



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

SECURITY

Protokolle für Authentizität

June 7, 2023

Marc Stöttinger

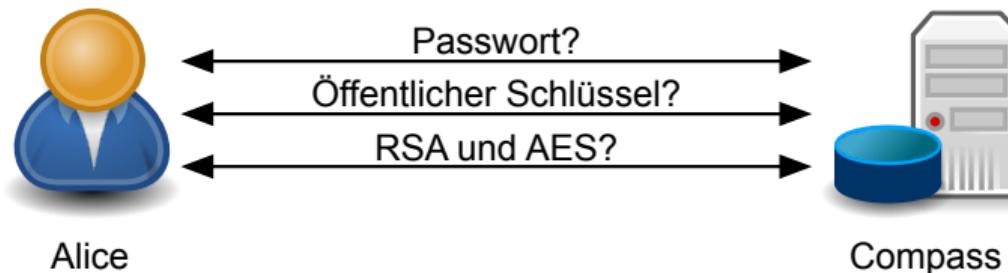


Wo das Wissen aufhört, beginnt der Glaube.

Augustinus Aurelius

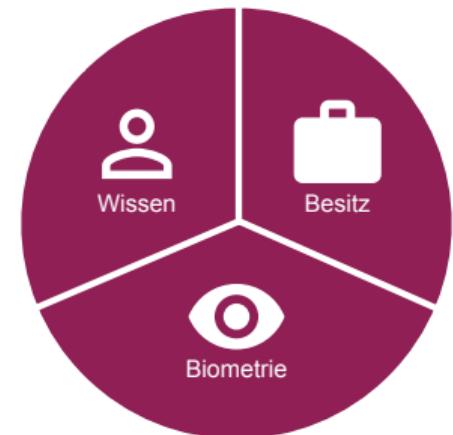
MOTIVATION

- **Bisher:** Kryptographische Verfahren ohne Anwendungskontext
 - Wie werden sie in der Praxis eingesetzt und was gibt es zu beachten?
- **Anwendungsbeispiel:** Alice möchte sich bei Compass einloggen
 - Wie kann sich Alice authentifizieren?
 - Wie kann sich der Server authentifizieren?



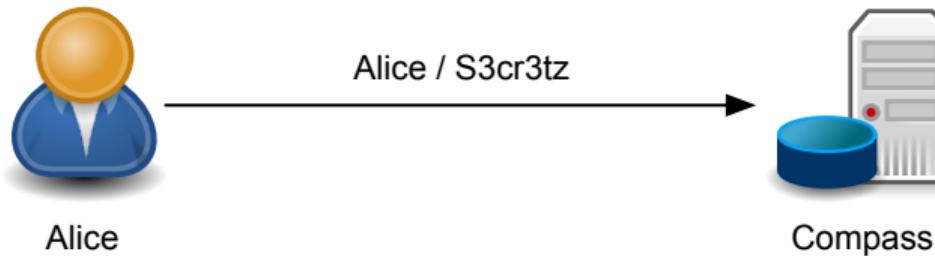
AUTHENTIFIKATIONSMERKMALE

Merkmal	Sicherheit basiert auf	Beispiel
Wissen	Nur Nutzer bekannt	Passwort, PIN, Sicherheitsfrage
Biometrie	Physiologisch oder verhaltenstypisch einzigartige und unkopierbare Merkmale einer Person	Fingerabdruck, Gesicht, Iris, Retina, Gang, Tastaturanschläge
Besitz	Im Besitz von Nutzer und Entwendung wird bemerkt	Chipkarte (z.B. SIM-Karte), Smartphone, USB Token, TAN Generatoren



PASSWORT

- Passwörter sind das häufigste Authentifizierungsverfahren im Internet
- **Nur Alice kennt** einen geheimen String, der nicht erraten werden kann. Probleme:
 1. **Nicht erraten**: Wie sieht ein gutes, nicht-erratbares Passwort aus?
 2. **Nur Alice kennt**: Das Passwort wird eingegeben, übertragen und auf dem Server gespeichert!
 3. **Sicher zu benutzen**: Das Passwort kann eventuell im Klartext übertragen werden!



User	Password
Alice	S3cr3tz
Bob	1233456
Eve	*Lauschi*
...	...

ERRATBARKEIT PASSWORT

- Menschen wählen Passwörter nicht sicher, sondern leicht merkbar
- Passwort Policies, um Menschen zu sicheren Passwörtern zu **erziehen**
- Jahrelanges Tauziehen zwischen Nutzer und Policies führte nicht zu sichereren Passwörtern

Beispiel Passwort	Angriffsstrategie	Passwort Policy Update
asdf	Zufällige Buchstabenkombination testen	Mindestens 8 Zeichen
passwort	Wörter aus Wörterbuch testen	Großbuchstaben müssen enthalten sein
PassWort	Buchstabenkombinationen klein und groß	Ziffern hinzufügen
PassWOrt21	Jahreszahlen anhängen und Gängige Substitutionen (e → 3)	Sonderzeichen hinzufügen
P\$ssWOrt21	Gängige Substitutionen (4 → \$)	Passwortupdates nach 90 Tagen
P\$ssWOrt22	Counter am Ende hochzählen	...

PASSWORT-TIPPS FÜR NUTZER

- **Kurz und komplex:** 8+ Zeichen aus Groß/Kleinschreibung, Ziffern und Sonderzeichen
 - Beispiel: Ein Passwort, das aus einem Satz abgeleitet wurde! → 1P,dae\$aw!W2
- **Lang und weniger komplex:** 20+ Zeichen aus Groß/Kleinschreibung
 - Beispiel: FischbrötchenKaufErinnerungMaerkteFrisch
- Nutzen Sie **einzigartige Passwörter** für wichtige Dienste
 - Z.B., eMail, Online Banking, Unternehmens-IT
 - Passwortmanager, um Passwörter zu generieren und zu speichern
- Regelmäßig **Passwort-Leaks** prüfen (z.B. [HIBP])
 - Für kontrollierte Domains ist eine Registrierung möglich
 - Passwörter können via HIBP API geprüft werden

GEGENMASSNAHMEN ZU BRUTE-FORCE ANGRIFFE

- Brute-Force Angriffe können Passwörter zwar raten, müssen diese aber auch prüfen
 1. Probe-hafter Login auf Webseite
 2. Abgreifen geheimer Daten auf dem Server
- Detektierende und reaktive Gegenmaßnahme **Probe-hafter Login**
 - Wartezeit zwischen Anmeldeversuchen als exponentiell wachsend konfigurieren
 - Blockieren einer IP nach X Versuchen
 - Sperrung eines Accounts für X Minuten
 - Logging der Versuche zur Erkennung eines Angriffs
- Preventive Gegenmaßnahmen zum **Abgreifen geheimer Daten** auf dem Server
 - Speichern in einer nicht-auslesbaren Form via Hashing

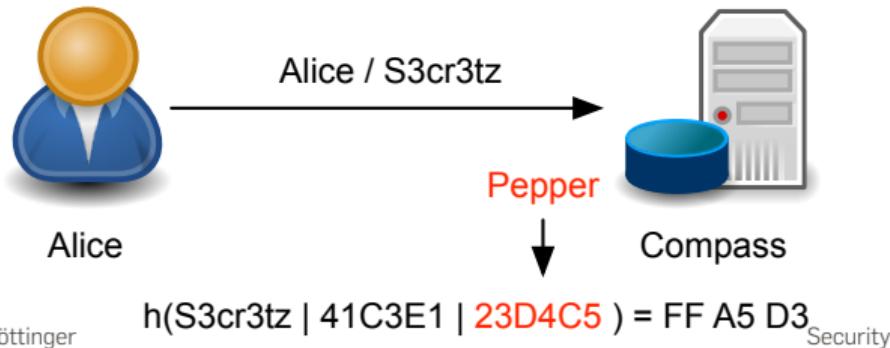
SICHERES SPEICHERN VON PASSWÖRTERN - SALT UND PEPPER HASHING

→ Passwörter werden mit zusätzlichen und zufälligen Daten gehashed

1. **Salt**: Individueller Zufallswert, der mit Passwort gespeichert wird
2. **Pepper**: Zufälliger und geheimer Wert, der für alle Passwörter konstant ist
3. Berechnung als $h(\text{Passwort} \parallel \text{Salt} \parallel \text{Pepper})$

→ Sicherheitsgewinn:

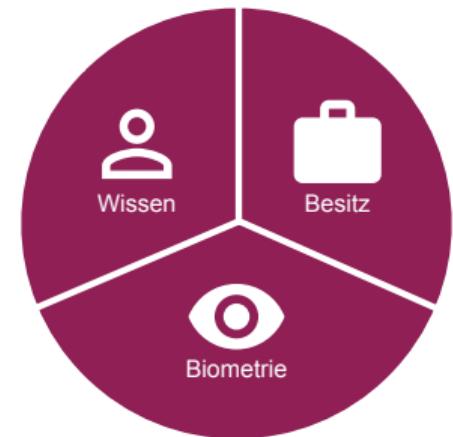
- **Salt**: Rainbow Tables verhindert, da für jedes Passwort ein eigener Salt gewählt wird
- **Pepper**: Brute-force erschwert, da Pepper geraten werden muss (solange Pepper unbekannt)



User	PW Hash	PW Salt
Alice	FF A5 D3	41C3F1
Bob	ED 12 4F	12D32E
Eve	1F 3E 38	2945F2
...

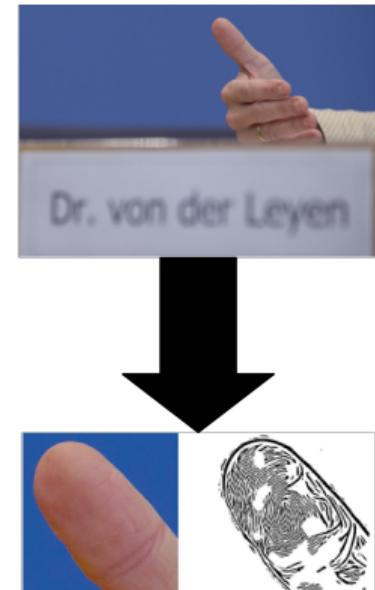
AUTHENTIFIKATIONSMERKMALE

Merkmal	Sicherheit basiert auf	Beispiel
Wissen	Nur Nutzer bekannt	Passwort, PIN, Sicherheitsfrage
Biometrie	Physiologisch oder verhaltenstypisch einzigartige und unkopierbare Merkmale einer Person	Fingerabdruck, Gesicht, Iris, Retina, Gang, Tastaturanschläge
Besitz	Im Besitz von Nutzer und Entwendung wird bemerkt	Chipkarte (z.B. SIM-Karte), Smartphone, USB Token, TAN Generatoren



BIOMETRISCHE IDENTIFIKATION

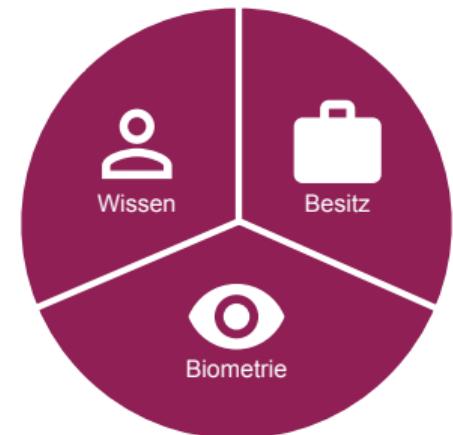
- Biometrische Merkmale sind zwar schwer von Menschen zu kopieren, aber
 - Öffentlich einsehbar und nicht sonderlich geschützt
 - Maschinell nachstellbar, wenn das Modell bekannt ist
 - Unsicher wenn Daten einmal veröffentlicht wurden
- Nutzbarkeit ist sehr gut
 - Biometrische Merkmale immer dabei
 - Intuitiv, da weite mediale Verbreitung
- Erkennung häufig fehlerhaft
 - Externe Einflüsse wirken störend (Licht, Kälte)
 - Merkmale ändern sich über die Zeit (Gesicht, Deutlichkeit des Fingerabdrucks)



Quelle: CCC2014

AUTHENTIFIKATIONSMERKMALE

Merkmal	Sicherheit basiert auf	Beispiel
Wissen	Nur Nutzer bekannt	Passwort, PIN, Sicherheitsfrage
Biometrie	Physiologisch oder verhaltenstypisch einzigartige und unkopierbare Merkmale einer Person	Fingerabdruck, Gesicht, Iris, Retina, Gang, Tastaturanschläge
Besitz	Im Besitz von Nutzer und Entwendung wird bemerkt	Chipkarte (z.B. SIM-Karte), Smartphone, USB Token, TAN Generatoren



AUTHENTIFIKATION VIA BESITZ

- Wissen und Biometrie können kopiert werden ohne, dass Nutzer es bemerkt
- **Lösung:** Physisches Objekt, das Nutzer ständig bei sich tragen kann und dessen Diebstahl bemerkt wird
 - Das Objekt sollte nicht einfach kopierbar sein
 - Geheimnisse auf dem Objekt dürfen nicht auslesbar sein
- Zur Authentifikation darf der Schlüssel den Chip nicht verlassen
 - Spezielle Protokolle benötigt



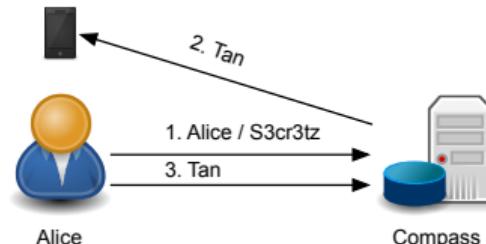
VOR- UND NACHTEILE DER AUTHENTIFIZIERUNGSTECHNIKEN

Merkmal	Vorteile	Nachteile
Wissen	<ul style="list-style-type: none"> → Einfach zu implementieren → theoretisch sicher → keine zusätzliche Technik 	<ul style="list-style-type: none"> → Sicherheit abhängig vom gewählten Passwort → schwer zu merken bei vielen Zugängen → Kopieren nicht bemerkbar
Biometrie	<ul style="list-style-type: none"> → Kein Transport oder Merken notwendig → Eindeutig pro Mensch 	<ul style="list-style-type: none"> → Physikalischer Scanner benötigt → Externe Faktoren können zu Fehlern führen → Kopieren manchmal nicht bemerkbar → Merkmal nicht wechselbar → Ein Leak reicht, um Sicherheit des Merkmals zu korrumpern → Kann auch gegen Nutzer verwendet werden
Besitz	<ul style="list-style-type: none"> → Kein Merken notwendig → Entwendung ist bemerkbar → Standardmäßig hohe Sicherheit 	<ul style="list-style-type: none"> → Ggf. physikalische Schnittstelle benötigt (Smart Card Reader) → Aufwändig in der Umsetzung → Muss von Nutzer mittransportiert werden

ZWEI- UND MEHRAKTOURAUTHENTIFIZIERUNG

- Mechanismen aus Wissen, Besitz und Biometrie können kombiniert werden, um mehr Sicherheit zu erreichen
 - **Zweifaktor Authentifizierung**: Kombination von zwei Mechanismen aus verschiedenen Kategorien
 - **Multifaktor Authentifizierung**: Kombination von mehr als zwei Mechanismen aus verschiedenen Kategorien

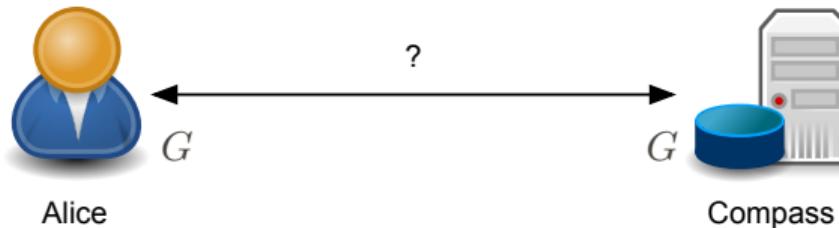
- **Beispiel: Online Überweisung**
 - Wissen (Passwort)
 - Besitz (SIM Karte / Smartphone)



DISKUSSION IN KLEINEN GRUPPEN

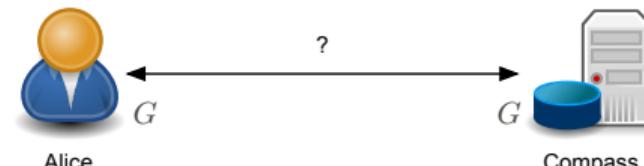
Authentifizieren ohne Senden von Geheimnissen

Alice's Smart Card und der Server kennen ein gemeinsames Geheimnis G (Passwort, Schlüssel, ...). Wie kann sich Alice's Smart Card gegenüber dem Server authentifizieren, ohne das Geheimnis G zu senden (Nehmen Sie an, dass der Angreifer alle Nachrichten lesen und wiedereinspielen kann.)?



CHALLENGE-RESPONSE PROTOKOLLE

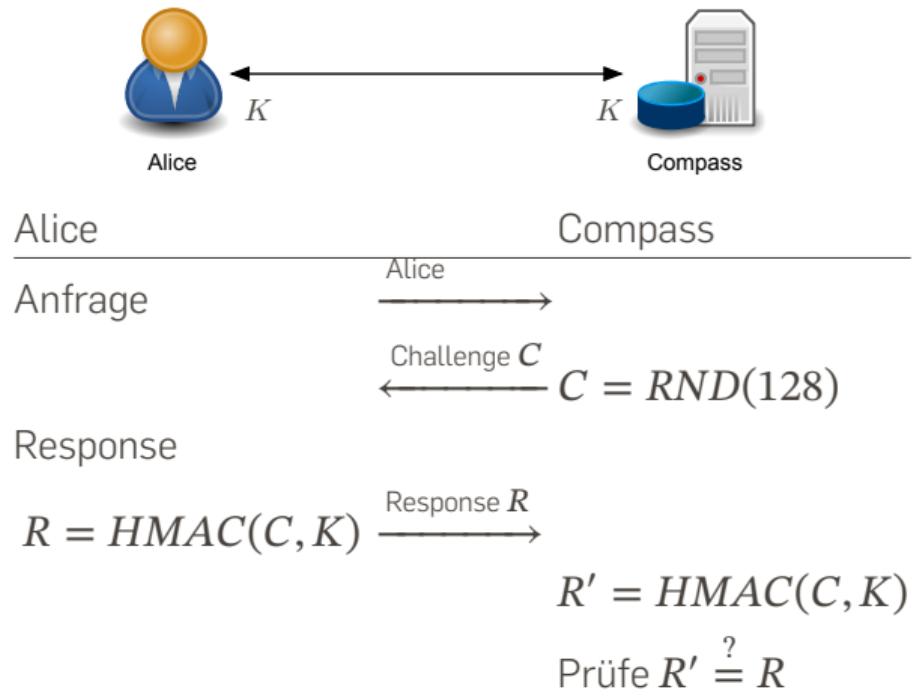
- **Initial:** Der Server und Alice's Smart Card kennen beide ein Geheimnis G
- Funktion f kann durch verschiedene kryptographische Verfahren instanziert werden (z.B.: (A)Symmetrische Verschlüsselung, MAC Funktion, ...)
- Nötige Eigenschaften von f :
 - Einwegeigenschaft
 - Schwache Kollisionsresistenz



Alice	Compass
Anfrage	→ zufällig generierte
	Challenge C
Response	← Challenge C
$R = f(C, G)$	→ Prüfe Response

CHALLENGE-RESPONSE PROTOKOLLE MIT HMAC [RFC2617]

- Geheimnis besteht aus symmetrischem Schlüssel K
- Funktion f ist HMAC-SHA1 (**HMAC**)
- Eigenschaften für f direkt gegeben durch MAC Verfahren
- Als Challenge wird eine 128-Bit Zufallszahl generiert
- Zur Verifikation: Server führt die gleiche Berechnung durch



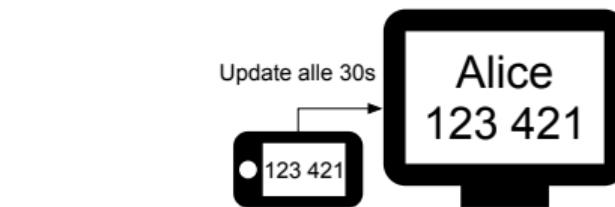
AUTHENTIFIKATION VIA BESITZ EINMALPASSWORT (OTP)

→ **Einmalpasswort (OTP)**: Automatisiert generierter und einmalig nutzbarer Verifikationscode

- HMAC-based OTP (HOTP): HMAC-SHA1 auf Zähler und Schlüssel [RFC4226]
- Time-based OTP (TOTP): HOTP mit Zeit statt Zähler [RFC6238]



1. Backend generiert Schlüssel K für Nutzer
2. Gerät liest und speichert Schlüssel K

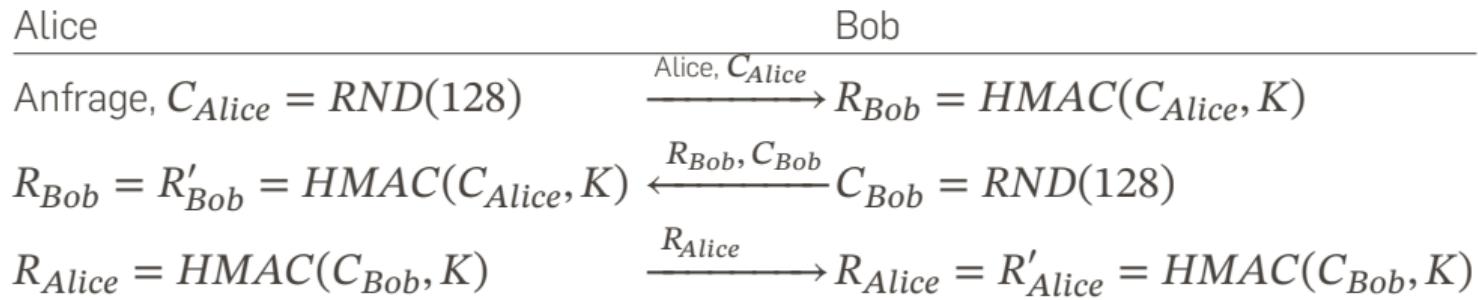


1. Gerät berechnet alle 30s $HMAC(K, Zeit)$
2. Backend verifiziert MAC und prüft Zeittoleranz

MUTUAL CHALLENGE-RESPONSE PROTOKOLLE



OPTIMIERTES MUTUAL CHALLENGE-RESPONSE PROTOKOLL?



KEINE EIGENEN MUTUAL CHALLENGE-RESPONSE PROTOKOLLE ENTWICKLEN

Mallory

Anfrage, $C_{Mallory} = RND(128)$ $R_{Bob} \stackrel{?}{=} R'_{Bob} = HMAC(C_{Alice}, ?)$ $R_{Alice} = HMAC(C_{Bob}, ?)$ R_{Bob} von Session 2

Bob

 $\xrightarrow{\text{Alice, } C_{Mallory}} R_{Bob} = HMAC(C_{Mallory}, K)$ $\xleftarrow{R_{Bob}, C_{Bob}} C_{Bob} = RND(128)$ $\xrightarrow{R_{Alice}} R_{Alice} \neq R'_{Alice} = HMAC(C_{Bob}, K)$ $\xrightarrow{R_{Bob}} R_{Bob} = R'_{Alice} = HMAC(C_{Bob}, K)$

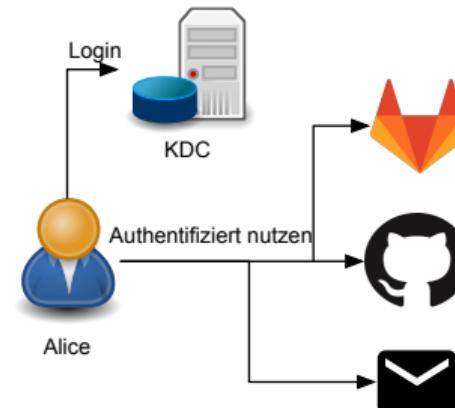
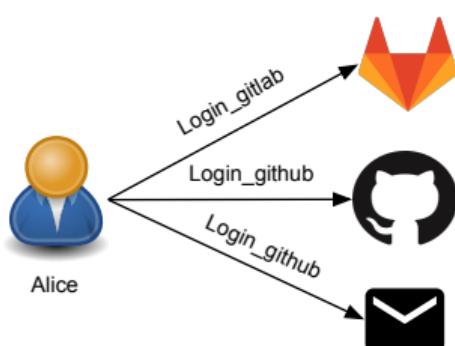
Session 2:

Anfrage

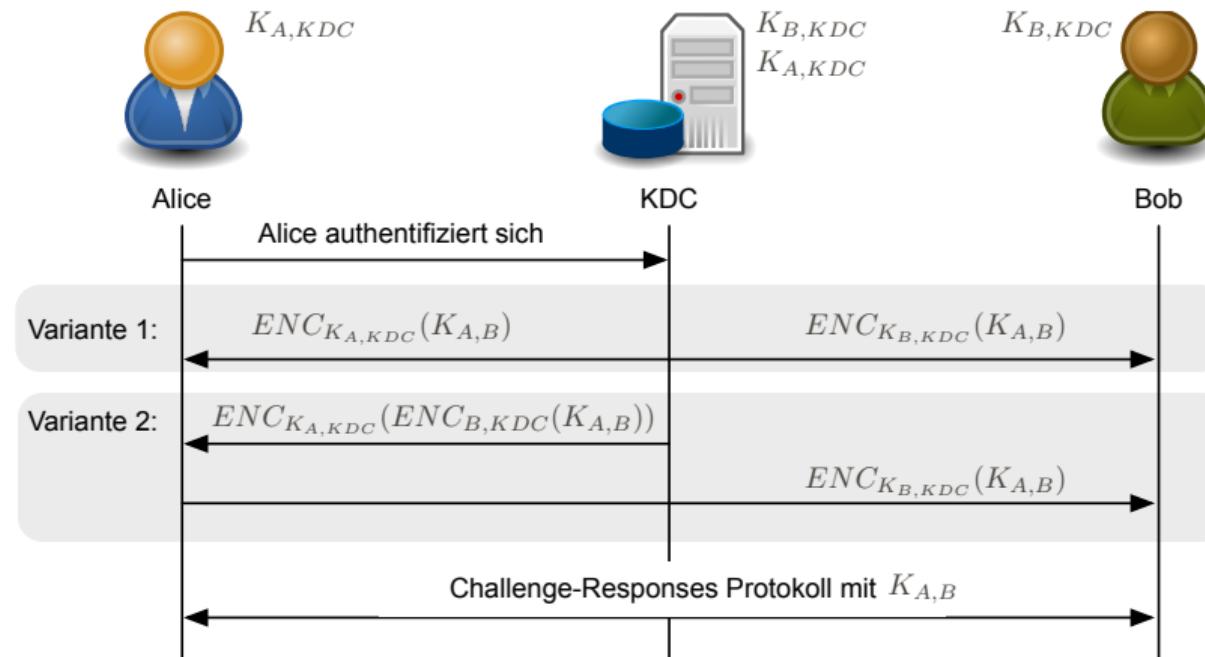
 R_{Bob} $\xrightarrow{\text{Alice, } C_{Bob}} R_{Bob} = HMAC(C_{Bob}, K)$ $\xleftarrow{R_{Bob}, C_{Bob}} C_{Bob} = RND(128)$

SINGLE SIGN-ON (SSO) SYSTEME

- Viele Systeme verlangen eine individuelle Authentifizierung
- Lösung: Ein Schlüsselverwaltungsservers (KDC) ermöglicht einen Single Sign-On (SSO) Service, die zentrale Authentifizierung ermöglichen
 - Unternehmensnetzwerke: Kerberos
 - Privat: OAuth/OpenID Systeme (Google, Facebook, ...)



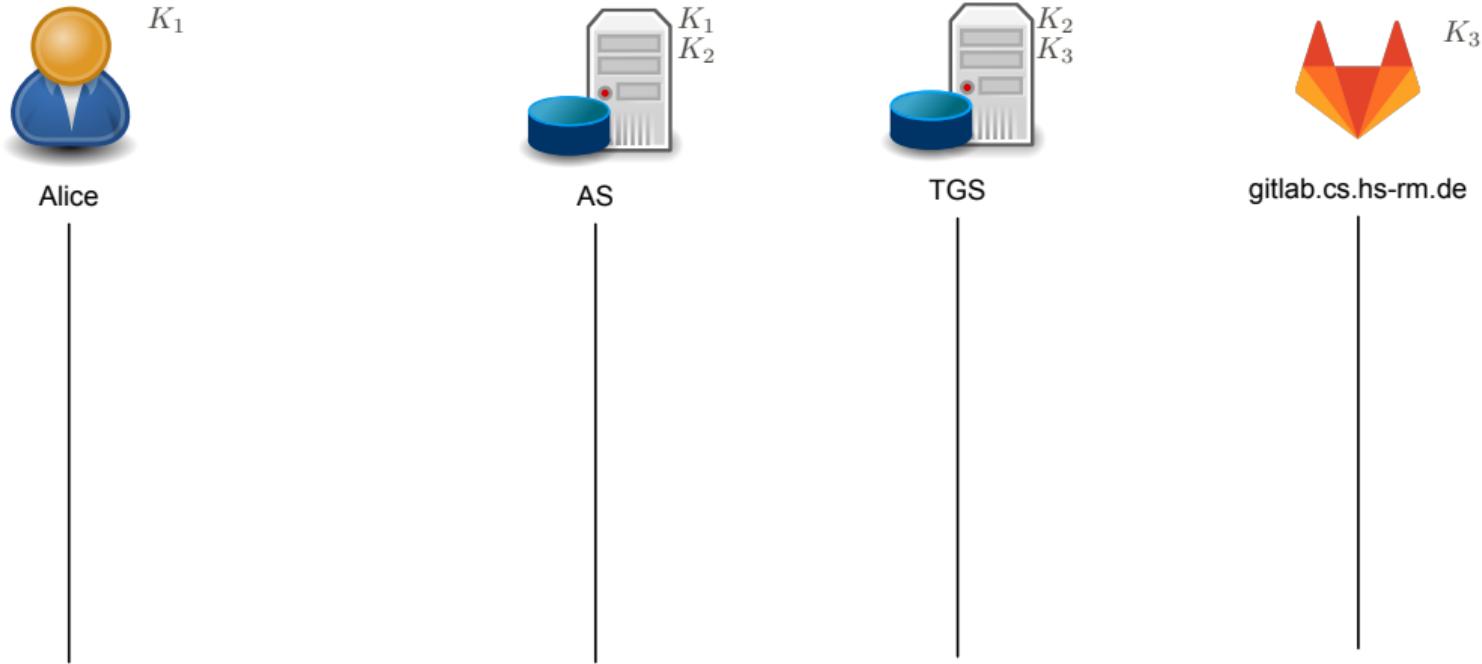
AUFBAU VON KEY DISTRIBUTION CENTERS FÜR AUTHENTIFIZIERUNG



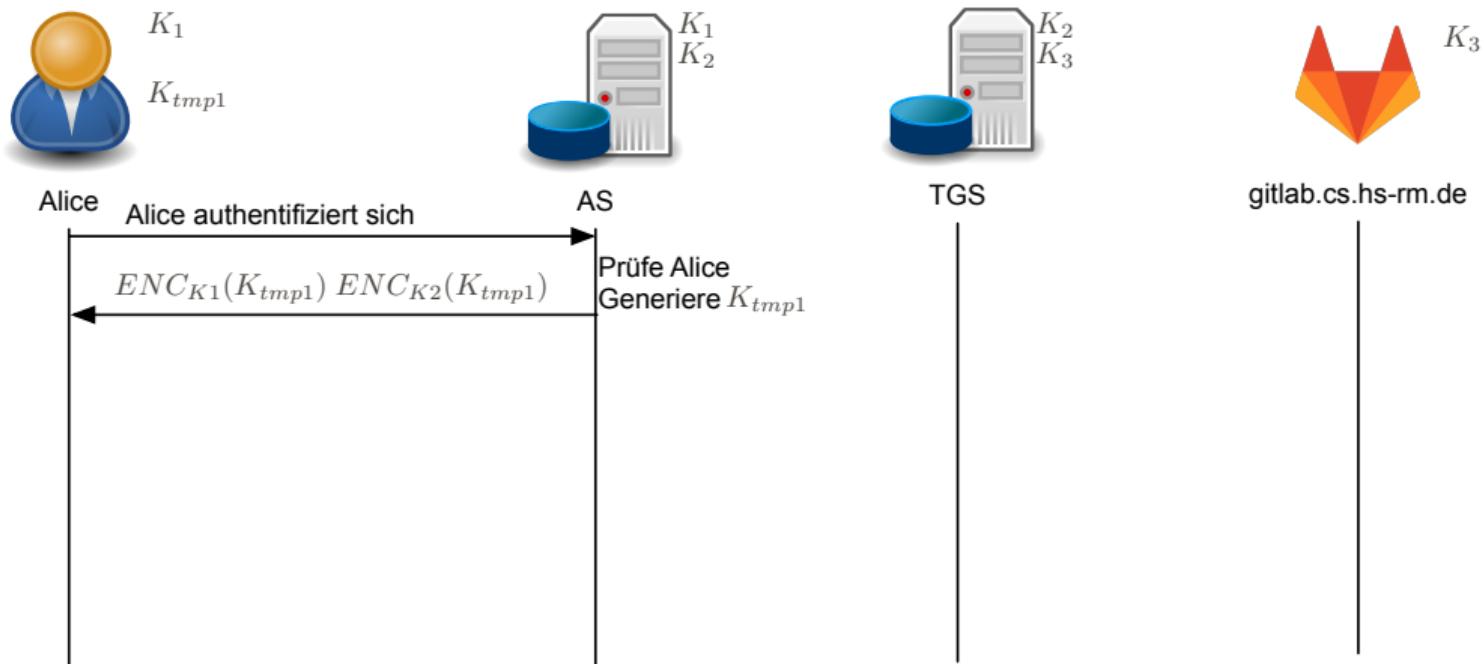
KERBEROS SINGLE SIGN-ON

- Kerberos wurde am MIT entwickelt und ist der de-facto Standard für SSO in Unternehmen
- Nutzer authentifiziert sich nur gegenüber Domain Controller und bekommt von diesem Tickets zur Nutzung von Services ausgestellt
- Der Kerberos Server übernimmt die Rolle des KDC
 - **Authentifizierungsserver (AS)**: Prüft Identität und stellt ein Ticket für eine Sitzung über eine gewisse Dauer aus
 - **Ticketgenehmigungsserver (TGS)**: Prüft, ob Nutzer-Sitzung aktiv ist und stellt Tickets zur Nutzung der Services aus

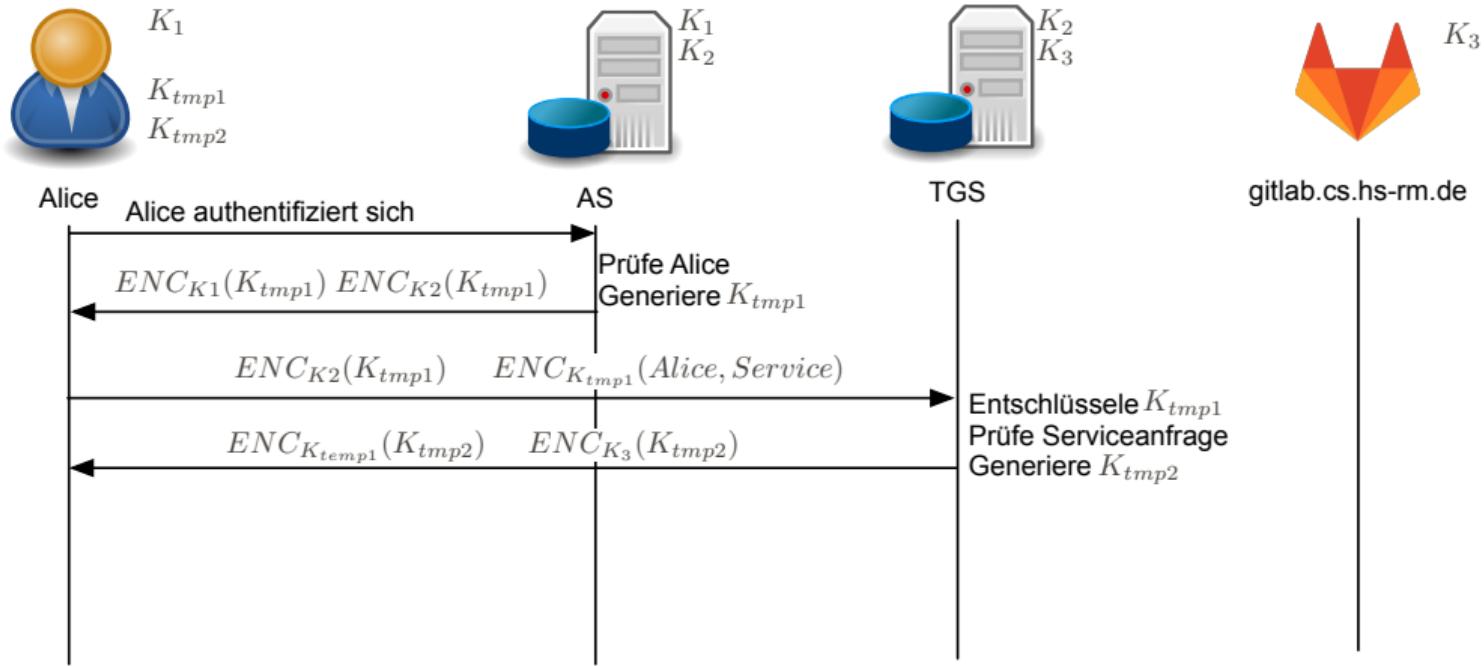
ABLAUF KERBEROS



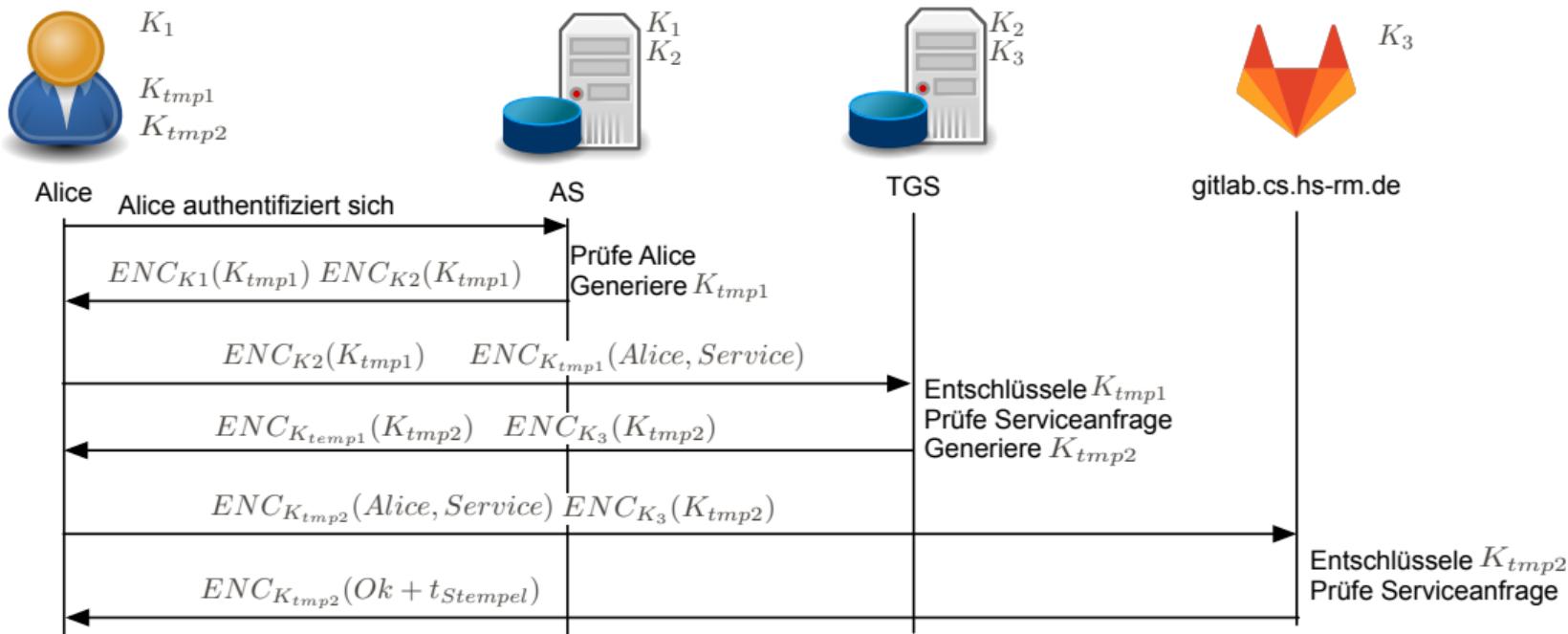
ABLAUF KERBEROS



ABLAUF KERBEROS

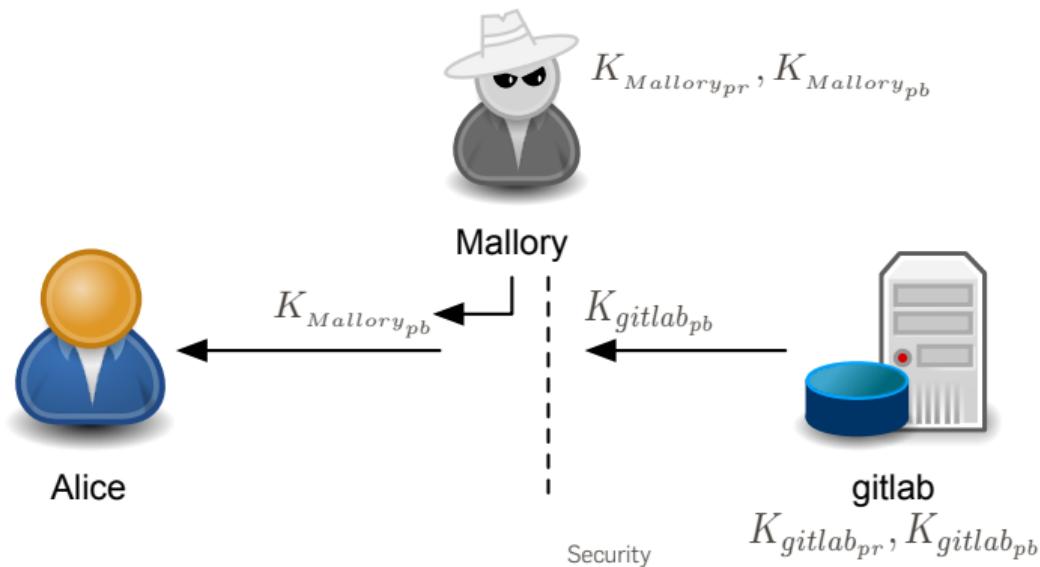


ABLAUF KERBEROS



AUTHENTIFIKATION OHNE AUSGETAUSCHTE GEHEIMNISSE

- **Problem:** Webseite wurde noch nie besucht, kein ausgetauschtes Geheimnis verfügbar
- Bedrohung: Mallory tauscht öffentlichen Schlüssel bei initialer Übertragung aus, um Kommunikation abzufangen



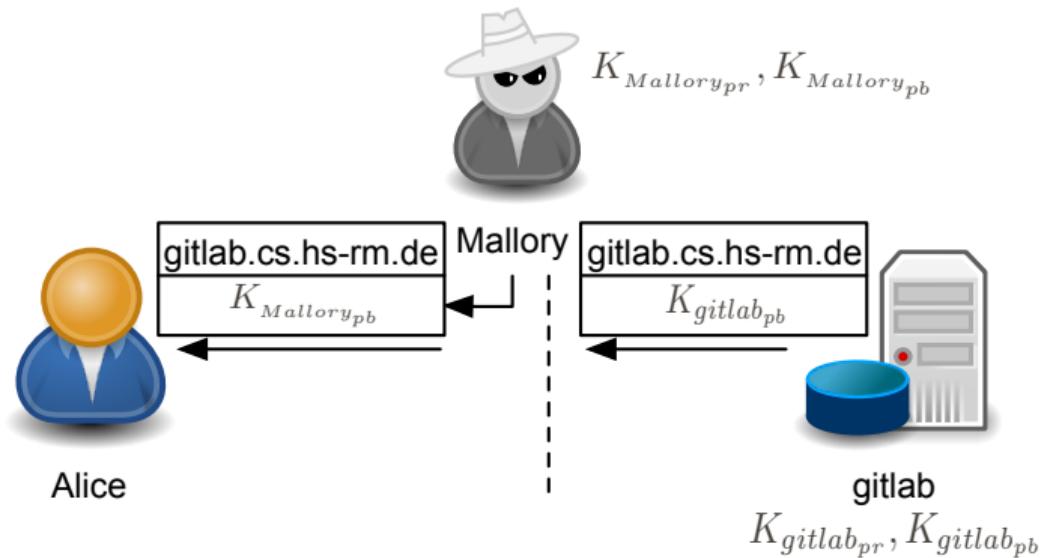
ZERTIFIKATE(1/3)

- Zertifikate verknüpfen den öffentlichen Schlüssel des Webservers mit Attributen, die nachgeprüft werden können (z.B. DNS-Name oder IP)
- Zertifikate enthalten u.a. die folgenden Informationen
 - Gültigkeitsdauer (Beginn / Ende)
 - Verwendungszweck des Schlüssels (Signieren / Verschlüsseln)
 - Verwendete kryptographischen Algorithmen
- Zertifikat mit öffentlichem Schlüssel wird übertragen

Subject Name	
Common Name	gitlab.cs.hs-rm.de
Issuer Name	
Country or Region	US
Organisation	Let's Encrypt
Common Name	R3
Serial Number	03 7C BE 99 A1 D0 4B 85 E0 36 24 F3 5D
Version	3
Signature Algorithm	SHA-256 with RSA Encryption (1.2.840.11
Parameters	None
Not Valid Before	Tuesday, 21. February 2023 at 08:21:00 C
Not Valid After	Monday, 22. May 2023 at 09:20:59 Centra
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	None
Public Key	256 bytes: BB 3E B9 B2 5E 15 22 D2 ...
Exponent	65537
Key Size	2.048 bits
Key Usage	Encrypt, Verify, Wrap, Derive
Signature	256 bytes: 34 DB 0E C9 49 10 1D 50 ...

ZERTIFIKATE(2/3)

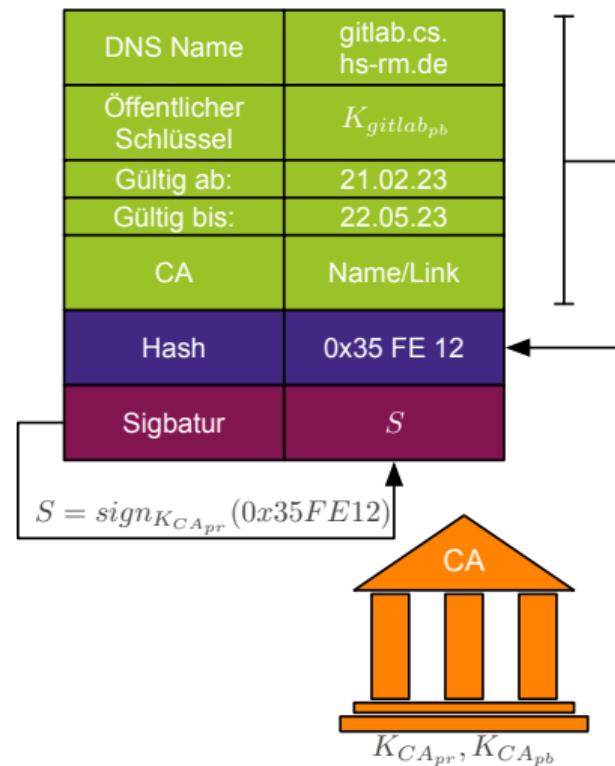
→ **Problem:** Authentizität des Zertifikates ist nicht sichergestellt.



ZERTIFIKATE(3/3)

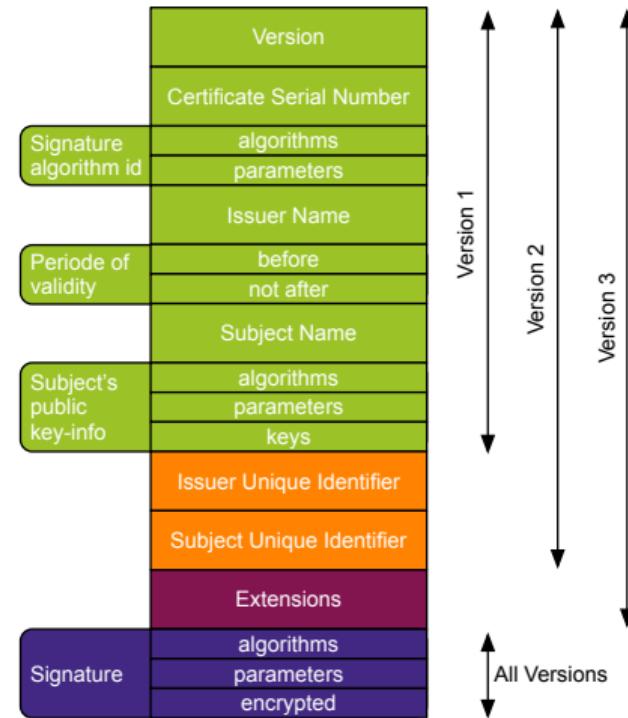
- Zertifizierungsstelle (CA) erstellt digitale Signatur eines Zertifikates
 1. Identität der CA wird in Zertifikat aufgenommen
 2. Hashwert über Zertifikatsinhalt wird gebildet
 3. CA mit Schlüsselpaar (K_{pr}, K_{pb}) signiert Hashwert des Zertifikates
 4. Signatur wird an Zertifikat angehangen

- Nutzer kann Zertifikat mit Schlüssel $K_{CA_{pb}}$ der CA prüfen



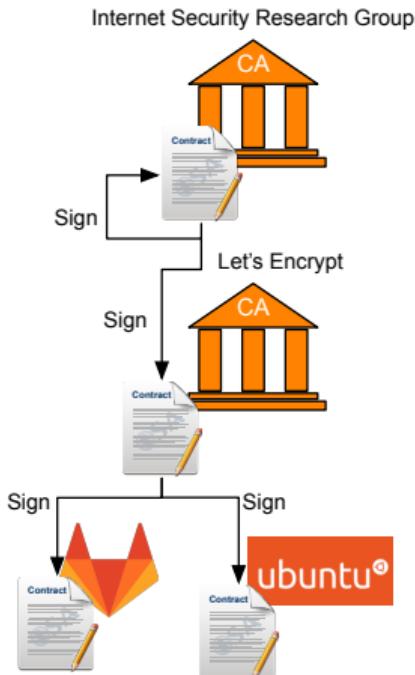
X.509 ZERTIFIKAT

- Version 1:
 - X.509-Version
 - Seriennummer
 - Algorithmus der Unterschrift
 - Name des Ausstellers
 - Gültigkeit
 - Zertifikatsinhaber
 - Öffentlicher Schlüssel des Zertifikatsinhabers
- Ergänzungen in Version 2:
 - Austeller-ID
 - Zertifikatsinhaber-ID
- Ergänzungen in Version 3:
 - Erweiterungen
- Signatur



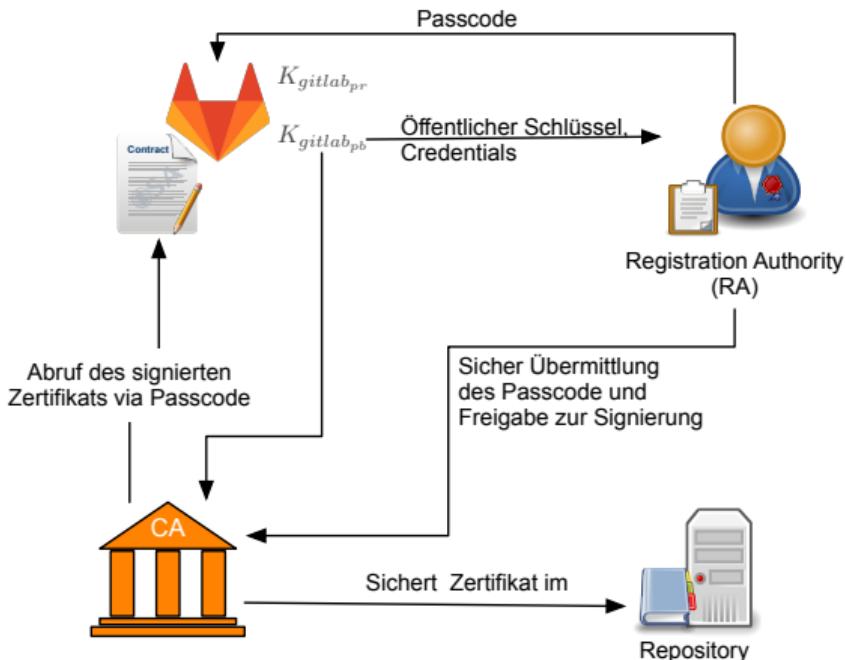
PUBLIC KEY INFRASTRUKTUR (PKI)

- Eine **Public Key Infrastruktur (PKI)** wird benötigt, um
 - Zertifikate zu erstellen und zu signieren
 - Informationen über Zertifikate bereitzustellen
 - Unsichere Zertifikate zu entfernen
- Teilnehmende in einer PKI
 - **Wurzel-CA**: Oberste Zertifizierungsstelle
 - **CA**: Stellt Zertifikate aus
 - **Webseiteninhaber**: Beantragen Zertifikate unter Nachweis der Identität (z.B. durch gitlab einer zufällig erzeugten Datei unter einer gegebenen URL)
- Zertifikat der Wurzel-CA liegt im Betriebssystem



ZERTIFIKAT ERZEUGEN

1. Generieren von $K_{gitlab_{pr}}$ und $K_{gitlab_{pb}}$
2. Übermitteln der Credentials und $K_{gitlab_{pb}}$ an (RA)
3. RA übermittel Passcode zum Abruf des Zertifikats
4. Übermittlung des $K_{gitlab_{pb}}$ an die CA
5. RA übermittelt Passcode an CA, wenn Credentials valide sind
6. CA stellt signiertes Zertifikat bereit, welches über Passcode abrufbar ist
7. CA übermittel Zertifikat an ein Repository zur Archivierung



ZERTIFIKAT PRÜFEN

1. gitlab Zertifikat unbekannt
2. Prüfe Gültigkeit und Identität
3. Prüfe Signature
4. CA Zertifikat (Let's Encrypt) unbekannt
5. Prüfe Gültigkeit und Identität
6. Prüfe Signature
7. Root CA Zertifikat bekannt und verifiziert
8. Signatur CA Zertifikat OK
9. CA Zertifikat Authentisch
10. gitlab Zertifikat Signatur OK
11. gitlab Zertifikat ist authentisch

Internet Security Research Group



ZERTIFIKAT WIDERRUFEN

- Grund für Certificate Revocation:
 - K_{pr} kompromittiert
 - Gerät , Benutzer oder Identität sperren
- CA erstellt einen Certificate Revocation List (CRL)
 - Seriennummer der widerrufen Zertifikate
 - CRL werden auf Webservern oder in Active Directories hinterlegt
 - Gültigkeit von Tagen, somit keine kurzfristige Sperrung
- Alternative Online Certificate Status Protocol (OCSP [RFC 6960])
 - Validierungsdienst zur Gültigkeit von Zertifikaten

ZERTIFIKAT PINNING

- Angabe welches Zertifikat für welche Webseite anerkannt wird
 - Öffentlicher Schlüssel-Hash verweist auf ein Zertifikat der Website selbst oder auf ein Stamm-/Zwischenzertifikat einer CA, die der Website bekannt ist
- Erschwert MiTM-Angriffe und Austausch von Zertifikaten
 - Direkte Verbindung zwischen einem Zertifikat und einem Hostnamen
 - Bei Erstzugriff wird der Schlüssel-Hash im Browser gespeichert
 - Hashwert kann Zertifikatsschlüssel oder von der ausstellenden CA sein
- Erhöhter Managementaufwand durch Zertifikat Pinning
 - Es werden mindestens zwei valide Hashwerte benötigt, damit ein neuer Hashwert nachgeladen werden kann

ZUSAMMENFASSUNG

- Merkmale zur Authentifizierung sowie Kombinationen dieser (2FA)
- Vor- und Nachteile der verschiedenen Merkmale
- Challenge-Response Protokoll zur Authentifizierung
- Konstruktion von Challenge-Response Protokollen basierend auf kryptographischen Primitiven konstruieren
- Prinzip von Single-Sign-On Protokollen und Kerberos
- Verwendungszweck von Zertifikaten und einer PKI
- Generations- und Verifikationsprozess von Zertifikaten innerhalb einer PKI