yubico

JULI 2020

YubiHSM 2:

Ein sicherer Schlüssel-speicher für die Fertigungsindustrie

Kurzbeschreibung

Als ein Ergebnis der Zusammenarbeit mit etlichen Partnern aus dem Fertigungssektor, die elektronische Bauteile in großen Volumen herstellen, hat Yubico jetzt eine Standardlösung zum Schutz von Betriebsgeheimnissen entwickelt, in deren Mittelpunkt das YubiHSM 2 steht. In einer globalisierten Welt wird die Integrität der Lieferkette immer wichtiger. Deshalb wenden sich produzierende Unternehmen an Yubico, um ihre Lieferkette und ihr geistiges Eigentum zu schützen. Das mögliche Einsatzspektrum reicht dabei von Sensoren und medizinischen Geräten bis hin zu Automobil- und anderen Produkten mit elektronischen Komponenten.

In allen Fällen bildet das YubiHSM 2 das Kernstück einer Lösung, die entweder den Herstellungsprozess schützt, indem sie dafür sorgt, dass nur zertifizierte Programmierstationen an die Komponenten angebunden werden können, oder dazu dient, digitale Signaturen auf jede Komponente zu schreiben, um die Authentizität sicherzustellen. In beiden Szenarien tragen die zusätzlichen Sicherheitsfunktionen des YubiHSM 2 dazu bei, die Reputation des Unternehmens zu wahren und zu gewährleisten, dass die Produkte erwartungsgemäß funktionieren, wenn sie die Produktion verlassen.

Dieses Dokument beschreibt zwei Ansätze, die heute in der Fertigungsindustrie gang und gäbe sind. Der erste basiert auf Joint Test Action Group (JTAG) und einer elektronischen Steuereinheit (ECU) und wird häufig in der Automobilindustrie angewandt. Der zweite Ansatz beruht auf X.509-Zertifikaten oder CVCs (Card Verifiable Certificate) und ist üblich bei der Herstellung von kleineren elektronischen Bauteilen, Smartcards und Geräten mit kontaktlosen Schnittstellen (etwa IoT-Devices). Nachfolgend wird sich zeigen, dass die Teilprozesse der Schlüsselverwaltung, des Signierens und Verifizierens bei beiden Ansätzen häufig vergleichbar sind, während der Output unterschiedlich ist. Dieser Vorschlag zielt somit darauf ab, den Output zu verallgemeinern und so zu einem einheitlichen Gesamtdesign zu gelangen, das keine spezifischen Formate vorgibt. Hervorzuheben ist dabei, dass sich dieser generelle Ansatz neben den beschriebenen Szenarien auch auf jeden anderen Produktionsvorgang übertragen lässt und auf jeden Prozess anwendbar ist, bei dem Geheimnisse und die Authentizität elektronischer Komponenten zu gewährleisten sind.

Abb. 1 – High-Level-Übersicht über die vorgeschlagene Lösung

Die obige *Abb. 1* zeigt eine High-Level- Übersicht über die vorgeschlagene Lösung auf Basis des YubiHSM 2, die als Ersatz für eine vorhandene Lösung oder als ganz neuer Prozess in Fabriken und/oder Fertigungsstraßen implementiert werden könnte. Das Wichtigste dabei: Der kryptographische Schlüssel, der zum Signieren und/oder Zertifizieren von Komponenten verwendet wird, *verlässt niemals* die YubiHSM 2-Hardware. Dies gewährleistet ein hohes Maß an Sicherheit.

Einführung

Hintergrund

In der Fertigungsindustrie muss die Authentizität aller Komponenten gewährleistet werden, die an einem durchgängigen Prozess beteiligt sind. Dies schützt vor unerwünschten Nachbauten und Diebstahl, dient aber auch der Qualitätssicherung – schließlich sollen in einer Fertigungsstraße nur authentische Produkte verwendet werden, da das Ganze immer eine Summe aller Teile ist. Deshalb ist stets eine Lösung erforderlich, die die Integrität aller Komponenten und das damit verbundene geistige Eigentum der Hersteller in allen Phasen schützt, von der Fertigung und Montage bis hin zur Instandhaltung und Erneuerung.

Deshalb schlägt dieses Papier vor, dass Hersteller, die derzeit Lösungen einsetzen, bei denen sensible kryptographische Informationen in Software gespeichert werden, einen Umstieg auf eine Lösung mit dem YubiHSM 2 in Betracht ziehen. Eine solche Lösung verringert das Risiko, dass Master-Keys kompromittiert werden. Das ergibt sich unmittelbar daraus, dass sowohl die Speicherung der kryptographischen Informationen als auch die Ausführung der kryptographischen Operationen in Hardware realisiert werden. Dadurch wird der Export vertraulicher Informationen abgesichert und die Gefahr einer Preisgabe privater Schlüssel minimiert. Um dies an einem Beispiel zu veranschaulichen: Da vertrauliche Daten auf einem YubiHSM 2 *niemals* offengelegt werden, gibt es selbst dann keine offensichtlichen Angriffsvektoren, wenn es einem entfernten Angreifer gelingt, ein Netzwerk oder einen damit verbundenen Computer zu kompromittieren. Gelingt es dem Angreifer dagegen, sich vollen Zugriff auf eine entsprechende Software zu verschaffen, hätte er mindestens die Chance, den Speicher oder lokale Dateien auf mögliche Schwächen oder Muster hin zu analysieren.

Die gleiche Argumentation gilt auch, wenn Hersteller eine ganz neue Lösung implementieren möchten und die Vorteile softwarebasierter und hardwarebasierter Ansätze gegeneinander abwägen. Darüber hinaus ermöglicht das hardwarebasierte YubiHSM 2 dank seiner geringen Größe und seines niedrigen Preises Flexibilität im Fertigungsdesign. Die Sicherheitskontrollen können auf lokalerer Ebene erfolgen, was die Abhängigkeit von Internetverbindungen verringert und den Sicherheitsperimeter verkleinert.

Zweck dieses Dokuments

Zweck dieses Dokuments ist es, eine generische Lösung vorzuschlagen, die auf produzierendes Gewerbe aber auch andere Unternehmen zugeschnitten ist und das YubiHSM 2 als zentrales Element nutzt, um kryptographische Informationen in Hardware zu verarbeiten und zu verwalten.

Andere Branchen neben der Fertigung

Dieses Whitepaper wendet sich zwar primär an die verarbeitende Industrie. Dabei eignet sich das YubiHSM2 keineswegs nur für Anwendungsfälle in der Fertigung. Vielmehr kann *jede* Branche oder Firma, die ihr geistiges Eigentum oder ihre Marke mit hardwarebasierter Kryptographie schützen möchte, von der hier vorgeschlagenen Lösung profitieren. Das gilt für den öffentlichen Sektor genauso wie für das Gesundheitswesen, für Finanzdienstleister oder andere Branchen. Das Design selbst ist standardisiert, lässt sich leicht skalieren und so an die unterschiedlichsten Gegebenheiten anpassen, solange der Schutz von Unternehmensdaten das Ziel ist.

Abkürzungen in diesem Dokument

API	Application Programming Interface
CA	Certification Authority
CMC	Certificate Management over CMS
CMP	Certificate Management Protocol
CMS	Cryptographic Message Syntax
CNG	Cryptographic API Next Generation
CVC	Card Verifiable Certificate
DER	Distinguished Encoding Rules
EAC	Extended Access Control
ECDSA	Elliptic Curve Digital Signature Algorithm
ECU	Electronic Control Unit
HMAC	Hashed Message Authentication Code
HSM	Hardware Security Module
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IoT	Internet of Things
ITU-T	Telecommunications Union's Telecommunication Standardization Sector
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extensions
JTAG	Joint Test Action Group
KSP	Key Storage Provider
PBKDF2	Password-Based Key Derivation Function 2
PCB	Printed Circuit Board
PKCS #5	Public Key Cryptography Standard #5 (Password-Based Cryptography)
PKCS #10	Public Key Cryptography Standard #10 (Certification Request Syntax)
PKCS #11	Public Key Cryptography Standard #11 (Cryptographic Token Interface)
PKI	Public Key Infrastructure
RA	Registration Authority
RFID	Radio-frequency identification
RSA	Rivest Shamir Adleman
SCEP	Simple Certificate Enrollment Protocol
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SN	Serial Number
TLS	Transport Layer Security
TLV	Tag-Length-Value
TPM	Trusted Platform Module
TRNG	True Random Number Generator

Was spricht für das YubiHSM 2?

Der Bedarf an Telemetrie-Diensten, verteilten oder ausgelagerten Produktionsprozessen und Remote-Arbeitsumgebungen steigt heute in allen Branchen und Sektoren. Dies bedingt eine noch größere Nachfrage nach Vorkehrungen zum Schutz der unternehmenseigenen digitalen Assets, die durch diese Entwicklungen entstehen. Unternehmen, die nicht auf die heute erforderliche datensicherheit orientierte Arbeitsweise vorbereitet sind, wird es möglicherweise schwerfallen, den modernen Herausforderungen gerecht zu werden.

Während der YubiKey mit dem Ziel entwickelt wurde, durch überlegene Hardware die Identität und die Daten von Personen zu schützen, standen bei der Entwicklung des YubiHSM 2 speziell die Anforderungen von Unternehmen im Fokus. Mit dem YubiHSM 2 setzt Yubico seine Erfolgsgeschichte als Spezialist für hardwarebasierter, ebenso hochkarätiger wie bedienungsfreundlicher Sicherheitsmodule fort. Das YubiHSM 2 gibt Unternehmen ein hochentwickeltes, hardwarebasiertes Verschlüsselungsgerät an die Hand, das gegen das herkömmliche Hardware-Sicherheitsmodul (HSM) leicht bestehen kann. Wie nachfolgend ausgeführt, bietet das YubiHSM 2 im Vergleich zu traditionellen Lösungen eine Reihe von Vorteilen. Es wird auch bereits von mehreren Yubico-Partnern im Fertigungsbereich eingesetzt, um eine Vielzahl von Datensicherheitsproblemen zu lösen.

Die einzigartigen Vorteile

Die Implementierung einer Lösung, die nicht auf ein beliebiges HSM, sondern auf das YubiHSM 2 setzt, bringt eine Reihe einzigartiger Vorteile. Dazu zählen insbesondere die folgenden:

- Das YubiHSM 2 ist das *kostengünstigste* HSM auf dem Markt – sein Preis beträgt nur einen Bruchteil der Kosten von Konkurrenzprodukten.
- Etwa fingernagelgroß, hat das YubiHSM 2 einen der marktweit kleinsten Formfaktoren. Zudem ist es mit einem standardmäßigen USB-A-Anschluss ausgestattet und dadurch mit einem breiten Spektrum an Systemen kompatibel. Ein Hinweis: *Abhängig vom Auftragsvolumen* kann der Formfaktor auf Wunsch so angepasst werden, dass er jedem beliebigen YubiKey gleicht.
- Dank der beiden oben genannten Faktoren ist das YubiHSM 2 leicht und schnell an zahlreichen Standorten (z. B. Fabriken, Produktionsstraßen) einsetzbar – zu geringeren entsprechenden Kosten als ein einzelnes HSM anderer Hersteller.
- Alle kryptographischen Informationen können im YubiHSM 2 erzeugt und gespeichert werden. Und da die Schlüssel das sichere Element nie im Klartext verlassen, werden sie Außenstehenden niemals offengelegt.
- Das YubiHSM 2 Authentifizierungskennwort für die Schlüssel- und Geräteverwaltung ist von allen zugrunde liegenden kryptographischen Schlüsseln getrennt und unabhängig. Die Schlüssel und die von den privaten Schlüsseln abgeleiteten Hash-Werte bleiben also selbst dann sicher, wenn das Authentifizierungskennwort kompromittiert wird.
- Mit der Wrap-Funktion für den Export kann das Schlüsselmaterial von einem YubiHSM 2 AES-verschlüsselt exportiert werden. Dies ermöglicht identische oder Backup-Implementierungen an mehreren Standorten.
- Das YubiHSM 2 unterstützt offene, bewährte Standards wie PKCS #11 und Microsoft CNG. Das verleiht ihm die nötige Flexibilität, um im Einklang mit den jeweiligen Anwendungsfällen in zahlreiche bestehende Umgebungen und Lösungen nachgerüstet zu werden.
- Das YubiHSM 2 unterstützt sowohl symmetrische Verschlüsselungssysteme wie AES und HMAC-SHA als auch asymmetrische Verfahren wie RSA und ECDSA. Dadurch eignet es sich für eine Vielzahl verschiedener Szenarien mit symmetrischen wie auch asymmetrischen Verfahren. In den folgenden Abschnitten zu JTAG und Zertifikaten wird dies näher beschrieben.

Anwendungsfall Nr. 1: Schutz von JTAG mit symmetrischen Verfahren

JTAG – kurz erklärt

JTAG ist eine gängige Hardwareschnittstelle, über die Computer und andere Geräte direkt mit den Pins und Chips kommunizieren können, die auf eine Leiterplatte (PCB) aufgelötet werden. Der JTAG-Standard wurde Mitte der 80er-Jahre von einem Konsortium entwickelt – der Joint Test Action Group –, um den zunehmenden Schwierigkeiten beim Testen integrierter Schaltungen zu begegnen, da die Leiterplatten immer komplexer wurden. Seither wird JTAG weithin verwendet. Mittlerweile hat sich der Einsatzbereich auf das Debuggen, Programmieren und Testen praktisch *aller* embedded Geräte erweitert.

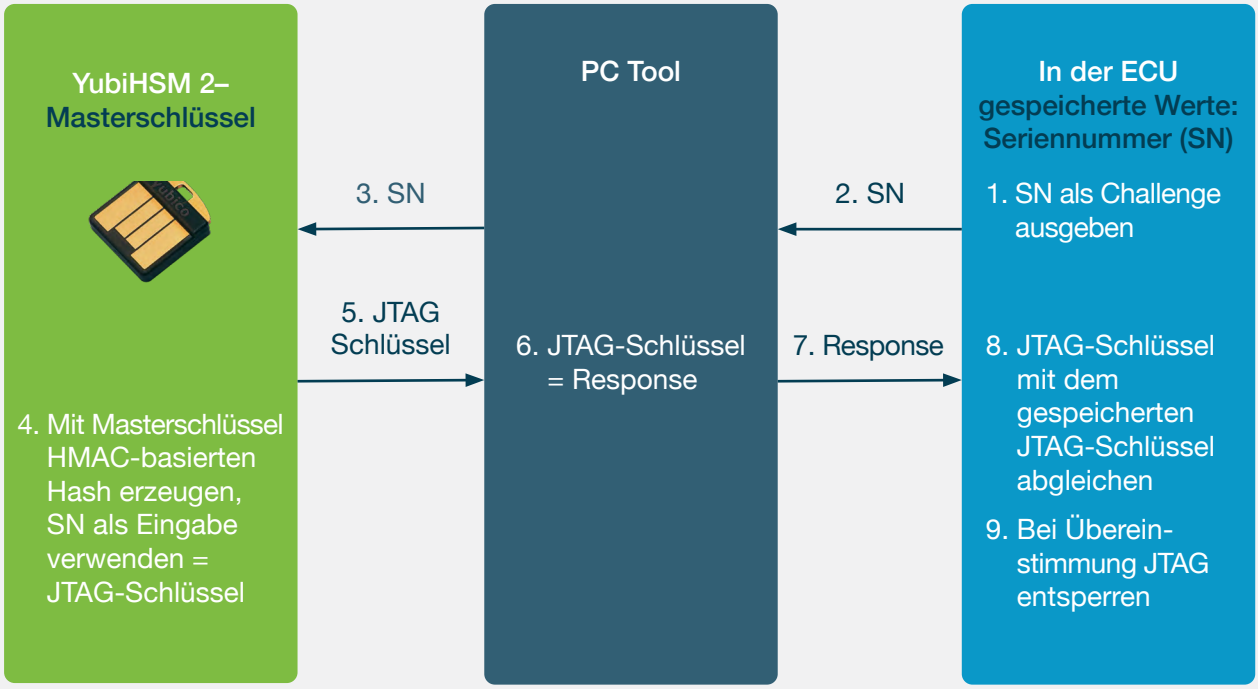
Es ist wichtig anzumerken, dass sich das „Debuggen“ einer Leiterplatte mit JTAG in der Elektrotechnik stark vom Debugging bei herkömmlicher Software unterscheidet. Bei JTAG bezieht sich der Begriff auf das Verfahren, mit dem sichergestellt wird, dass beispielsweise Pin A auf Chip A physisch mit Pin B auf Chip B verbunden ist und dass generell alle Pins korrekt funktionieren. Ein weiterer gebräuchlicher Begriff zur Beschreibung dieses Prozesses, der auch synonym zu JTAG verwendet wird, ist „Boundary Scan“.

JTAG-Verfahren sind für Chiphersteller in der Design-, Test- und sogar Produktionsphase außerordentlich nützlich. Gleichzeitig sind sie aber auch ein enormer potenzieller Angriffsvektor, da sie den direkten Zugriff auf die zugrunde liegenden Komponenten ermöglichen. Zum Glück sind sich die Hersteller dieses Risikos bewusst und ergreifen oft Maßnahmen, um den Zugriff auf JTAG-Schnittstellen zu verhindern. So werden etwa die JTAG-Leiterbahnen auf der Platine verborgen, komplett durchtrennt oder es wird sogar dafür gesorgt, dass während des Herstellungsverfahrens die Sicherungen in der JTAG-Verdrahtung durchbrennen. Diese Methoden bieten einen gewissen Schutz, wenngleich ein entschlossener Angreifer, der mit einem Lötkolben umzugehen weiß, den Schaden fast immer reparieren kann. Eine andere Option, die physisch weniger invasiv ist und verlässlicher sein dürfte, ist die Absicherung der Schnittstelle mit kryptographischen Mitteln. Dazu können beispielsweise Challenge-Response-Verfahren eingesetzt werden, doch selbst ein einfaches statisches Passwort reicht vielfach aus.

Statisches Passwort und Challenge-Response

Bei diesem Ansatz verschlüsselt oder hasht der Hersteller die Seriennummer jeder Leiterplatte, um einen eindeutigen *JTAG-Schlüssel* zu erzeugen, und stellt ihn dann während des Produktionsprozesses auf dem entsprechenden Steuergerät bereit. Der JTAG-Schlüssel dient einzig und allein dazu, die zugehörige JTAG-Schnittstelle zu schützen – er dient als eine Art Tor, das ein Benutzer vor dem Debuggen passieren muss. Wenn sich der Benutzer via JTAG verbinden will, stellt ihm das Steuergerät eine „Challenge“ auf Basis des JTAG-Schlüssels. Er muss dann eine „Response“ liefern, die *exakt* der erwarteten statischen Ausgabe oder der Ausgabe einer algorithmischen Berechnung entspricht, bei der die Challenge als Eingabe verwendet wurde. Die nachstehenden *Abb. 2* und *3* zeigen den Ablauf des Verfahrens mit einem statischen Passwort bzw. mit Challenge-Response, dargestellt in Schritten von 1 bis 9.

Abb. 2 – Statisches Passwort zur Autorisierung oder Entsperrung von JTAG



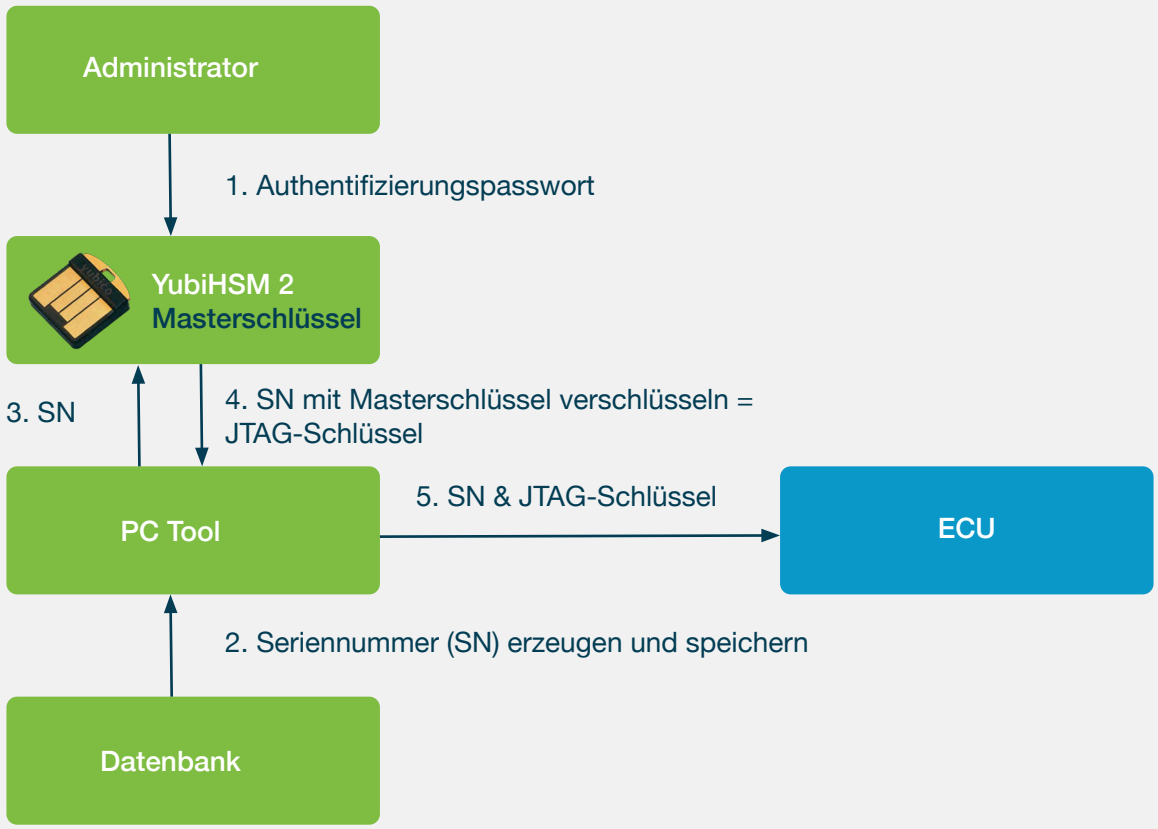
Challenge-Response statt statischem Passwort

Warum aber würde sich ein Hersteller überhaupt für das Challenge-Response-Verfahren entscheiden anstatt für eine statische Passwort-Implementierung? Einfach deshalb, weil das Verfahren sicherer ist, da sowohl die Challenge als auch die Response bei jedem neuen Versuch *dynamisch* erzeugt werden. Ein statisches Passwort ist dagegen anfällig für einen Replay-Angriff, sollte die Ausgabe abgefangen werden. Der offensichtliche Nachteil einer Challenge-Response-Authentifizierung ist allerdings, dass sie schwieriger oder kostspieliger sein kann, abhängig von den Tools und Prozessen, die der Hersteller zur Implementierung und Response verwendet.

Abb. 3 – Challenge-Response zur Autorisierung oder Entsperrung von JTAG

Derzeit müssen Hersteller, bevor sie ein symmetrisches Verfahren anwenden können, sowohl die Seriennummern als auch die JTAG-Schlüssel generieren, die ausgegebenen Seriennummern erfassen und schließlich die JTAG-Schlüssel auf den entsprechenden Steuergeräten bereitstellen. Demgegenüber zeigt *Abb. 4* (s. u.), wie Hersteller das Verfahren mit einem HMAC-basierten kryptographischen Algorithmus – oder einem YubiHSM 2 – realisieren könnten.

Abb. 4 – Erzeugen eines JTAG-Schlüssels und Schreiben auf ein Steuergerät



Erläuterung zu den einzelnen Elementen des Diagramms:

1. Das Authentifizierungspasswort des Administrators wird für den Zugriff auf den kryptographischen Algorithmus verwendet. Es *unterscheidet sich* vom zugrunde liegenden HMAC-Masterschlüssel (um eine Offenlegung des Schlüssels zu verhindern).
2. Jede erfolgreiche Erzeugung einer Seriennummer (SN) sollte in einer (sicheren) Datenbank aufgezeichnet werden, sowohl für Audits als auch zur künftigen Nachverfolgung.
3. Wahrscheinlich wird ein zwischengeschaltetes Tool benötigt, das auf einem angeschlossenen PC läuft, um zu steuern, wo die SN und JTAG-Schlüssel gesendet und empfangen werden.
4. Bei jedem Steuergerät wird die SN als „Salt“ für den Verschlüsselungsalgorithmus verwendet und der Masterschlüssel zur Verschlüsselung. Ausgegeben wird immer ein eindeutiger JTAG-Schlüssel.
5. Der JTAG-Schlüssel wird dann auf das Steuergerät geschrieben, zusammen mit der entsprechenden SN.

In Fällen, in denen dieses Verfahren bereits angewandt wird, jedoch mit irgendeiner Form von Software zur Ausführung des kryptographischen Algorithmus, sollte stattdessen ein YubiHSM 2 eingesetzt werden können, da die Ein- und Ausgaben im Allgemeinen standardisiert sind (nähere Informationen dazu weiter unten). Insgesamt ist der Lösungsvorschlag nicht nur einfach, sondern auch sicher und elegant. Belegt wird dies auch dadurch, dass die Lösung von Yubico-Kunden bereits erfolgreich implementiert wurde.

Anwendungsfall Nr. 2: Erstellen von Zertifikaten zur Bestätigung der Authentizität

Public-Key-Infrastruktur (PKI)

Bevor wir uns näher ansehen, wie ein YubiHSM 2 den Einsatz von Public-Key-Zertifikaten unterstützen könnte, müssen wir zunächst das Konzept einer *Root-CA* und ihre Rolle in einem Trust Framework verstehen, das auch als Public-Key-Infrastruktur (PKI) bezeichnet wird. Eine Root CA (auch Stammzertifizierungsstelle genannt) ist der *erste oder primäre Knoten,* der in einem künftigen Netzwerk aus Knoten eingerichtet wird, und gilt innerhalb einer PKI als die "Quelle der Wahrheit" oder auch "Vertrauensanker" (Trust Anchor). Zunächst kann nur die Root-CA selbst das Netzwerk erweitern, indem sie explizit neuen Knoten das Vertrauen ausspricht (oder, technisch ausgedrückt, jedes neue Zertifikat selbst signiert). Wenn das Netzwerk wächst, kann neuen Knoten jedoch auch durch Knoten vertraut werden, denen die Root-CA bereits vertraut hat. Ein Knoten auf der ersten Ebene zusätzlicher Knoten wird in der PKI oft als Zertifizierungsstelle (Certificate Authority, CA) bezeichnet. Als Proxy für die Root-CA können diese Knoten die Ausdehnung des Netzwerks exponentiell vergrößern. Logischerweise führt jede solche Ausweitung des Vertrauens zu einer „Vertrauenskette", die letztlich (und stets) bis zur Root-CA d.h. dem Vertrauensanker zurückverfolgt werden kann. Ein Beispiel: Wenn die Root-CA A dem Knoten B vertraut und der Knoten B anschließend Knoten C, entsteht aus Sicht einer externen Entität eine „Vertrauenskette" von A über B bis C. Und wenn A eine *vertrauenswürdige* Root-CA ist, lässt sich daraus schließen, dass die Knoten B und C ebenfalls vertrauenswürdig sind.

Das Konzept einer PKI ergibt sich aus den Grundsätzen der *asymmetrischen Kryptographie.* Danach sind die Root-CA, jede CA sowie alle anderen Knoten im alleinigen Besitz ihres eigenen privaten Schlüssels und kommunizieren mit anderen Knoten nur über Nachrichten, die mit ihrem öffentlichen Schlüssel entschlüsselt werden können. In einer sicheren PKI können die privaten Schlüssel aufgrund ihrer Länge und Komplexität nicht erraten, repliziert oder mit Brute-Force-Methoden gebrochen werden. Daher beweisen sie ohne jeden Zweifel, dass der *Inhaber* derjenige ist, für den er sich ausgibt. Aufgrund der mathematischen Eigenschaften der asymmetrischen Kryptographie (deren Erläuterung den Rahmen dieses Dokuments sprengen würde) kann jedes Zertifikat, das mit dem privaten Schlüssel einer Root-CA signiert wurde, nur mit dem entsprechenden öffentlichen Schlüssel verifiziert werden und umgekehrt. Mit anderen Worten: Wenn sich eine verschlüsselte Nachricht mit einem öffentlichen Schlüssel entschlüsseln lässt, ist die einzige logische Schlussfolgerung, dass nur der Inhaber des entsprechenden privaten Schlüssels die ursprüngliche Nachricht geschrieben oder erstellt haben kann.

Der Zweck einer CA ist letztlich, Public-Key-*Zertifikate* auszustellen. Das sind im Wesentlichen strukturierte und vordefinierte öffentliche Dateien, die Angaben zum Inhaber und natürlich den öffentlichen Schlüssel selbst enthalten. In diesem Whitepaper werden zwei Arten von Zertifikaten beschrieben: X.509-Zertifikate und Card Verifiable Certificates (CVC).

Die Registrierungsstelle (RA) schließlich ist ein Knotenpunkt in der PKI, der Zertifikatsanträge prüft und an eine geeignete CA im Netzwerk weiterleitet. Zudem ist die die RA noch für weitere Funktionen im Lebenszyklus von Zertifikaten zuständig, wie etwa Widerrufe.

X.509-Zertifikate

Ein X.509-Zertifikat enthält Informationen zur Identität des Zertifikatsinhabers, zu seinem öffentlichen Schlüssel und zur ausstellenden CA. Die ersten X.509-Zertifikate wurden als Teil des X.500 Directory Services Standards der ITU-T (International Telecommunications Union's Telecommunication Standardization Sector) ausgestellt. Später entwickelte die PKIX-Arbeitsgruppe der IETF ein Internet-Profil des X.509-Zertifikats.

Aus technischer Sicht ist ein X.509-Zertifikat gemäß DER (Distinguished Encoding Rules) codiert. Darüber hinaus hat die PKIX-Arbeitsgruppe mehrere weitere Formate und Protokolle zur Ausstellung, dem Widerruf und dem Lebenszyklusmanagement von X.509-Zertifikaten spezifiziert. Diese zu behandeln würde jedoch den Rahmen dieses Dokuments sprengen. Heute sind Milliarden von ausgestellten X.509-Zertifikaten in Gebrauch. Sie werden vorwiegend für TLS (Transport Layer Security) über das Internet, elektronische Signaturen, Verschlüsselung und verschiedene andere Sicherheitsprotokolle verwendet.

Card Verifiable Certificates (CVC)

CVCs sind digitale Zertifikate, die speziell auf die Verarbeitung durch Geräte mit begrenzter Rechenleistung ausgelegt sind. Sie sind häufig in Geräte oder Objekte wie Kreditkarten, Smart IoT-Geräte, ePässe o.ä. eingebettet. Die nötigen Informationen werden als „Felder“ in das CVC geschrieben: etwa eine eindeutige Kennung (oder Seriennummer), produktspezifische Details, Informationen zum Hersteller und, wie nachstehend näher erläutert, Informationen zur Sicherheit oder Zertifizierung.

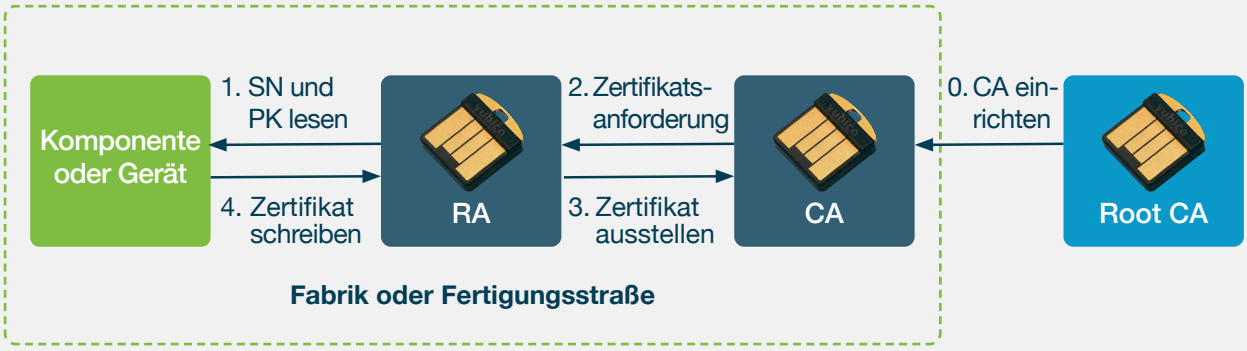
Aus technischer Sicht verwenden CVCs ein Format, das als TLV (Tag-Length-Value)-Kodierung bezeichnet wird. Dabei hat jedes Feld im zugrunde liegenden Zertifikat eine festgelegte Länge (falls Werte die maximale Länge nicht erreichen, werden sie aufgefüllt) und erscheint in einer vordefinierten Reihenfolge. Dadurch lassen sich die Werte sehr leicht parsen, anders als bei anderen Formen von variablen Eingaben, die vielleicht mehr Verarbeitungsleistung oder Speicherregister benötigen, um die Feldwerte vor der Verarbeitung zwischenzuspeichern.

Das YubiHSM 2, die PKI und der Ausstellungsprozess

Ein YubiHSM 2 kann eingesetzt werden, um eine PKI und ihr Netzwerk aus kryptographischen Schlüsseln zu schützen, also die Schlüssel der Root-CA sowie aller CAs und RAs.

Die nachstehende *Abb. 5* veranschaulicht den Ausstellungsprozess von Zertifikaten mit einer PKI auf Basis des YubiHSM 2, etwa in einer Fabrik, einer Produktionsstraße oder einem anderen Einsatzort.

Abb. 5 – Überblick über die PKI und den Ausstellungsprozess für ein Zertifikat



Beim abgebildeten Design verwendet die Root-CA ein YubiHSM 2, um ihr Schlüsselpaar (d. h. den privaten und den öffentlichen Schlüssel) sowie das zugehörige Zertifikat zu speichern. Dank der geringen Größe des YubiHSM 2 kann sich der Root-CA-Server praktisch überall befinden – wenngleich die Root-CA traditionell in einem externen Rechenzentrum gehostet wird, aus Sicherheitsgründen, aber auch aufgrund der enormen physischen Größe von Serverfarmen und typischen HSMs

In einer Fabrik oder Fertigungsstraße kann eine CA mit einem Zertifikat installiert und konfiguriert werden, das die Root-CA signiert hat. Dieser Prozess wird in *Abb. 5* als Schritt 0 bezeichnet und wird in der Regel offline und nur einmal durchgeführt, wenn das Werk zum ersten Mal ans Netz geht. Das Schlüsselpaar und das Zertifikat der CA werden ebenfalls durch ein YubiHSM 2 geschützt, ebenso wie die Berechtigungsnachweise der nachgeschalteten RA.

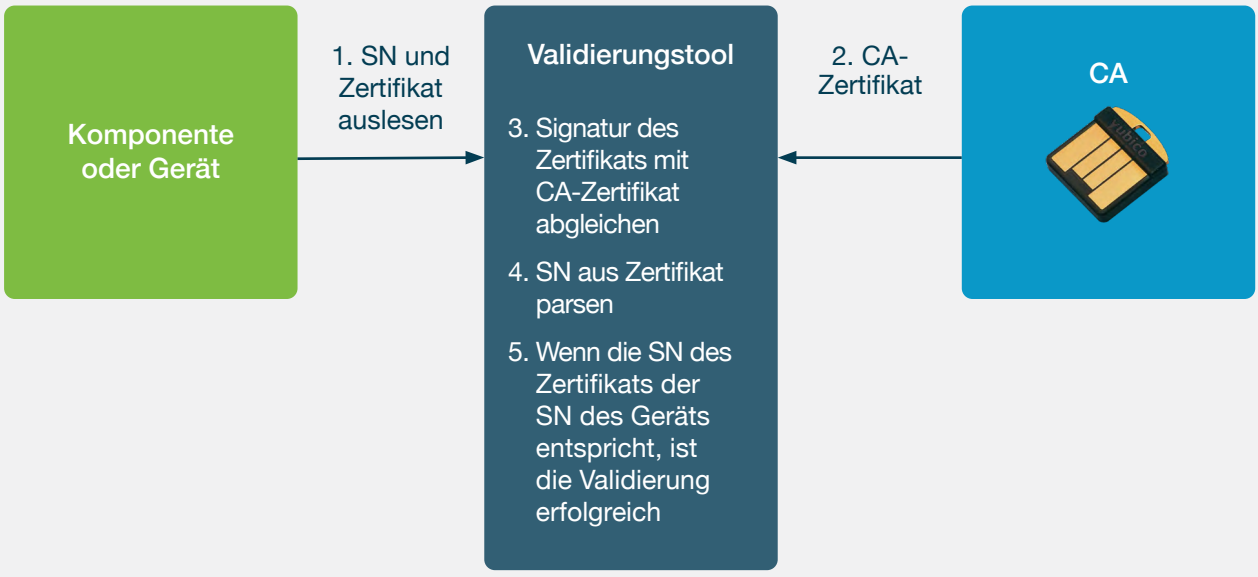
Der Ausstellungsprozess von Zertifikaten sieht folgendermaßen aus:

1. Die RA liest die Seriennummer (SN) oder eine gleichwertige eindeutige Kennung aus dem Bauteil oder Gerät aus, das gefertigt wird. Wenn das Gerät ein TPM enthält, das ein asymmetrisches Schlüsselpaar erzeugen kann, wird auch der öffentliche Schlüssel ausgelesen.
2. Die RA erstellt eine Zertifikatsanforderung (z. B. im PKCS#10-Format), die die Seriennummer und den öffentlichen Schlüssel der Komponente oder des Geräts enthält. Ein Hinweis: Falls zuvor kein öffentlicher Schlüssel ausgelesen wurde, wird jetzt einer für die Komponente oder das Gerät erzeugt. Die RA verwendet ebenfalls ein YubiHSM 2, um die Zertifikatsanforderung mit ihrem privaten Schlüssel zu signieren. Dann leitet sie das Zertifikat an die CA weiter (über ein Protokoll wie CMC, CMP oder SCEP).
3. Die CA signiert das Zertifikat mit ihrem privaten Schlüssel, der auf dem HSM liegt. Das ausgestellte Zertifikat wird an die RA gesandt.
4. Die RA schreibt das Zertifikat auf das Gerät, wo es (gegebenenfalls) auch mit dem privaten Schlüssel im TPM verknüpft wird. Wenn das Gerät über kein TPM verfügt, wird das Zertifikat zusammen mit dem privaten Schlüssel auf die Komponente oder das Gerät geschrieben.

Validierung eines Zertifikats zur Feststellung der Authentizität

Um die Authentizität eines hergestellten Geräts zu verifizieren, muss das Lesegerät prüfen, ob das Zertifikat einer Komponente oder eines Geräts gültig ist. Die nachstehende *Abb. 6* gibt einen Überblick über den Validierungsprozess.

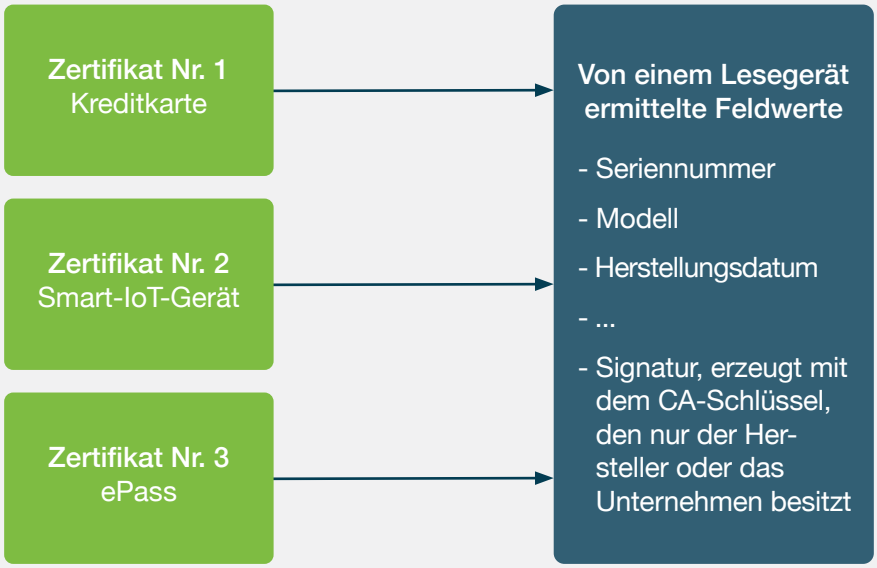
Abb. 6 – Ablauf der Zertifikatsvalidierung



Hier einige Erläuterungen zu den fünf Schritten:

1. Das Validierungstool liest die Seriennummer (SN) und das Zertifikat aus dem Gerät aus, das überprüft werden soll
2. Anhand der CA-Informationen im Geräte-Zertifikat wird das entsprechende CA-Zertifikat direkt von der ausstellenden CA heruntergeladen. Das Zertifikat der ausstellenden CA kann seinerseits mithilfe des öffentlichen Schlüssels der Root-CA überprüft werden, um die Authentizität der ausstellenden CA sicherzustellen. In einigen Fällen kann das validierte CA-Zertifikat vom Validierungstool zwischengespeichert werden, um künftige Prüfungen zu beschleunigen.
3. Die Signatur des Zertifikats wird geprüft, indem sie mit dem öffentlichen Schlüssel der CA entschlüsselt und mit dem im Zertifikat gespeicherten Fingerabdruck verglichen wird.
4. Die Seriennummer (oder eine ähnliche eindeutige Kennung) wird aus dem Zertifikat gelesen. Dieser auch Parsing genannte Prozess verläuft bei allen Geräten gleich, wie nachstehend in *Abb. 7* dargestellt.
5. Wenn die Seriennummer, die aus dem gültigen Zertifikat geparst wird, mit der Seriennummer des Bauteils oder Geräts übereinstimmt, kann dieses folglich als authentisch angesehen werden.

Abb. 7 – Feldwerte aus Zertifikaten auslesen und einen Echtheitsnachweis erhalten



Dynamisches Challenge-Response-Validierungsprotokoll

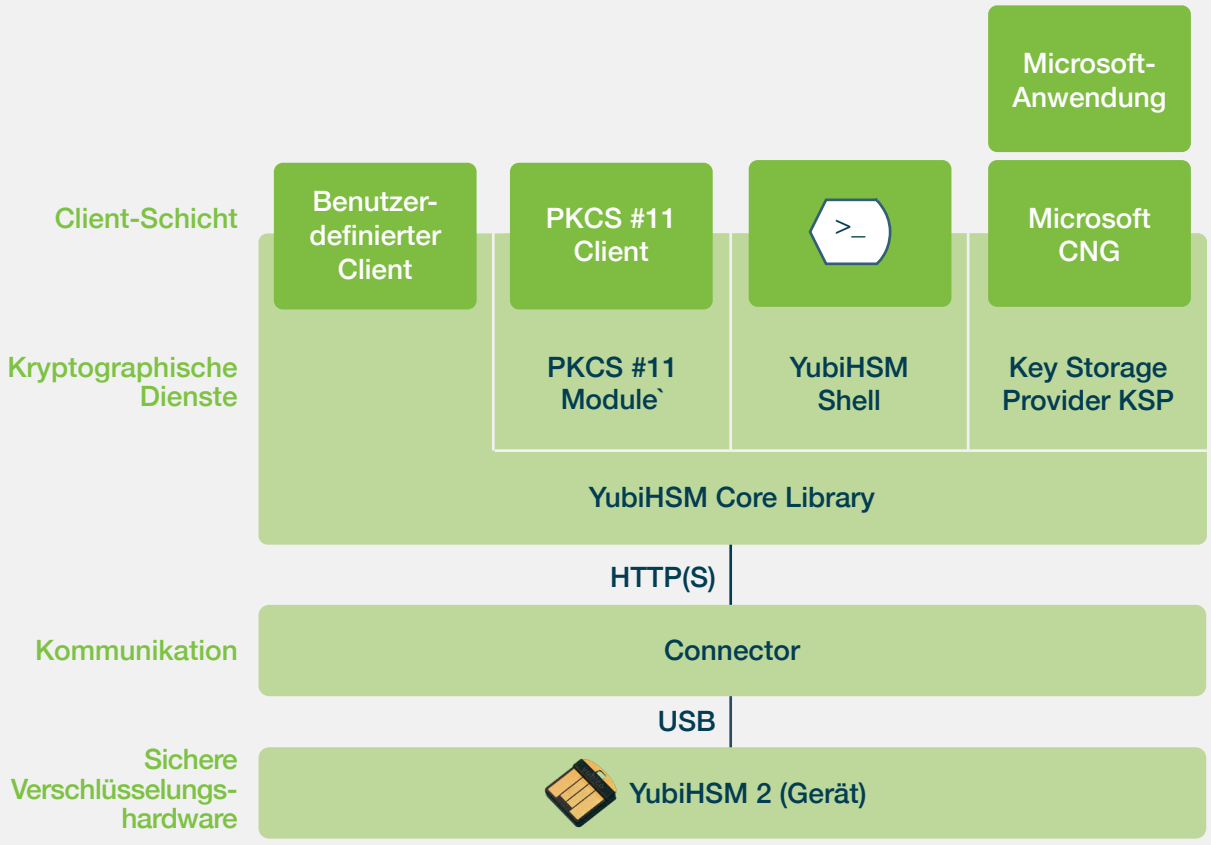
Wenn die Komponente oder das Gerät auch ein TPM unterstützt, das den privaten Schlüssel des Geräts enthält, kann ein dynamisches Challenge-Response-Protokoll entworfen werden, ähnlich wie in dem Beispiel, das beim JTAG-Anwendungsfall besprochen wurde. In diesem Szenario erzeugt das Validierungstool eine zufällige Challenge, die mit dem privaten Schlüssel des Geräts im TPM signiert wird. Das Validierungstool kann dann mithilfe des öffentlichen Schlüssels des Zertifikats die signierte Challenge überprüfen, bevor weiteren Schritte der oben beschriebenen Zertifikatsvalidierung ausgeführt werden.

Kommunikationsstack von der Hardware zur Software

Der YubiHSM2-Kommunikationsstack ist in *Abb. 8* (s. u.) dargestellt, um zu veranschaulichen, wie die Client-Schicht mit dem physischen Gerät verbunden ist. Hier eine kurze Zusammenfassung (von unten nach oben):

1. Ein YubiHSM2 wird physisch in einen USB-A-Anschluss eingesteckt, der seinerseits mit einem Computer und/oder Netzwerk (über ein USB2LAN Device)verbunden ist.
2. Das Betriebssystem des verbundenen Computers bzw. Netzwerks kommuniziert via HTTP mit dem HSM, und zwar über eine von Yubico bereitgestellte Binärdatei, die als Connector (oder *yubihsm-connector*) bezeichnet wird und sich ähnlich wie ein Treiber verhält.
3. Die für die kryptographischen Dienste zuständige Schicht, genauer gesagt, die YubiHSM Core Library (oder *libyubihsm*), stellt eine API zur Verfügung, die Nachrichten zwischen der im Stack über ihr angesiedelten Client-Ebene und dem Connector „übersetzt“, der im Stack direkt unter ihr liegt.
4. Die Client-Schicht schließlich umfasst eine Vielzahl von Methoden zur Steuerung und Verwaltung kryptographischer Funktionen. Dazu zählen etwa PKCS#11 (ein weit verbreiteter kryptographischer Standard für den Zugriff auf kryptografische Geräte); eine von Yubico bereitgestellte Befehlszeilenschnittstelle, die als YubiHSM-Shell (oder *yubihsmshell*) bezeichnet wird; die Microsoft Cryptography API: Next Generation (CNG) für Microsoft-spezifische Anwendungen; sowie eine beliebige Anzahl benutzerdefinierter Implementierungen.

Abb. 8 – Überblick über den YubiHSM 2-Kommunikationsstack



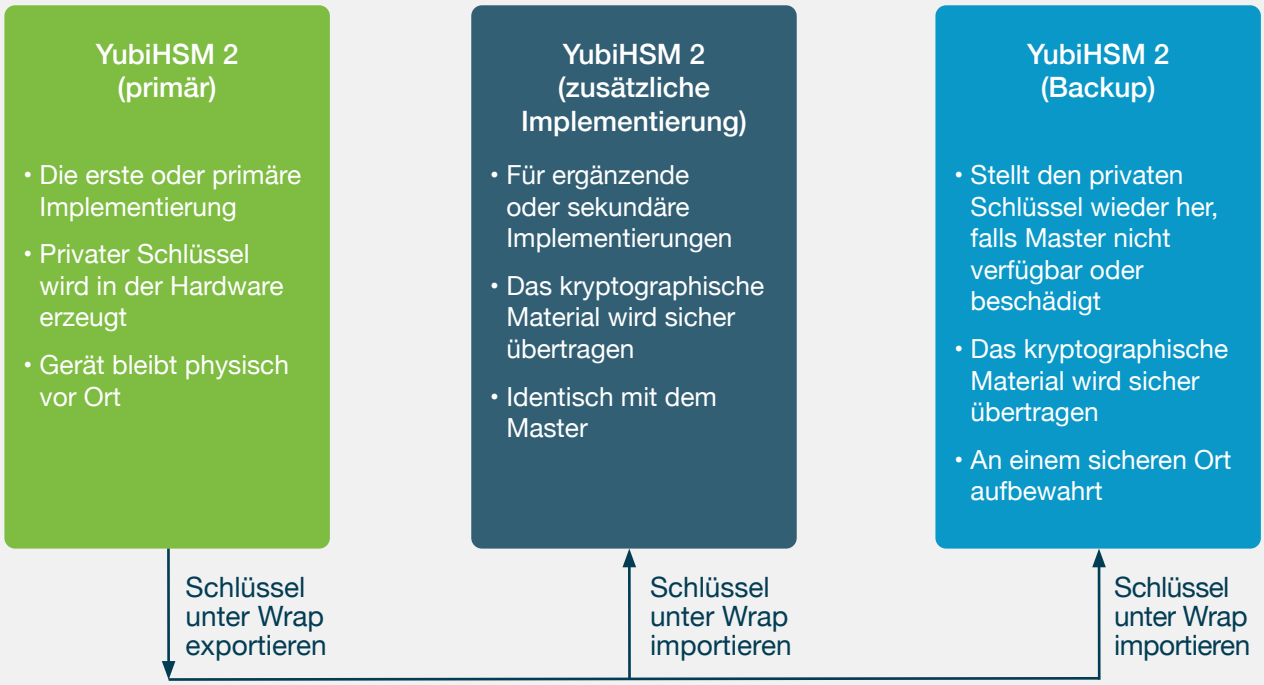
Weitere Informationen zur YubiHSM-Shell, PKCS #11 und der Microsoft CNG finden Sie weiter unten im Abschnitt „Integration mit dem YubiHSM 2“.

Replikation der Inhalte des YubiHSM 2

Alle privaten YubiHSM 2-Schlüssel können auf anderen YubiHSM 2-Geräten repliziert werden. Dazu wird zunächst das Schlüsselmaterial “under Wrap” vom Quellgerät exportiert und dann auf die Zielgeräte importiert. Vereinfacht ausgedrückt, wird dabei ein Schlüssel mithilfe eines anderen Schlüssels codiert (Key Wrapping), um ihn sicher speichern oder über einen *nicht vertrauenswürdigen* Kanal sicher übertragen zu können. Wobei im Zusammenhang mit dem YubiHSM 2 ausnahmslos jeder Ort *außerhalb* des sicheren Elements als nicht vertrauenswürdiger Kanal gilt.

Auf diesem Weg lässt sich ein YubiHSM2, in dem ein privater Schlüssel gespeichert ist, replizieren, um auf sichere Weise einen identischen Authentifizierungsschlüssel zu erstellen. Dies schafft eine schnelle, skalierbare und standortübergreifende Lösung, ohne dass Schlüsselmaterial offengelegt wird. *Abb. 9* (s. u.) zeigt dies im Überblick und fasst die Vorteile der Replikation zusammen. Hervorzuheben ist, dass das Authentifizierungspasswort für die einzelnen Geräte nicht gleich sein muss. Vielmehr kann (und sollte) es entsprechend den spezifischen Einsatzpräferenzen aktualisiert werden.

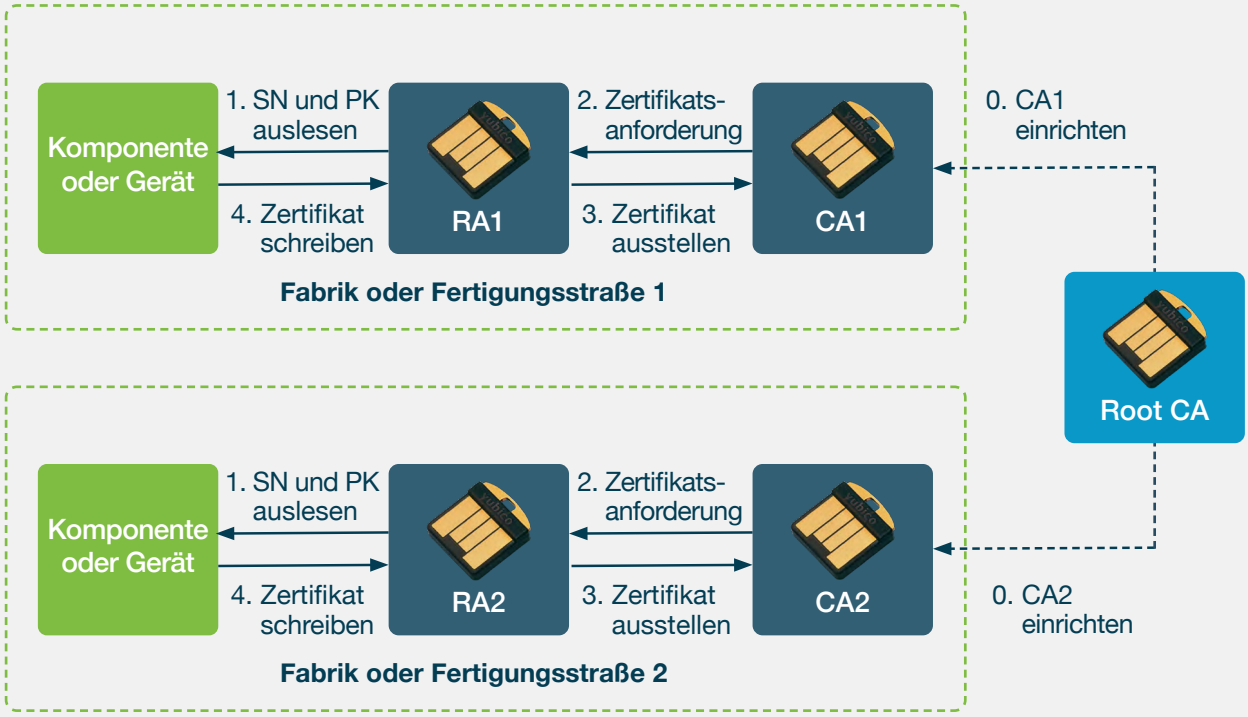
Abb. 9 – Überblick über die Bereitstellung an mehreren Standorten



Blicken wir noch einmal kurz auf den JTAG-Anwendungsfall zurück, der oben behandelt wurde. Dank des Konzepts der Replikation könnte ein YubiHSM 2-Gerät direkt in einer Fertigungsstraße eingesetzt werden, um verschlüsselte Seriennummern auf die Steuergeräte zu schreiben, während ein anderes Gerät an einen vertrauenswürdigen Kunden geliefert wird, um die entsprechenden JTAGs zu Testzwecken zu entsperren. Und ein weiteres Gerät könnte in einem Safe als Backup für den Fall aufbewahrt werden, dass das primäre Gerät verloren geht, gestohlen oder beschädigt wird.

Ähnliches gilt für den Anwendungsfall mit Zertifikaten. Mehrere Fabriken könnten mit einer replizierten Konfiguration – von der CA über die RA bis zum Bauteil/Gerät – ans Netz gebracht werden. *Abb. 10* (s. u.) erweitert das ursprüngliche Design aus *Abb. 5* durch Hinzufügen zusätzlicher Standorte. Hier werden zwar nur zwei dargestellt, doch in der Praxis gibt es keine zahlenmäßige Obergrenze.

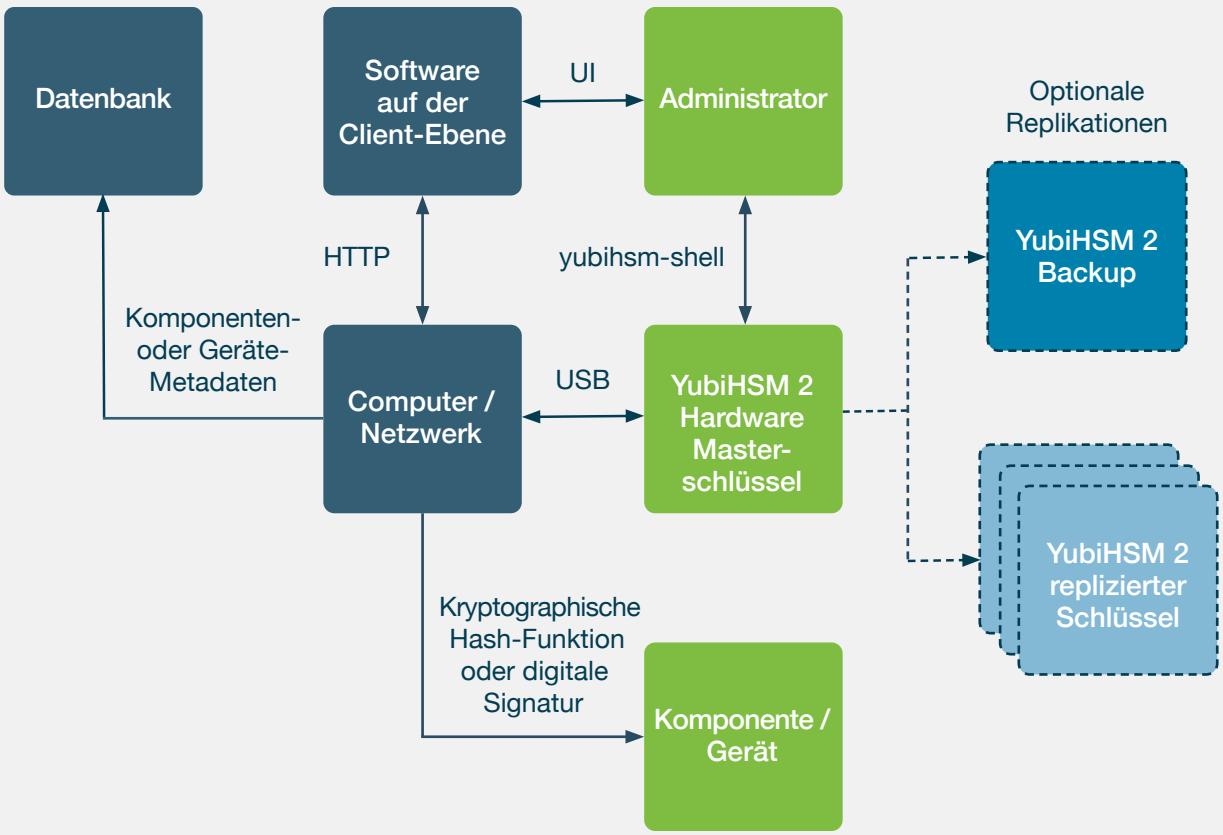
Abb. 10 – Erweiterung des ursprünglichen Verfahrens zur Zertifikatsausstellung



Das vorgeschlagene einheitliche Design mit dem YubiHSM 2

Wenn wir alle bisherigen Informationen zu den Anwendungsfällen, zum Kommunikationsstack und zur Möglichkeit der Replikation zusammenführen, können wir verallgemeinern, wie sich das YubiHSM 2 in den Fertigungsprozess eines Herstellers integrieren (oder nachträglich integrieren) lässt. Die nachstehende *Abb. 11* veranschaulicht die Zusammenführung aller Faktoren zu einem stimmigen, einheitlichen Designvorschlag.

Abb. 11 – Design der vorgeschlagenen einheitlichen Lösung



Zwar ist jeder Plan zum Ausrollen der Lösung und jede Fertigungssituation einzigartig und erfordert vielleicht nicht unbedingt alle in *Abb. 10* dargestellten Elemente. Das übergreifende Ziel sollte jedoch stets sein, alle privaten Schlüssel sicher in einem YubiHSM 2 zu speichern, um die Bedrohungen für den kryptographischen Teilprozess zu minimieren. Die Vorteile eines Designs mit dem YubiHSM 2 wurden bereits erläutert, und das Design lässt sich leicht skalieren, indem man einfach die Zahl der replizierten Geräte erhöht. So eignet sich diese Lösung für große Hersteller genauso wie für kleine und für lokale genauso wie für solche mit zahlreichen Standorten.

Integration mit dem YubiHSM

Für die Integration mit dem YubiHSM 2 stehen verschiedene APIs und Tools zur Verfügung:

- YubiHSM Shell
- PKCS #11
- Microsoft CNG
- Java-Bibliotheken
- Python

YubiHSM Shell

Die YubiHSM Shell ist ein Kommandozeilenwerkzeug, das zur Konfiguration des YubiHSM 2 und zur Interaktion mit den zugrunde liegenden kryptographischen Funktionen genutzt werden kann. Der typische Anwendungsfall für die YubiHSM Shell ist die Erstkonfiguration des YubiHSM 2 mit den Masterschlüsseln, aber auch die Replikation dieser Schlüssel auf mehreren YubiHSM 2-Geräten.

Zudem kann die YubiHSM Shell auch zur Erstellung von JTAG-Schlüsseln verwendet werden. Dies erfordert allerdings eine manuelle Interaktion durch einen Administrator und wird für umfangreiche Implementierungen nicht empfohlen. Zur programmatischen Integration mit dem YubiHSM 2 werden stattdessen die APIs PKCS#11 oder Microsoft CNG, Python oder die Java-Bibliotheken empfohlen.

PKCS #11

Eine potenziell bedienungsfreundlichere und zukunftsorientierte Lösung (im Vergleich zum Kommandozeilenbetrieb mit der YubiHSM Shell) ist die Integration von Anwendungen in das YubiHSM 2 mit der YubiHSM 2 PKCS#11 Library API auf der Client-Ebene. Solche Anwendungen bzw. Tools können den Benutzern dann ein grafisches Navigationsmenü oder Interface zur Verfügung stellen, das unter der Oberfläche YubiHSM 2-Operationen ausführt. Eine programmatische Integration mit der PKCS#11 API ermöglicht es zudem, JTAG-Schlüssel oder Zertifikate automatisiert zu erstellen. Die PKCS#11-Bibliothek bietet kryptografische Low-Level-Funktionen, die bei der Auflistung und Auswahl der Schlüssel hilfreich sind, die im YubiHSM 2 erzeugt werden.

Microsoft CNG

Wenn die fragliche Umgebung mit Microsoft Windows läuft, ist die Microsoft Cryptography API: Next Generation (CNG) eine praktikable Option. Der YubiHSM 2 Key Storage Provider (KSP) kann mit der CNG-Ebene verbunden und von jeder Anwendung aus aufgerufen werden, die auf Microsoft CNG aufsetzt. Microsoft CNG bietet mehr Funktionalitäten im Hinblick auf die Codierung kryptographischer Nachrichten und Zertifikate und könnte eine Alternative für die Zertifikatsausstellung sein. Für das Signieren von JTAG-Schlüsseln mit HMAC ist jedoch PKCS#11 die empfohlene Lösung.

Java-Bibliotheken

Für eine Java-Umgebung stehen bei der Integration mit dem YubiHSM 2 zwei Optionen zur Auswahl – der native JCE PKCS#11 Provider oder die YubiHSM 2 Java-Bibliothek.

Der JCE PKCS#11 Provider wird von Oracle (ehemals Sun) für die Java Cryptographic Architecture (JCA) implementiert und unterstützt. Der JCE Provider lädt die PKCS#11-Bibliothek über eine native Java-Schnittstelle und wird mit dem JCA-Framework verbunden. Mit dieser Architektur können Entwickler Java-Anwendungen auf die JCA aufsetzen und Zugriff auf die kryptographischen Schlüssel und Funktionen des YubiHSM 2 erhalten.

Als Alternative zur JCE/JCA-Architektur bietet Yubico die YubiHSM 2 Java-Bibliothek, die sich über HTTP(S) mit dem YubiHSM Connector verbindet. Die YubiHSM 2 Java-Bibliothek steht als GitHub-Projekt zur Verfügung. Es ist allerdings derzeit noch nicht offiziell als Yubico SDK freigegeben.

Python

Eine weitere Alternative schließlich ist die Integration mit dem YubiHSM 2 über die YubiHSM Python-Bibliothek. Die YubiHSM Python-Bibliothek ermöglicht es Entwicklern, über den Connector-Dienst eine Schnittstelle mit einem YubiHSM 2 herzustellen, wobei die Programmiersprache Python verwendet wird. Wenn Anwendungen in Python entwickelt werden, bietet sich die YubiHSM Python-Bibliothek natürlich als Option an.

yubico

Über Yubico

Yubico setzt neue globale Standards für den einfachen und sicheren Zugriff auf Computer, mobile Geräte, Server und Internetkonten.

Die zentrale Erfindung des Unternehmens, der YubiKey, bietet starken One-Touch-Hardwareschutz für eine beliebige Anzahl von IT-Systemen und Onlinediensten. Das YubiHSM, das ultraportable Hardware-Sicherheitsmodul von Yubico, schützt vertrauliche Daten, die auf Servern gespeichert sind.

Yubico leistet einen maßgeblichen Beitrag zu den offenen Authentifizierungsstandards FIDO2, WebAuthn und FIDO Universal 2nd Factor. Die Technologie des Unternehmens wird von 9 der 10 führenden Internetmarken und von Millionen von Benutzern in 160 Ländern verwendet und geschätzt.

Yubico wurde 2007 gegründet und befindet sich in Privatbesitz. Das Unternehmen unterhält Niederlassungen in Schweden, Großbritannien, Deutschland, den USA, Australien und Singapur. Weitere Informationen finden Sie auf www.yubico.com.