Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-SIT

Seidlgasse 22 / 9, 1030 Wien
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

Inffeldgasse 16a, 8010 Graz
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461
ZVR: 948166612
UID: ATU60778947

BEREITSTELLUNG VON SMARTPHONE-FEATURES AM DESKTOP

Version 1.0 vom 10.10.2020

Bernd Prünster – bernd.pruenster@a-sit.at

Abstract/Zusammenfassung: Aktuell genutzte Personal-Computing-Geräte wie Smartphones, Laptops und traditionelle Desktop-Computer sind auf unterschiedliche Anwendungsfälle ausgerichtet, ergänzen einander jedoch nur bedingt, da es nur begrenzte Möglichkeiten gibt, z.B. auf Smartphones vorhandene Sensoren oder kryptografische Hardware auch am Desktop einzusetzen. Auch die auf beiden Plattformen verfolgen Trusted-Computing-Modelle unterscheiden sich gravierend voneinander, was sich insbesondere in unterschiedlicher Zugänglichkeit und unterschiedlich einfacher, bzw. komplexer Nutzungsmöglichkeiten niederschlägt.

Im Rahmen dieses Projekts wurde ein Konzept erarbeitet, welches veranschaulicht, wie Sensoren, sowie Trusted-Computing-Features von Smartphones am Beispiel Android auch am Desktop verfügbar gemacht werden können. Im Zusammenhang mit der breiten Verfügbarkeit von Biometrie und kryptografischer Hardware in Smartphones ergeben sich auch Möglichkeiten, die Sicherheit von traditionellen Desktop-Workflows durch die Kopplung an Smartphones zu erhöhen und vor allem breit verfügbar zu machen. Dieses Dokument präsentiert die Projektergebnisse zu diesem Thema.

Inhaltsverzeichnis

Inhaltsverzeichnis 1
1 Einleitung 2
2 Technologische Hintergründe 2
2.1 Remote-Attestation und Trusted-Computing unter Android 3
2.2 Vernetzung von Geräten über libp2p 4
3 Vernetzung von Desktop und Smartphone 5
3.1 Enrolment 5
3.2 Biometrische Authentifizierung 6
3.3 Auslagerung sensibler Operationen 7
4 Fazit 8
Referenzen 9

© A-SIT, Partner für Sicherheitsfragen

1 Einleitung

Aktuell genutzte Personal-Computing-Geräte wie Smartphones, Laptops und traditionelle Desktop-Computer sind auf unterschiedliche Anwendungsfälle ausgerichtet, ergänzen einander jedoch nur bedingt, da es nur begrenzte Möglichkeiten gibt, z.B. auf Smartphones vorhandene Sensoren oder kryptografische Hardware auch am Desktop einzusetzen. Bestrebungen, auf Smartphones gespeichertes kryptografisches Schlüsselmaterial zu Authentifizierungszwecken für Webservices heranzuziehen, sind konzeptionell im Rahmen von *WebAuthn*[1] auf standardisiertem Wege möglich, adressieren aber nur einen Aspekt der Nutzung von Smartphone-Eigenschaften außerhalb von Smartphone-App-Ökosystemen. Dasselbe gilt für Lösungen wie *Share GPS*[2], welche GPS-Sensoren an Desktop-Betriebssystem-Sensor-APIs anbinden.
Der Fokus bestehender Systeme auf einen Aspekt ist für sich genommen nicht notwendigerweise eine Einschränkung, da mehrere Lösungen komplementär eingesetzt werden können. Im Kontext von IT-Sicherheit stellt sich jedoch vielmehr die Frage nach der Nutzung von Sicherheitsfeatures auf Smartphone-Plattformen, beziehungsweise das Verfügbarmachen ebendieser auf Desktop-Systemen, deren Sicherheitsmodell sich grundlegend von aktuellen Smartphone-Betriebssystemen unterscheidet. Im Zusammenhang mit der breiten Verfügbarkeit von biometrischer und kryptografischer Hardware in Smartphones ergeben sich potentiell neue Möglichkeiten, die Sicherheit von traditionellen Desktop-Workflows durch die Kopplung an Smartphones zu erhöhen, da im Desktop-Bereich z.B. das sichere Ablegen von kryptografischem Material in Hardware nur bedingt möglich ist. Ziel dieses Projekts ist die Demonstration der Einbindung von auf Smartphones vorhandener kryptografischer Hardware im Rahmen von Desktop-Anwendungen zur Erhöhung der Sicherheit.
Im Allgemeinen ergibt sich jedoch durch entsprechend gestaltete Einbindung aktueller Smartphones in Desktop-Workflows die Möglichkeit, Trusted-Computing am Desktop, inklusive Trusted IO zur Verfügung zu stellen, ohne dass dabei komplexe Protokolle, oder die Einbindung vertrauenswürdiger Dritter erforderlich werden. Zwar ist die Rechenleistung mobiler Geräte beschränkt, einzelne kritische Operationen, sowie Ein- und Ausgaben können jedoch ausgelagert werden. Im Rahmen dieses Projekts wurde ein Konzept erarbeitet, welches die prinzipielle Machbarkeit einer derartigen Vernetzung darlegt. Darüber hinaus wurde ein Demonstrator entwickelt, dessen Quellcode diese Konzepte veranschaulicht.

Nachfolgend werden die technologischen Hintergründe zu diesem Konzept beleuchtet. Abschnitt 3 beschreibt anschließend das erarbeitete Konzept, dessen Potential abschließend in Abschnitt 4 diskutiert wird. Darüber hinaus wird ein Ausblick über mögliche zukünftige Entwicklungen auf Basis der vorgestellten Arbeit gegeben.

2 Technologische Hintergründe

Dreh- und Angelpunkt des im Rahmen dieses Projekts entwickelten Ansatzes ist das Vertrauens- und Sicherheitsmodell von Mobilplattformen, das sich diametral vom Desktop-Bereich unterscheidet. Sowohl unter Apples *iOS*, als auch unter Googles *Android*-Systemen wurde von Anfang an auf restriktive Sicherheitsmodelle gesetzt, welche einerseits striktes Sandboxing von Applikationen und streng geregelten benutzerseitigen Zugriff auch auf das eigene Gerät beinhalten, sowie andererseits zunehmend um Trusted-Computing-Konzepte wie verifizierte Bootvorgänge bis hin zu signierten Betriebssystemabbildern erweitert wurden. Aus Sicht eines Benutzers, bzw. einer Benutzerin hat das zweierlei Konsequenzen: Einerseits kann selbst ein Geräteeigentümer, bzw. eine Eigentümerin nicht „frei“ über dieses Verfügen. Die Installation eines alternativen Betriebssystems ist üblicherweise unterbunden, ebenso wie wahlfreier Zugriff auf das Dateisystem. Gleichzeitig gehen mit diesem restriktiven Modell jedoch enorme Sicherheitsgewinne einher. Das Betriebssystem garantiert beispielsweise, dass Daten einer Applikation nicht von anderen installierten Applikationen ausgelesen oder gar verändert werden können. Ebenso muss jede Applikation den Nutzer, bzw. die Nutzerin explizit um Erlaubnis fragen, falls auf persönliche Daten oder Sensoren (z.B. GPS, Kamera, Mikrofon, …) zugegriffen werden soll.

[1] https://webauthn.io/
[2] https://play.google.com/store/apps/details?id=com.jillybunch.shareGPS&hl=en

Dieses Modell steht im krassen Gegensatz zum Desktop-Computing-Ansatz, im Rahmen dessen es seit je her möglich war, ein Betriebssystem nach Wahl zu installieren, de-facto voller Zugriff auf alle Aspekte eines Systems gegeben ist und auch Hardwarekomponenten ausgetauscht und so das eigene System den persönlichen Vorlieben angepasst werden kann. Eine Isolation von Anwendungen zueinander wird zwar auch im Desktop-Bereich vorangetrieben, tatsächlich ist es Stand 2020 jedoch nach wie vor so, dass eine Applikation, welche am System ausgeführt wird, auf die Daten anderer Applikationen, sowie auch auf persönliche Daten zugreifen kann.
Vergleicht man diese beiden Sicherheitsmodelle, wird offensichtlich, warum beispielsweise immer wieder über Kryptotrojaner[3] am Desktop berichtet wird, derartige Schadsoftware auf Mobilplattformen jedoch bereits konzeptionell verunmöglicht wird. Formal betrachtet ergibt sich daher am Desktop keinerlei verifizierbare Vertrauensbasis, da selbst ein integres Betriebssystem augenscheinlich keinen Schutz bietet. Desktop-zentrische Trusted-Computing-Konzepte wie die von Intel entwickelten *Software Guard Extensions* (SGX)[4] verdeutlichen dies. Das SGX-Sicherheitsmodell basiert auf einem isolierten Bereich in der CPU, welcher unabhängig vom Betriebssystem agiert. Anwendungen, Betriebssystem und Peripherie werden explizit als potentiell bösartig angesehen und lediglich eine minimale, scharf abgegrenzte Trusted-Computing-Basis in der CPU wird als vertrauenswürdig klassifiziert. Als Konsequenz daraus sind ohne weiteres weder verifizierbare, bzw. vertrauenswürdige Eingaben noch Ausgaben möglich, wodurch Trusted-Computing am Desktop sich tatsächlich auf eine wörtliche Interpretation des Begriffs bezieht [1].

2.1 Remote-Attestation und Trusted-Computing unter Android

Unter Android ist Trusted-Computing inklusive trusted IO nicht nur aus Nutzersicht möglich, sondern auch Entwicklern und Entwicklerinnen von Applikationen, bzw. Servicebetreibern zugänglich. In einem vorangegangenen A-SIT-Projekt wurde ein entsprechendes Konzept erarbeitet und praktisch umgesetzt, das es ermöglicht, die Authentizität und Integrität einer Applikation aus der Ferne zu überprüfen [2]. Dabei wird ein Mobilgerät als Gesamtsystem in den Status einer Trusted-Computing-Basis angehoben. Hierfür wird eine Kombination aus am Gerät vorhandener kryptografischer Hardware zur Verwaltung kryptografischer Schlüssel, verifiziertem Bootvorgang und diverser Eigenschaften des Android-Sicherheitsmodells [3] in Verbindung mit *Remote Attestation*, der Integritätsprüfung aus der Ferne, eingesetzt. Die technische Basis für diese Umsetzung von Trusted Computing unter Android in dieser Form ist dem entsprechenden Projektbericht zu entnehmen, weshalb an dieser Stelle lediglich ein Überblick über dieses Verfahren gegeben wird.
Im Kern basiert das Verfahren darauf, dass in vertrauenswürdigen Hardwarekomponenten am Smartphone vom Hersteller ein public-private Keypair, sowie eine zugehörige Zertifikatskette hinterlegt werden. Die Wurzel dieser Kette ist ein Google-Root-Zertifikat. Darüber hinaus sind Informationen über Widerrufspunkte (wenn auch nicht CRL und OCPS-konform) in dieses Zertifikat kodiert. Beim Erstellen eines symmetrischen Schlüssels oder eines asymmetrischen Schlüsselpaares aus einer Smartphone-Applikation kann optional eine kryptografische Nonce, die so genannte *Attestation Challenge*, mitgegeben werden. Wenn vorhanden, fließt dieser Parameter in die Generierung eines Zertifikats ein, welches in der Hardware vom vom Hersteller hinterlegten private Key signiert wird. Dadurch entsteht eine Zertifikatskette, welche den neu erstellten Schlüssel, bzw. das Schlüsselpaar zertifiziert. Über zusätzliche X.509-Zertifikatserweiterungen werden Zustand, bzw. Integrität des Bootloaders, des Betriebssystems und der Applikation im Zertifikat hinterlegt. Eine entsprechende Auswertung dieser Erweiterungen, sowie Validierung der Zertifikatskette ermöglichen es, die Integrität des Geräts, Betriebssystems, sowie der Applikation zu verifizieren. Dadurch können auch Modifikationen durch den Benutzer bzw. die Benutzerin ausgeschlossen, bzw. erkannt werden. Folglich ist es Applikationsentwicklern bzw. Applikationsentwicklerinnen und Service-Providern möglich, die volle Kontrolle über installierte Applikationen zu behalten und auch sensible Operationen auf Android-Smartphones auszulagern.

Voraussetzung für dieses Verfahren ist jedoch das Vorhandensein entsprechender Hardware-Module im Smartphone, was für faktisch alle seit Mitte 2019 erschienenen Geräte der Fall ist. Durch

[3]Kryptotrojaner bezeichnet eine Klasse von erpresserischer Schadsoftware, welche Daten nicht zerstört, oder an Dritte überträgt, sondern verschlüsselt, und erst gegen Zahlung wieder entschlüsselt.
[4]https://software.intel.com/en-us/sgx

die Auswertung der in Zertifikatserweiterungen kodierten Parameter wie Betriebssystemversion, Patch-Level, usw. ist es möglich, ein Vertrauensniveau zu attestieren. Ein nicht aktuelles Betriebssystem kann als weniger vertrauenswürdig angesehen werden als ein aktuelles. Wird beispielsweise ein entsperrter Bootloader detektiert, so senkt dies das Vertrauensniveau drastisch, da in diesem Fall Modifikationen am Betriebssystem nicht ausgeschlossen werden können. Sofern ein Smartphone verwendet wird, das schlicht zu alt ist und die notwendige Funktionalität nicht unterstützt, kann keine Vertrauensbasis aufgebaut werden. Dieser Fall wird jedoch zunehmend irrelevant, da derartige Geräte kaum noch vertrieben werden.

2.2 Vernetzung von Geräten über libp2p

Die Bibliothek *libp2p*[5] bietet einen modularen Netzwerk-Stack für diverse Programmiersprachen an. Ziel ist die einheitliche, architekturunabhängige Abstraktion von Konnektivität, Transportverschlüsselung, ohne auf ein bestimmtes Anwendungsszenario abzuziehen. Genutzt wird libp2p beispielsweise in *IPFS*[6] und als Basis der nächsten Generation der Kryptowährung *Ethereum*[7]. Auf Grund der applikationsunabhängigen Ausrichtung und der einfachen Erweiterbarkeit wird libp2p mittlerweile beispielsweise auch als Teil von Hosting-Angeboten[8] produktiv eingesetzt.

libp2p ist nicht nur anwendungs-, bzw. schnittstellenseitig modular konzeptioniert, sondern unterstützt prinzipiell beliebige Transportprotokolle, wie TCP oder *QUIC* [4], aber auch WebSockets [5], um browserbasierte Anwendungen zu unterstützen. Das vorwärtskompatible Adressierungsschema von libp2p-Endpunkten lässt dabei beliebige Erweiterbarkeit zu. In Kombination mit integriertem Relaying von virtuellen Verbindungen über bestehende Kommunikationskanäle, lassen sich unterschiedlichste Plattformen miteinander vernetzen.
Dieses Adressierungsschema, das im Kern auf der *Multiformats*-Spezifikation[9] aufbaut wird auf allen Ebenen des libp2p-Stacks angewandt. Daraus ergibt sich Erweiterbarkeit auf höheren Schichten. Somit lassen sich beliebige Verfahren für Transportsicherheit einsetzen, und anwendungsspezifische Protokolle definieren. Die abstrakt definierten Kommunikationskanäle solcher Protokolle lassen sich dadurch auch über bestehende Verbindungen, welche potentiell auch auf Dritte als Relay zurückgreifen, aufbauen, selbst wenn diese die Protokolle zwischen den Endpunkten nicht unterstützen. Gleichzeitig kann durch dein Einsatz von TLS oder einem auf dem *Noise Protocol Framework* [6] basierendem Transportsicherheitsprotokoll Ende-zu-Ende-Verschlüsselung garantiert werden.
Im Zusammenhang mit Transportsicherheit werden selbstzertifizierende Identifikatoren verwendet, welche ebenfalls integraler Bestandteil des auf Multiformats basierenden Adressierungsschemas sind. libp2p schreibt in diesem Zusammenhang Diffie-Hellman-Ephemeral-basierte Verfahren vor und verwendet direkt public Keys und keine X.509-Zertifikate. Entsprechend muss lediglich das Erstellen und Verifizieren von Signaturen implementiert werden, um neue asymmetrische kryptografische Verfahren im Rahmen von Diffie-Hellman-Prozessen zu unterstützen. Darüber hinaus werden keine Vorgaben zur Verwaltung, bzw. Speicherung von Schlüsselmaterial gemacht, wodurch sich mit vergleichsweise wenig Aufwand beispielsweise HSMs integrieren lassen, da lediglich eine einzige kryptografische Operation (das Erstellen einer Signatur) HSM-seitig integriert werden muss. Auch wenn im Rahmen dieses Projekts die Vernetzung von lediglich einem Smartphone und einem Desktop-Computer behandelt wird, eröffnet die Nutzung von libp2p die Möglichkeit, eine darüber hinausgehende Verbindung beliebig vieler Instanzen im Rahmen verteilter Anwendungen umzusetzen.
Auf Grund dieser Eigenschaften wurde libp2p als Netzwerkstack für die Vernetzung von Smartphones, welche über kryptografische Hardware verfügen, und Desktop-Systemen ausgewählt.

[5] https://libp2p.io/
[6] https://ipfs.io/
[7] https://github.com/ethereum/eth2.0-specs/pull/1328
[8] https://textile.io/
[9] https://multiformats.io/

3 Vernetzung von Desktop und Smartphone

Dieses Projekt umfasst die Kombination eines Smartphones und eines Desktop-Systems. Es wird davon ausgegangen, dass es sich dabei um Geräte desselben Nutzers, bzw. derselben Nutzerin handelt. Entsprechend einfach kann das Vertrauens-, bzw. Sicherheitsmodell gehalten werden. Aufbauend auf dem in Abschnitt 2.1 beschriebenen mobilen Trusted-Computing-Ansatz wurde jedoch darauf geachtet, dass am Smartphone ausgeführte Operationen, bzw. deren Ergebnisse auch von Dritten als solche verifiziert werden können. Konkret kommen kryptografische Signaturen zum Einsatz, welche den Integritätsstatus des Smartphones beinhalten und diesen somit auch extern, bzw. nachträglich überprüfbar machen. Umgesetzt wurde die Lösung als Kombination von einer Desktop-Applikation mit einer Smartphone-Applikation.
Sowohl Smartphone als auch Desktop-Computer benutzen (wie für libp2p üblich) selbstzertifizierende Identifikatoren, wobei der zu Grunde liegende kryptografische Schlüssel am Smartphone an dessen kryptografische Hardware gebunden ist. Um diesen auch wie in Abschnitt 2.1 beschrieben attestierbar umzusetzen, fließt in dessen Generierung eine am Desktop generierte kryptografische Nonce ein. Damit wird gleichzeitig auch ein rudimentärer Enrolment-Prozess umgesetzt, welcher für den initialen Verbindungsaufbau notwendig ist. Nähere Details sind Abschnitt 3.1 zu entnehmen.
Der Verbund von Desktop und Smartphone ist strikt hierarchisch organisiert; auszuführende Operationen werden am Desktop angestoßen und vom Smartphone ausgeführt, welches die Ergebnisse retourniert. Umgesetzt wurde biometrische Authentifizierung, sowie beispielhaft die Auslagerung von (sensiblen) Rechenoperationen ans Smartphone. Details sind den Abschnitten 3.2 und 3.3 zu entnehmen.

3.1 Enrolment

Der initiale Verbindungsaufbau und Enrolment-Prozess wird vom Desktop aus angestoßen. Dieser erstellt ein public-private Keypair, welches die Basis für einen selbstzertifizierenden Identifikator bildet. Zusätzlich wird eine kryptografische Nonce generiert, welche in die Schlüsselgenerierung am Smartphone als Attestation-Challenge einfließt. Folglich muss diese Information bereits vor dem initialen Verbindungsaufbau erfolgen. Entsprechend umgesetzt ergibt sich dadurch der Vorteil, dass bereits beim Herstellen der ersten Verbindung beide Kommunikationsteilnehmer gegeneinander authentifiziert werden können. Konkret läuft dieser Prozess wie folgt ab:

1. Am Desktop wird ein public-private Keypair generiert, aus dem ein selbstzertifizierender Identifikator abgeleitet wird.
2. Am Desktop wird eine kryptografische Nonce generiert.
3. Beides wird in einen QR-Code kodiert, welcher am Bildschirm angezeigt wird.
4. Ein libp2p-baiserter Netzwerkknoten wird am Desktop gestartet, dessen Identifikator, sowie IP-Adresse und Port mittels lokalem Multicast-DNS periodisch in der lokalen Kollisionsdomäne verbreitet werden.
5. In der Smartphone-Applikation wird der Enrolment-Prozess mit dem Abscannen des QR-Codes gestartet.
6. Die Nonce wird aus dem QR-Code extrahiert und als Attestation-Challenge zur Schlüsselgenerierung verwendet.
7. Auf Basis des in Hardware generierten Schlüssels wird in der Smartphone-Applikation ein selbstzertifizierender libp2p-Identifikator generiert.
8. Analog zum Prozess am Desktop wird ein libp2p-basierter Netzwerkknoten gestartet, dessen Identifikator, sowie IP-Adresse und Port mittels lokalem Multicast-DNS periodisch in der lokalen Kollisionsdomäne verbreitet werden.
9. Anschließend wird der Identifikator des Desktop-Knotens aus dem zuvor gescannten QR-Code extrahiert und mit eingehenden Multicast-DNS-Informationen abgeglichen.
10. Sobald eine Kombination aus Identifikator, IP-Adresse und Port gefunden wird, welche zum im QR-Code enthaltenen Identifikator des Desktops passen, kann vom Smartphone aus eine Verbindung zum Desktop initiiert werden. Voraussetzung ist, dass sich Desktop-Gerät und Smartphone im selben Netzwerk befinden.

11. Ist eine Verbindung hergestellt, wird vom Smartphone die Zertifikatskette übermittelt, welche attestiert, dass der selbstzertifizierende Identifikator tatsächlich in Hardware auf einem integren, unmodifizierten Smartphone aus einer unkompromittierten Applikation heraus generiert wurde (siehe Abschnitt 2.1). Entsprechend wird ein Vertrauensniveau attestiert. Die separate Übermittlung einer Zertifikatskette nach dem Aufbau der Verbindung ist notwendig, da libp2p lediglich den zum Identifikator gehörigen public Key übermittelt.
12. Am Desktop wird einerseits überprüft, ob die libp2p-Identität des Smartphones mit der Zertifikatskette übereinstimmt, andererseits die Zertifikatskette selbst ausgewertet, um eine Aussage über die Integrität des Smartphones und der Applikation treffen zu können.
13. Das Ergebnis der Überprüfung wird an das Smartphone übermittelt.
14. Bei erfolgreicher Prüfung wird die Zertifikatskette von der Wurzel bis zum Smartphone-Identifikator in der Desktop-Applikation hinterlegt.

Ist dieser Prozess abgeschlossen, ist eine wechselseitige Bindung hergestellt. Durch die Übertragung von Nonce und Identifikator der Desktop-Instanz mittels QR-Code steht bereits vor dem Aufbau einer gesicherten Netzwerkverbindung ein zuverlässiger Kommunikationskanal zur Verfügung. Tatsächlich ist die übermittelte Nonce im Zeitraum zwischen Erstellung und Herstellung einer Bindung eine sensible Information, da diese in die Zertifikats- und Identifikatorgenerierung am Smartphone einfließt, und beim Initialen Verbindungsaufbau im Zuge des Attestierungsprozesses als Referenzwert dient. Der Abgleich dieses Werts sichert der Applikation am Desktop zu, dass sich tatsächlich ein autorisiertes Gerät verbindet.
Die Verwendung von Multicast-DNS zur Kommunikation von Identifikatoren zugehörigen IP-Adressen und Ports erfordert, dass beide Geräte Teil derselben Kollisionsdomäne sind. Zwar handelt es sich dabei um eine (künstliche) Einschränkung, allerdings entfallen im Rahmen eines Demonstrators auch etwaige durch IP-Routing und/oder *Network Address Translation* (NAT) bedingte Probleme.
Nach erfolgreicher wechselseitiger Bindung können vom Desktop aus Operationen am Smartphone veranlasst werden, deren Ergebnisse retourniert werden. Durch den Einsatz von Attestation stehen somit Trusted-Computing und Trusted IO am Desktop zur Verfügung. Nachfolgend wird dies an Hand zweier Anwendungsfälle exemplarisch veranschaulicht.

3.2 Biometrische Authentifizierung

Am Desktop bzw. speziell auf portablen Desktop-Computing-Geräten wie Notebooks oder Tablets, welche auf die Benutzung von Desktop-Betriebssysteme wie Microsoft Windows, Apple macOS, oder Linux ausgerichtet sind, finden sich zwar seit einigen Jahren biometrische Sensoren zur Authentifizierung, allerdings gibt es dafür keine einheitlichen Verfahren, welche eine applikationsunabhängige, verifizierbare Authentifizierungsmöglichkeit anbieten.
Darauf aufbauend, dass die Attestierung eines Android-Smartphones als Trusted-Computing-Plattform im Kern auf der Verifikation einer X.509-Zertifikatskette basiert, ergibt sich enorme Flexibilität, auch im Bereich Authentifizierung. Darüber hinaus müssen biometrische Sensoren unter Android direkt an die kryptografische Hardware, welche für die Schlüsselverwaltung und die Ausführung kryptografischer Operationen zuständig ist, angebunden sein. Das Betriebssystem bietet entsprechende Programmierschnittstellen an, um die Nutzung von kryptografischem Material an biometrische Nutzerauthentifizierung zu knüpfen. Dies spiegelt sich auch im kryptografischen Material wider: Bereits bei der Erstellung eines an biometrische Authentifizierung gebundenen asymmetrischen Schlüsselpaars, muss dieser Umstand definiert werden, da im Zuge der Schlüsselgenerierung auch ein Zertifikat erstellt wird, das die Notwendigkeit ebensolcher Authentifizierung in Form einer Zertifikatserweiterung (Extension) beinhaltet. Biometrische Authentifizierung ist in Folge der Erstellung einer kryptografischen Signatur gleichzusetzen. Konkret wurde dies im Rahmen dieses Projekts wie folgt umgesetzt:

1. Der Desktop sendet eine Authentifizierungsanfrage, welche eine kryptografische Nonce beinhaltet, an das Smartphone. Die Nonce wird lokal am Desktop zwischengespeichert.
2. Am Smartphone wird ein neues Schlüsselpaar generiert, welches biometrische Authentifizierung zur Nutzung des privaten Schlüssels erfordert. Zusätzlich fließt die im Rahmen der Enrolment-Prozedur generierte Nonce in die Schlüsselgenerierung ein, um ein

Zertifikat zum Zweck von Attestierung zu erhalten. Die Schlüsselgenerierung entfällt im Rahmen zukünftiger Authentifizierungsvorgänge.
3. Am Smartphone wird dem Benutzer bzw. der Benutzerin ein Dialog angezeigt, der zur biometrischen Authentifizierung auffordert.
4. Nach erfolgter biometrischer Authentifizierung wird das Schlüsselmaterial für einen Singnaturvorgang freigegeben, um die in Schritt 1 übermittelte Nonce zu signieren.
5. Das in Schritt 2 generierte Zertifikat wird mit dem zum selbstzertifizierenden Identifikator des Smartphones gehörigem privaten Schlüssel signiert.
6. Die signierte Nonce und das signierte Zertifikat (inklusive vollständiger Zertifikatskette) wird an den Desktop übertragen.
7. Am Desktop wird die Signatur des übertragenen Zertifikats überprüft, sowie eine Attestierung auf Basis der Zertifikatskette durchgeführt.
8. Anschließend wird überprüft, ob das Vertrauensniveau der empfangenen Zertifikatskette zumindest dem Vertrauensniveau der dem Smartphone-Identifikator zugehörigen Zertifikatskette entspricht.
9. Abschließend wird überprüft, ob die empfangene Nonce-Signatur zur lokal gespeicherten Nonce passt und valide ist.

Schlägt nur einer der Überprüfungsschritte fehl, wird die Authentifizierung als nicht erfolgreich angesehen. Durch die Abbildung der Authentifizierung auf kryptografische Verfahren kann dieses Konzept auch über die Verbindung von lediglich einem Desktop-Computer und einem Smartphone hinaus verwendet werden, da alle Verifizierungen offline, d.h. ohne Einbezug dritter oder des Geräts, das die Signatur erstellt hat, durchgeführt werden können. Ein Desktop-Client kann im Rahmen verteilter Applikation verwendet werden und Operationen, welche zwingend Nutzerinterkation verlangen (wie z.B. Authentifizierungsverfahren) können auf die Smartphone-Features zurückgreifen. Beispielsweise kann die zur Authentifizierung herangezogenen Nonce von außen kommen und das Resultat wieder nach außen weitergegeben werden. Die Bindung an die Desktop-Instanz kann ebenfalls über kryptografische Signaturen nach außen kommuniziert werden: Beispielsweise kann der Authentifizierungsinput vor der Übermittlung an das Smartphone von der Desktop-Instanz mit dem ihrem selbstzertifizierenden Identifikator zugehörigen privaten Schlüssel signiert werden und am Smartphone eine biometrisch ausgelöste Signatur nur durchgeführt werden, wenn diese Signatur zum am Smartphone gespeicherten Identifikator der Desktop-instanz passt. Die resultierende verschachtelte Signatur würde sowohl Bindung als auch biometrische Auslösung nach außen hin widerspiegeln.

Abgesehen von inhärent an kryptografische Primitive geknüpfte Vorgänge lassen sich jedoch beliebige Operationen an ein Smartphone, das sich als Trusted-Computing-Base qualifiziert, auslagern. Details werden nachfolgend an Hand eines Beispiels erläutert.

3.3 Auslagerung sensibler Operationen

Im Kontext dieses Projekts wird ein Android-Smartphone in seiner Gesamtheit als Trusted-Computing-Basis angesehen. Dadurch ist es möglich, beliebige Operationen auszulagern, und in einer vertrauenswürdigen Umgebung ausführen zu lassen.
Veranschaulicht wurde dies im Rahmen dieses Projekts an Hand eines beispielhaften *Remote Procedure Call* (RPC)-Protokolls auf Basis eines libp2p-Programierbeispiels, welches Addition und Multiplikation von Ganzzahlen unterstützt. Um auch in diesem Fall gegenüber Dritten eine nachweisliche Ausführung von Operationen innerhalb einer vertrauenswürdigen Umgebung zu erreichen, kann das zuvor skizzierte Co-Signaturkonzept wiederverwendet werden:

1. Operatoren und Operanden werden am Desktop signiert. Um Replay-Angriffen vorzubeugen, ist außerdem eine Nonce (ebenfalls signiert) zu übergeben.
2. Die signierten Eingangswerte der Operation werden inklusive Nonce an das Smartphone übermittelt.
3. Am Smartphone werden alle Werte erneut signiert.
4. Die Rechenoperation wird ausgeführt.

5. Das Ergebnis wird am Smartphone inklusive Nonce signiert, sodass diese ebenfalls Teil des signierten Ergebnisses ist, und eine nachverfolgbare Verbindung zwischen Eingangsdaten und Ergebnis entsteht.
6. Die signierten Daten werden an die Desktop-Instanz retourniert.

Alle kryptografischen Operationen verwenden die den selbstzertifizierenden Identifikatoren zu Grunde liegenden privaten Schlüssel, um so eine Bindung an diese zu erreichen. Folglich kann der Weg der Eingangsdaten bis in die vertrauenswürdige Umgebung, sowie ein damit assoziiertes Ergebnis auch von Dritten nachverfolgt und überprüft werden. Dieses Prinzip lässt sich auf beliebige Operationen erweitern, wobei im Falle der Erzeugung, bzw. Übermittlung von Streaming-Daten vom Smartphone an Dritte (wie z.B. einem kontinuierlichen Strom an GPS-Daten) eine nähere Betrachtung unter Berücksichtigung von Ende-zu-Ende-Verschlüsselung notwendig ist. Andernfalls würde sich ein unter realistischen Bedingungen potentiell zu hoher Overhead durch die hohe Anzahl zu erzeugender und zu prüfender Signaturen ergeben.

4 Fazit

Die Möglichkeit Smartphones als Trusted-Computing-Basis zu verwenden kann durch die weite Verbreitung und unkomplizierte Attestierung genutzt werden, um auch Desktop-Applikationen um Trusted-Computing-Features zu erweitern. Da es nicht notwendig ist, dafür Dritte zu kontaktieren, ergeben sich auch keine softwarearchitektonischen Einschränkungen in der Umsetzung derartiger Anwendungen. Somit ist die gesamte Bandbreite von Client-Server-Anwendungen, über verteilte (Peer-to-Peer-)Dienste, bis hin zu lokalen Applikationen, beispielsweise zu sicheren Verwaltung von Schlüsselmaterial realisierbar. Exemplarisch wurden in diesem Projekt biometrische Authentifizierung, sowie trusted RPC umgesetzt. Das Potential ist jedoch weit größer: Angefangen von Implementierungen eines Java-Security-Providers auf Basis der in Smartphones verbauten Hardware, sowie vertrauenswürdigen Sensordaten und generell Trusted IO lassen sich prinzipiell beliebige Arbeitsabläufe um Trusted-Computing-Fähigkeiten erweitern, ohne dass erst Geräte an die Nutzerbasis verteilt und ein Vertrauensverhältnis über elaborierte Enrolment-Verfahren hergestellt werden muss. Im Rahmen dieses Projekts setzen die umgesetzten Operationen stark auf kryptografische Signaturen. Im rein lokalen Kontext ist dies nach der Bindung jedoch nicht erforderlich, da eine ECDHE-basierte Ende-zu-Ende-verschlüsselte Verbindung zwischen Smartphone und Desktop besteht, welche ohnehin nur bei entsprechendem Vertrauensniveau des Smartphones aufgebaut wird. Aus diesem Grund gibt es keine technische Notwendigkeit für den Einsatz von Signaturen im engeren Sinn – schließlich besteht eine Verbindung zu einer Trusted-Computing-Basis, deren Ausgaben ohnehin als vertrauenswürdig anerkannt werden. Im Sinne eines Nachweises gegenüber Dritten (um auch potentielle verteile Anwendungen zu berücksichtigen) wurden jedoch Signaturen wie in den Abschnitt 3 beschrieben eingesetzt.
Weitere Entwicklungen auf Basis der Projektergebnisse bieten sich entsprechend im Rahmen verteilter Anwendungen, speziell im Kontext von Operationen an, welche kontinuierlich Daten liefern. In solchen Fällen ist eine Signatur einzelner Datenpakete potentiell problematisch, da dies zu unerwünschtem Overhead führen kann. Des Weiteren besteht ohne weitere Maßnahmen speziell im Fall von eventuell vom Smartphone bezogenen GPS-Daten im Kontext verteilter Anwendungen die Problematik von Relay-Attacken. Grund dafür ist, dass keine räumliche Nähe zwischen Smartphone- und Desktop-Instanz erzwungen wird. Hierfür ist die aktuelle Umsetzung, welche voraussetzt, dass beide Geräte Teil derselben Kollisionsdomäne sind, nicht ausreichend.

Referenzen

[1] S. Weiser, Enclave Security and Address-based Side channels, Graz: Technische Universität Graz, 2020.

[2] B. Prünster, „Erhebung State-of-the-Art Remote Attestation," A-SIT, 17 06 2019. [Online]. Available: https://technology.a-sit.at/erhebung-state-of-the-art-remote-attestation/. [Zugriff am 01 09 2020].

[3] R. Mayrhofer, J. Vander Stoep, C. Brubaker und N. Kralevich, „The Android Platform Security Model," 11 04 2019. [Online]. Available: http://arxiv.org/abs/1904.05572. [Zugriff am 11 06 2019].

[4] J. Iyengar und M. Thomson, „QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, 10 09 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-quic-transport-30. [Zugriff am 30 09 2020].

[5] I. Fette und A. Melnikov, „The WebSocket Protocol," Internet Engineering Task Force, 12 2011. [Online]. Available: https://tools.ietf.org/html/rfc6455. [Zugriff am 15 09 2020].

[6] T. Perrin, „The Noise Protocol Framework," 11 07 2018. [Online]. Available: http://www.noiseprotocol.org/noise.pdf. [Zugriff am 01 09 2020].