die gesendeten Nachrichten lesen bzw. austauschen. Um dies zu verhindern wird die Verbindung zwischen Smartphone und Rechner via TLS abgesichert. Der Austausch des dafür notwendigen Zertifikats erfolgt beim erstmaligen Verbindungsaufbau über einen weiteren Kanal (in diesem Fall über einen QR-Code) um den Austausch des Zertifikats durch einen Angreifer zu verhindern.

Um die Sicherheit weiter zu erhöhen ist es möglich die Vertrauenswürdigkeit eines Android-Smartphones durch Kontrollieren der mitgesendeten Attestation sicherzustellen. Dadurch kann auch festgestellt werden, ob ein Smartphone über aktuelle Sicherheitsaktualisierungen verfügt.

Da das Android-Smartphone in der Regel unterwegs verwendet wird ist könnte es verloren gehen oder gestohlen werden. In dem Fall wären die gespeicherten Schlüssel prinzipiell in der Hand des Finders bzw. des Diebes. Durch die Speicherung der Schlüssel im Keystore können die Schlüssel jedoch nicht extrahiert werden. Ebenso ist selbst für die Verwendung des Schlüsselmaterials eine biometrische Bestätigung der rechtmäßigen Besitzerin bzw. des rechtmäßigen Besitzers notwendig. Dadurch ist auch bei Verlust des Geräts keine Anmeldung durch eine dritte Person möglich.

5. Fazit In diesem Bericht wurde gezeigt, wie ein Android-Smartphone als Schlüsselspeicher für Web Authentication verwendet werden kann. Dazu wird die sichere Hardware der Smartphones zur Speicherung der Schlüssel verwendet. Zur Erstellung bzw. zur Verwendung der gespeicherten Schlüssel ist eine aktive Verbindung zwischen Smartphone und Rechner nötig. Wie sich gezeigt hat ist eine Verbindung per USB zwischen einer Android-App und dem Rechner nicht möglich. Ebenso ist eine Verbindung per NFC nicht zielführend da diese Schnittstelle am Rechner kaum verfügbar ist. Dadurch kann die Verbindung zwischen dem Rechner und dem Smartphone über das lokale Netzwerk, oder falls von Browser und Betriebssystem unterstützt, per Bluetooth erfolgen. Durch die größere Angriffsfläche dieser Verbindungstypen und die dadurch entstehende Möglichkeit eines Man-in-the-Middle-Angriffs ist eine gesonderte Sicherung dieser Verbindung erforderlich. Im Bericht wurde gezeigt, wie diese Verbindung mittels Scans eines QR-Codes aufgebaut werden kann. Zur Autorisierung der Nutzerin bzw. des Nutzers bei der Verwendung des gespeicherten Schlüsselmaterials zur Anmeldung kommt die Biometrie des Smartphones zum Einsatz um unrechtmäßigen Einsatz zu verhindern.

Literaturverzeichnis

[1] W3C, „Web Authentication: An API for accessing Public Key Credentials,“ [Online]. Available: https://www.w3.org/TR/webauthn-2.

[2] Android Developers, „Android Keystore system,“ [Online]. Available: https://developer.android.com/training/articles/keystore. [Zugriff am 12 12 2022].

[3] Google Inc, „Integrierten Sicherheitsschlüssel Ihres Smartphones verwenden,“ [Online]. Available: https://support.google.com/accounts/answer/9289445. [Zugriff am 30 12 2022].

[4] Duo Security, „Developments to WebAuthn and the FIDO2 Framework,“ [Online]. Available: https://duo.com/blog/developments-to-webauthn-and-the-fido2-framework. [Zugriff am 28 12 2022].

[5] Passkeys, „Passkeys,“ [Online]. Available: https://www.passkeys.com/. [Zugriff am 30 12 2022].

[6] Akamai, „Akamai MFA - Multi Factor Authentication Solution,“ [Online]. Available: https://www.akamai.com/products/akamai-mfa. [Zugriff am 29 12 2022].

[7] Chiff, „Chiff FAQ,“ [Online]. Available: https://www.chiff.app/faq/. [Zugriff am 30 12 2022].

[8] W3C, „Web Authentication: An API for accessing Public Key Credentials,“ [Online]. Available: https://www.w3.org/TR/webauthn-2/#enumdef-authenticatortransport.