Quantenschlüssel unter der Verwendung von Kerberos

Bachelorarbeit von

Kris Schirmer

1184657

Universität der Bundeswehr München

Fakultät für Informatik

der Bundeswehr
Universität München

Quantenschlüssel unter der Verwendung von Kerberos

Bachelorarbeit von
Kris Schirmer
1184657

Aufgabensteller:	Prof. Dr. Udo Helmbrecht
Zweitprüfer:	Dr. Wolfgang Gehrke
Betreuer:	Dr. Nils gentschen Felde Hedwig Körfgen
Abgabetermin:	15. März 2021

Universität der Bundeswehr München
Fakultät für Informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Im Zeitalter der Digitalisierung wird die sichere Kommunikation immer bedeutsamer. Neue Technologien stellen Verfahren in Frage, die heute noch als sicher gelten. Um auch in Zukunft für eine sichere Kommunikation sorgen zu können, muss bereits präventiv geplant werden. Mit dem MuQaNet Projekt wird an der Universität der Bundeswehr ein Quantennetz realisiert, welches in der Lage ist, auch in Zukunft die Sicherheit des Datenverkehrs zwischen den Einrichtungen zu gewährleisten. Mit Technik der Firma ID Quantique SA, werden Quantenschlüssel erzeugt und zur Kommunikation verwendet. Quantenschlüssel sind Schlüssel, die durch echten physikalischen Zufall entstehen. [18] In den ID Quantique Geräten wird dieser echte Zufall durch die Polarisierung des Lichts und der anschließenden Messung der Photonen erreicht. [16] Quantenprotokolle wie das BB84 Protokoll, verwenden sichere Quantenkanäle, um die Übermittlung der Quantenschlüssel zu erreichen. Zum heutigen Stand der Technik beträgt die maximale nutzbare Distanz für die Übermittlung von Quantenschlüsseln circa 100 Kilometer. [12] Mit Hilfe dieses Quantenkanals lässt sich ohne Umwege eine symmetrische Verschlüsselung einrichten, welche in den Grundlagen Kapitel genauer betrachtet wird. Wie diese Quantenschlüssel genutzt und zu den Diensten und Nutzern gelangen, ist Thema dieser Arbeit. Um dieses Ziel zu erreichen, wird das vom MIT entwickelte Kerberos Protokoll herangezogen. Dieses Protokoll wird in mehreren Fällen modifiziert werden, damit es die Quantenschlüssel für eine sichere Kommunikation verwendet. Das MIT Kerberos Protokoll wird verwendet, da es bereits mehrere Jahrzehnte in Gebrauch ist und damit viele Erfahrungswerte gewonnen wurden. Die Ausgangssituation

wird sein, dass zwei der ID Quantique Geräte in den Gebäuden des Instituts aufgestellt sind. Diese sind über einen Quantenkanal verbunden und erzeugen über diesen identische Schlüssel auf beiden Geräten. Zu Beginn wird auf die notwendigen Grundlagen zu dem Verständnis dieser Arbeit eingegangen. Nachdem die Grundlagen gelegt sind, werden die Anwendungsfälle aufgestellt, in denen Quantenschlüssel in dem Kerberos Protokoll verwendet werden. Anschließend werden einige dieser Anwendungsfälle auf ihre Umsetzbarkeit geprüft. Ziel dieser Arbeit soll es sein, das Kerberos Protokoll mit Quantenschlüsseln zu erweitern. Mit diesem quantenmodifizierten Authentifizierungsprotokoll soll anschließend eine sichere Kommunikation zwischen den Einrichtungen erreicht werden.

Kapitel 2

Grundlagen

In diesem Kapitel wird das benötigte Wissen für das Verständnis zu dieser Arbeit vermittelt. Ziel ist es nach diesem Kapitel zu Verstehen wie das Kerberos Protokoll funktioniert, zu Wissen was Quantum Key Distribution (QKD) ist und wie dieses genutzt wird.
Es wird erläutert, welche heutigen kryptographischen Verfahren existieren und weshalb die Notwendigkeit für das QKD besteht. Außerdem wird das Kerberos Protokoll betrachtet, weshalb sich dieses besonders gut eignet zur Umsetzung eines sicheren Netzes und wie das Protokoll aufgebaut ist beziehungsweise wie dessen Abläufe sind.

2.1 Kryptographie

Die Kryptographie als Teilgebiet der Kryptologie und ist die Wissenschaft der Verschlüsselung von Informationen. Die Worte kryptos und graphein haben ihren Ursprung im Altgriechischen und bedeuten verborgen und schreiben. Wie der Name bereits Aufschluss gibt, wird die Kryptographie dazu verwendet, Verfahren zu entwickeln, um Texte beziehungsweise Informationen zu verschlüsseln, um sie nur für Sender und Empfänger lesbar zu machen. Die Kryptographie erfüllt durch ihre Verfahren folgende Ziele:

- die Vertraulichkeit,

- die Integrität,

- die Authentizität und

- die Verbindlichkeit.

Alle vier Ziele dienen dem Schutz von Daten und Nachrichtenkanälen. Nicht im jedem Nutzungsfall müssen aber alle vier dieser Ziele erreicht werden. Zum Informationsaustausch nutzen moderne kryptographische Verfahren verschiedene Arten der Verschlüsselung. Im weiteren Verlauf werden wir die symmetrische und asymmetrische Verschlüsselung betrachten. [19]

2.1.1 Symmetrische Verschlüsselung

Bei der symmetrischen Verschlüsselung handelt es sich um ein Verfahren, dass lediglich einen von allen Parteien genutzten Secret Key verwendet. Hier besteht die Herausforderung darin, diesen gemeinsamen Schlüssel sicher an alle Parteien zu übermitteln, ohne dass eine dritte Partei Zugang zu diesem hat. Es gibt verschiedene Verfahren, mit denen man durch einen symmetrischen Schlüssel seinen Klartext chiffrieren kann. Generell lässt sich sagen, je länger der Schlüssel desto länger die benötigte Zeit, um im Falle eines Angriffes den verschlüsselten Text zu dechiffrieren. Die symmetrische Verschlüsselung findet beispielsweise Anwendung in dem Data Encyrption Standard (DES), beim 3DES und dem Advanced Encryption Standard (AES), letzteres wird bis heute verwendet und gilt als ungebrochen.

2.1.2 Asymmetrische Verschlüsselung

Typische asymmetrische Verschlüsselungsverfahren nutzen das mathematische Problem der Primfaktorzerlegung. Hierfür übermitteln Alice und Bob, um bei den traditionellen Figuren der Kryptographie zu bleiben, ihre Public Keys untereinander. Diese Nutzen beide, um die Nachricht mit dem Public Key des Empfängers zu verschlüsseln. Sendet nun Alice an Bob eine Nachricht ist nur dieser in der Lage, die geheime Nachricht mit seinem Privat Key zu entschlüsseln. Mit diesem Verfahren wird das mathematische Problem der Faktorisierung genutzt. Hierbei wird darauf gesetzt, dass die Berechnung der Schlüssel nicht adäquater Zeit geschehen kann. Schon heute ist der Shor-Algorithmus bekannt, der mit Hilfe eines potenten Quantencomputers dieses Verfahren in kürzester

Zeit unbrauchbar machen würde. Heutige Verfahren wie Diffie-Hellman oder das RSA nutzen diese asymmetrische Verschlüsselung.

2.1.3 Hybride Verschlüsselung

Die hybride Verschlüsselung bezeichnet ein Verfahren, welches sowohl die symmetrische als auch die asymmetrische Verschlüsselung nutzt. Es wird zunächst die asymmetrische Verschlüsselung genutzt, um einen sicheren Kanal zur Kommunikation zu generieren, über welchen dann anschließend die symmetrischen Schlüssel übertragen werden. Der Wechsel auf die symmetrische Verschlüsselung findet statt, da die symmetrische Verschlüsselung deutlich schneller durchgeführt werden kann, als die asymmetrische.

2.1.4 Schlüsselmanagement

Um diese kryptographischen Verfahren nutzen zu können ist es notwendig diese Schlüssel zu verwalten und an die kommunizierenden Parteien zu übermitteln. Hierfür werden Schlüsselmanagement Systeme genutzt. Die zentralen Aufgaben eines solchen Systems sind die Generierung, Übermittlung und anschließende Vernichtung alter oder kompromittierter Schlüssel (Abbildung 2.1). Eines dieser Systeme ist das QKD.

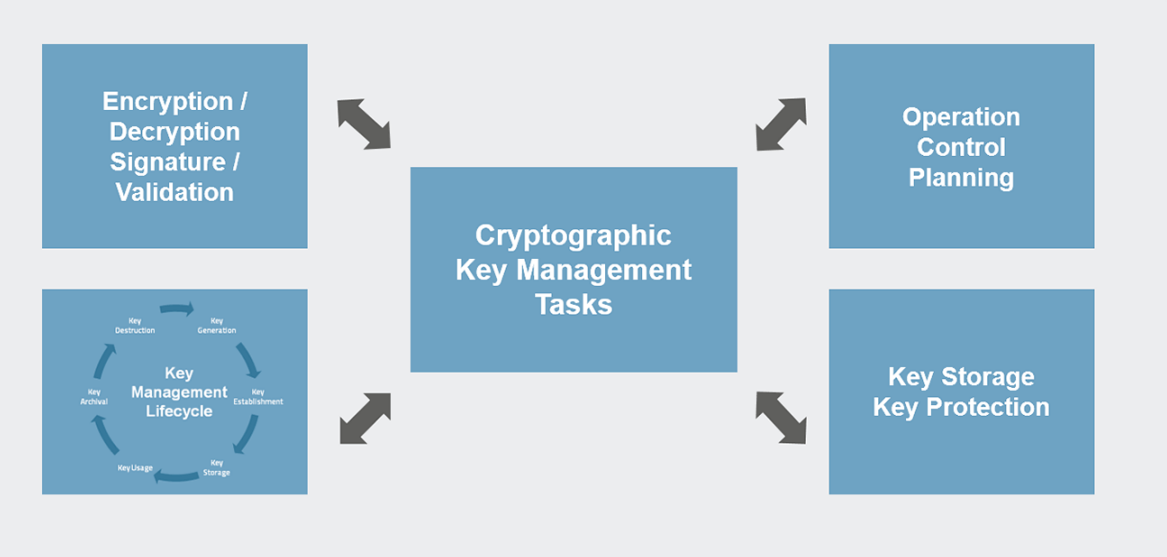


Abbildung 2.1: Darstellung einer Cross-Realm Authentifikation [1]

2.2 Quantum key Distribution

Beim QKD handelt es sich um ein kryptographisches Schlüsselmanagement, welches aufgrund der Art der Generierung der Schlüssel aber deutlich sicherer ist. Die Quantenschlüssel werden durch die Polarisierung von Lichtphotonen gewonnen und unterliegen somit einem physikalisch echtem und nicht wie aktuell verwendete Schlüssel dem maschinellen Zufall, welcher im **Pseudo Random Number Generator (PRNG)** generiert wird. Aufgrund der quantenmechanischen Eigenschaften ist es mathematisch beweisbar, dass die Quantenschlüssel bei Verwendung als One-Time-Pad sicher sind vor Angriffen. Des Weiteren ist es durch die Polarisierung der Lichtphotonen unmöglich, zweimal den selben Schlüssel zu erzeugen, was ein Abfangen und Weiterleiten bereits im Ansatz unterbindet.
Um in der Quantum Communication Infrastructure (QCI) auch zukünftig Sicherheit zu garantieren, müssen die durch das QKD generierten Schlüssel sicher verwaltet und übertragen werden. Dies lässt sich durch quantensichere Protokolle bewerkstelligen. [24] [5] [4] [17]

BB84 Protokoll

Das 1984 von Charles H. Bennett und Gilles Brassard entwickelte Protokoll dient zum Schlüsselaustausch in der Quantenkryptographie und ist eines der bekanntesten Verfahren, dass sich mit dieser Thematik beschäftigt. Hierbei werden die Photonen, die zur Informationsübertragung dienen, beim Sender im Filter in eine zufällige Richtung polarisiert. Beim Empfänger wird ebenfalls zur Messung eine zufällige Ausrichtung des Filters gewählt.
Wurden die Photonen gemessen, muss zunächst abgesprochen werden, ich welcher Filterausrichtung gemessen, beziehungsweise Empfangen wurde. Abschließend werden einige Messergebnisse beider Parteien verglichen und die Korrektheit des Schlüssels überprüft. Sollte diese ausreichend zufriedenstellend sein, ($> 25\%$ Übereinstimmung) werden falsche Bits verworfen und eine bestimmte Sequenz der Übertragung als Schlüssel verwendet.
Sollte ein Angreifer die Übertragung abfangen wollen, würde er sich selbst verraten. Aufgrund quantenmechanischer Prinzipien würden sich Photonen beim Mitlesen in ihrer Polarisation verändern. Dank dieser Eigenschaft ist es möglich zu erfahren, ob und

wie viel von der Übertragung abgefangen wurde. Ebenso ist es physikalisch unmöglich, den exakt selben Schlüssel (Non Cloning Theorem[23]) nach dem Abfangen weiterzuleiten, was eine hohe Sicherheit des Protokolls bewirkt. [24] [22] [8] [20]

Ekert Protokoll

Einen weiteren Ansatz für das QKD fand Artur Ekert. Er nutzte den Effekt aus der Quantenphysik, mit der die Informationen der Quantenpartikel durch die Quantenverschränkung übermittelt werden. Bei dem Ekert Protokoll wird eine einzige Quelle zur Versendung der Photonen genutzt. Diese Quelle verschränkt die Photonen und versendet diese jeweils an die beiden Kommunikationspartner Alice und Bob. Zum Empfangen müssen sich Alice und Bob ebenso wie bei dem BB84 Protokoll absprechen. Nach der Übertragung der Photonen müssen sie kommunizieren, in welcher Polarisationsausrichtung sie die Photonen gemessen haben. Es werden dann nur die Messergebnisse verwendet, die in der selben Ausrichtung gemessen wurden. Aufgrund der Quantenverschränkung verhält es sich so, dass Bob das exakte Komplement von Alice gesendeten Informationen erhält. Somit muss entweder Bob oder Alice ihre Messung lediglich flippen, um einen gemeinsamen Key zur Verschlüsselung zu erhalten. Hier kann die Quelle für die Schlüssel günstig zwischen den beiden Empfängern platziert werden, um so die Reichweite des Protokolls zu verdoppeln, was ein Vorteil gegenüber dem BB84 Protokoll ist. [22] [8] [20]

2.2.1 QKD Analyse

Schlüsselmanagement-Systeme dienen zur Verwaltung verschiedener Arten von Schlüsseln egal ob symmetrisch oder asymmetrisch. Zur Verwaltung zählen ebenso die Aufgabenfelder der Generierung neuer Schlüssel sowie die Zustellung an die angeforderte Stelle (Abbildung 2.1). Mit der Entwicklung der Quantenprotokolle wurde eine neue sichere Art entdeckt, mit der man Schlüssel erzeugen, verteilen und sicherstellen kann, dass niemand außer den beiden Empfängern diesen erhalten hat. Die Effekte der Quantenphysik ermöglichen es, Schlüssel aus physikalisch echtem Zufall zu erzeugen, da diese aus der Polarisierung von Lichtphotonen ausgelesen werden. Zur Erzeugung dieser so gewonnen Quantenschlüssel werden Protokolle wie etwa das BB84 oder das Ekert Protokoll verwendet. Trotz der unterschiedlichen Ansätze nutzen beide physi-

kalische Effekte aus der Welt der Quanten und sind somit resistent gegen jeden Versuch eines herkömmlichen Angriffs. Hier werden Denial of Service Angriffe, die durch Beschädigung des Übertragungsmediums entstehen außer Acht gelassen. [9] [6]

Stärken des QKD

Herkömmliche Schlüssel Management Systeme sind bereits der Kern der sicheren Kommunikation eines jeden Netzwerks, das Wert auf Integrität und Authentizität legt. Als solches erfährt es auch einen erhöhten Schutzbedarf und die damit einhergehenden Maßnahmen. Die Quantum key distiribution bringt von Natur aus einen hohen Schutz mit sich. Durch die veränderte Strukturierung des Systems und den quantenphysikalischen Eigenschaften wird ein Angriff auf die Schlüssel deutlich erschwert.

Effekte der Quantenphysik

Aufgrund der Quantenphysik werden die Quantenschlüssel mathematisch beweisbar sicher. Außerdem wird es unmöglich, diese Schlüssel abzufangen, da sie aufgrund des Non-Cloning-Theorems [23] nicht mehr repliziert werden können. Sollte dennoch der Versuch unternommen werden, die Schlüssel zu kompromittieren, ließe sich anhand der Veränderung des Schlüssels beim Abgleich von Sender und Empfänger ermitteln, ob und wie viel der Angreifer von der Sequenz empfangen hatte.

Dezentralität

Gewöhnliche Schlüsselmanagement Systeme sind zentral angelegt auf einem Server, von welchem aus sie ihren Dienst verrichten. Worin die Herausforderung besteht, jeden Schlüssel aus einer Datenbank, an beide benötigten Parteien sicher zu übermitteln. Dieses Problem wird beim QKD umgangen. Die Schlüssel werden dort über einen Quantenkanal übermittelt und somit bei jeder Partei parallel in einer Datenbank gespeichert. Eine Steuerkonsole, die an jedem Quantengerät angeschlossen ist, reguliert dort den Zugriff und übermittelt die Nutzung eines Schlüssels an das jeweilige andere Ende der Leitung.

Schwächen des QKD

Das QKD ist eine verhältnismäßig neue Entwicklung der Technik und steht somit noch am Beginn seiner potentiellen Fähigkeiten. Aufgrund der andauernden Verbesserungen dieses Systems weist es noch leichte Schwächen auf, die mit voranschreitender Zeit entschärft werden dürften.

Datenübertragungsrate

Die Erzeugungsrate der Quantenschlüssel ist nach heutigen Maßstäben noch gering, da sie selbst in den neuesten Projekten auf eine Rate von maximal 17 Mbit/s kommen. Die Übertragungsraten der Informationen mittels QKD fallen dagegen noch deutlich niedriger aus. Im Jahr 2008 begannen erste Versuche zur Erstellung eines Quantennetzes über 20 Kilometer Distanz. In diesem schaffte man es, Schlüssel mit der Geschwindigkeit von 1 Mbit/s auszutauschen. Mit steigender Größe des Netzes vielen die Übertragungsraten deutlich in den Kbit/s Bereich ab. Neueren Projekten gelang es die Datenübertragung deutlich zu steigern, eine chinesische Firma schaffte es auf eine Übertragungsrate von 13,7 Mbit/s, allerdings unter Laborbedingungen. [10]

Reichweite

Aktuell ist es möglich, Quantenschlüssel über Glasfaser bis zu einer maximalen Entfernung von 100km zu übertragen. Allerdings fällt mit zunehmender Reichweite die Übertragungsrate stark ab, was dazu führt, dass nach 100 km nur noch wenige Kbit/s beim Empfänger ankommen. Alles, was die 100 km Grenze übersteigt, ist als nicht mehr nutzbar anzusehen. Um dieses Limit zu umgehen, müsste daher ein dichtes Netz an Trusted Nodes aufgebaut werden, die den Schlüssel empfangen und weiterleiten bis er den Empfänger erreicht. Die Alternative ist, sich vom Kabel zu lösen und die Quantenschlüssel über Laser von einem Satelliten aus zu übertragen, der den Schlüssel an die gewünschten Bodenstationen übermittelt. Mit dieser Technik können Inter-Kontinentale Quantennetze implementiert werden. Hierbei ist jedoch zu beachten, dass diese bisher nur bei besten Wetterbedingungen übertragen können, da Nebel und Wolken die Lichtphotonen verändern und den Schlüssel verfälschen. [12]

Architektur des QKD

Das QKD ermöglicht es zwischen zwei Parteien (A und B) Schlüssel auszutauschen. Wird eine dritte Partei (C) hinzugezogen, muss diese jeweils mit A und B einen Schlüssel austauschen. Aufgrund der Architektur des QKD ist es nicht möglich, dass C den selben Schlüssel verwendet, den A und B für ihre Kommunikation verwenden. Es ist also nicht möglich einen Netzübergreifenden Schlüssel zu verwenden, sondern muss für jede Verbindung separat angelegt werden.

2.2.2 Fazit QKD

Auf den ersten Blick bietet das QKD alles was für eine sichere Kommunikation nötig ist. Es verbindet zwei Punkte und bietet unter diesen einen sicheren Schlüsselaustausch zur Kommunikation. Protokolle wie das BB84 oder Ekert Protokoll wurden entwickelt, um Informationen auf quantenphysikalischer Basis zu übertragen. Das QKD hat aber zwei stark limitierende Faktoren. Zum einen die Reichweite für die Schlüsselübertragung, die zum heutigen Stand im Glasfaserkabel nicht über 100 Kilometer hinausreicht, und zum anderen die noch sehr geringe Datenübertragungsmenge von nur wenigen Kbit/s, wenn die Kommunikationspartner sich nicht beinahe in nächster Nähe befinden. Des Weiteren kommt die starke Empfindlichkeit der Lichtphotonen hinzu. Da kein Übertragungsmedium völlig rauschfrei ist, wird es bei Übertragungen zu häufigen Fehlern in der Information kommen.
Somit bleibt das QKD als Lösung zum initialen und weiteren Austausch von symmetrischen Schlüsseln, wodurch ein asymmetrisches Verfahren und eine potenzielle Gefahr ausgeschlossen werden. Mit QKD können die klassischen Authentifizierungsprotokolle sinnvoll erweitert und über einen sicheren Kanal initialisiert werden. Aber nicht nur der erste Schlüsselaustausch könnte damit erzeugt werden, es bieten sich wie im Folgenden beschrieben durchaus mehrere Möglichkeiten an, das QKD mit Blick auf die Sicherheit einzubringen.

2.3 Kerberos Protokoll

Das ursprünglich am MIT entwickelte Kerberos Protokoll wird zur Authentifizierung von Nutzern und Diensten in verteilten Systemen genutzt. Kerberos kann durch seine

Funktionsweise in unsicheren Netzen verwendet werden, da es auf die Übertragung von Passwörtern verzichtet und anstelle dessen Tickets durch das Netz sendet, welche nur vom jeweiligen Sender und Empfänger entschlüsselt werden können. Das Protokoll wird seit Jahrzehnten verwendet und stetig verbessert. Microsoft hat es seit Windows 2000 standardmäßig installiert. Linux bietet ebenfalls eine Kerberos Implementierung an. Beide Versionen können miteinander ohne Probleme interagieren. [14] [11]
Wie der namensgebende Hund aus der griechischen Mythologie, teilt sich das Protokoll Kerberos in drei große Bausteine auf, die teils direkt, teils indirekt miteinander agieren. Der Kern von Kerberos befindet sich im **Key Distribution Center (KDC)**, welches sich aus dem **Authentification Server(AS)** und den **Ticket Granting Server(TGS)** zusammensetzt. Die verbleibenden beiden „Köpfe“ sind der Client oder Service, der bei dem KDC sein Anliegen schildert und sich dort authentifizieren muss, um Zugang zu erhalten. Der letzte Teil ist der Server, auf dem sich der angeforderte Dienst oder Nutzer befinden.
In Kerberos werden alle Entitäten als Principals bezeichnet, die von dem Systemadministrator angelegt werden müssen. Ausnahmen bilden hier der AS und der TGS, die bei der Installation des Protokolls automatisch erstellt werden. Jeder Principal verfügt über einen eindeutig definierten Namen, einer ID und einem zugehörigen Passwort, welche ihm die Verifizierung im KDC ermöglichen.

Zur Strukturierung und besseren Verwaltung mehrerer KDCs, kann jedes KDC einem eigenem Bereich zugeteilt werden. Diese Bereiche werden **Realm** genannt und sind als Objekte in einer Baumstruktur zu betrachten. Für diese Arbeit wird die Top-Level Realm „UNIBW.DE“ angelegt sein mit den beiden Sub Realms „INF.UNIBW.DE“ und „CODE.UNIBW.DE“. Diese Baumstruktur ist mit Windows zwingend erforderlich, unter Linux könnte aber auf den Top-Level Realm UNIBW.DE verzichtet werden. Die Funktionsweise und Kommunikation der Realms wird in der **Cross-Realm (CR)** Authentifizierung genauer betrachtet.

Kerberos verwendet zur Verschlüsselung zwei verschiedene Arten von Schlüssel. Zum einen die Long-Term Passwörter der Principals. Jeder Teil in diesem Protokoll, der mit einem anderem kommunizieren kann, auch die des KDCs, werden als Principal behandelt und somit hat auch jedes sein eigenes Passwort. Somit verfügen der TGS, jeder

Beschreibung der essenziellen Principals	
Principal Name	Beschreibung
K/M@INF.UNIBW.DE	Der Master Key Principal.
kadmin/history@INF.UNIBW.DE	Speichert die Kennwortverläufe anderer Principals. In jedem Master-KDC vorhanden.
kadmin/kdc@INF.UNIBW.DE	Principal der Zugriff auf das KDC erlaubt.
kadmin/changepw@INF.UNIBW.DE	Principal der Passwortänderungen erlaubt.
krbtgt/INF.UNIBW.DE@INF.UNIBW.DE	Principal der zu Generierung des tgt genutzt wird.
krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE	Principal für Cross-Realm Trust und zur Erzeugung eines Cross-Realm tgt.
root/inf.unibw.de@INF.UNIBW.DE	Der root Principal.
username@INF.UNIBW.DE	Normaler Principal eines Nutzers.
username/admin@INF.UNIBW.DE	Administrator Principal, der zur Administration des KDC genutzt werden kann.

Tabelle 2.1: Wichtige Kerberos Principals

Service und jeder Nutzer über ein Long-Term Passwort. Das KDC führt zusätzlich eine **Datenbank (DB)** über alle Principals. Diese DB wird für die verschlüsselte Kommunikation zwischen KDC und Principals benötigt. Die DB verschlüsselt die Passwörter mit dem Masterkey, welcher ebenfalls ein Long-Term Passwort ist. Folgend eine Auflistung der für diese Arbeit wichtigsten Principals: 2.1

Wenn ein Nutzer nun einen Dienst anfordert, erhält dieser im Lauf der Anmeldung mindestens zwei Short-Term Passwörter. Diese sogenannten **Session Keys** (Abbildung 2.2) bestehen für die Dauer einer Session und sind nach Ablauf dieser zu erneuern. Die Dauer einer solchen Session beträgt standardmäßig zehn Stunden, kann aber nach Bedarf administriert werden. Zum einen erhält der Nutzer vom AS einen **TGS Session Key** und im weiteren Verlauf vom TGS den **Service Session Key**. Letzterer berechtigt schließlich zum Zugang zu dem geforderten Dienst.
Nun ist es bei Kerberos aber ebenso möglich und nötig, dass sich Dienste im KDC authentifizieren können. Diese können sich selbstverständlich nicht selbst mit einem Password bei dem AS verifizieren. Hierfür bietet Kerberos die Möglichkeit für diese Service Principals ihren Schlüssel in einer keytab, zu deutsch Schlüsseltabelle, abzu-

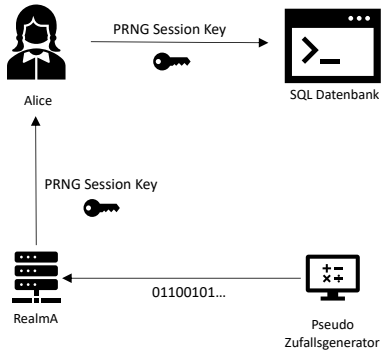


Abbildung 2.2: Erzeugung eines Session Keys

legen und sich mit dieser beim KDC anzumelden. Service Principals sind wie folgt aufgebaut: service-name / hostname @ realm. In der Tabelle 2.2 einige Beispiele für Service Principals.

Um Zugriff auf das KDC zu erhalten, werden die kadmin und kadmin.local Kommandozeilen Befehle benötigt, die die Schnittstelle zur Administration des Kerberos Systems darstellen. Der Unterschied zwischen diesen beiden Befehlen besteht darin, dass kadmin.local direkt Zugang zur KDC Datenbasis hat, während kadmin auf den Daemon kadmind zugreift. Um zu den kadmin Befehlen Zugriff zu erhalten, muss der Nutzer sich mit dem Administrator Passwort des KDC authentifizieren. Folgend einige ausgewählte kadmin Befehle, die diese Arbeit betreffen:

2.3.1 Kerberos Authentifizierung

Die Authentifizierung im Kerberos Protokoll ist ein Prozess der mehrere Schritte durchläuft, von denen der Nutzer aber nur die wenigsten manuell behandeln muss. Die Kerberos Authentifizierung verlangt vom Nutzer lediglich eine einmalige Aufforderung zur Passwort Eingabe (Single-Sign-On). Werden im Verlauf der Session weitere Dienste benötigt, registriert der AS die bereits vorliegende Verifizierung des Nutzers und dieser wird automatisch durch das Protokoll bei weiteren Diensten authentifiziert.
Sollte Alice nun einen Dienst anfordern, so ist sie gezwungen sich bei ihrem KDC anzumelden. Im Verlauf der Authentifizierung wird sie sich bei weiteren Principals

Darstellung Service Principals	
Principal Name	Beschreibung
host/ssh.inf.unibw.de@INF.UNIBW.DE	Dieser Principal wird von kerberisierten Services und Anwendungen genutzt. Er wird außerdem zur Verifizierung genutzt, um zu zeigen, dass das TGT zu dem Client aus dem richtigen KDC stammt.
psql/pgs.inf.unibw.de@INF.UNIBW.DE	Principal für kerberisierte PostgreSQL Anwendung.
mongodb/mdb.inf.unibw.de@INF.UNIBW.DE	Principal für kerberisierte MongoDB Anwendung.
neo4j/neo.inf.unibw.de@INF.UNIBW.DE	Principal für kerberisierte Neo4j Anwendung.
HTTP/www.code.unibw.de@CODE.UNIBW.DE	Principal der Webauthentifizierung ermöglicht.
kadmin/changepw@INF.UNIBW.DE	Principal der zur Änderung des Passwortes herangezogen werden kann.

Tabelle 2.2: Service Principal Name

Wichtige kadmin Befehle	
addprinc (options) *neuerprinc*	Fügt einen neuen Principal hinzu
-randkey	Erzeugt für den Principal ein PRNG Passwort
-policy	Legt die Passwort Policy für einen Principal fest, muss zunächst in **add_policy** definiert werden
-kvno	Gibt die Schlüssel Versionsnummer aus
delete_principal *principal*	Löscht den ausgewählten Principal
change_password (options) *principal*	Führt zur Änderung des Passwort des Principals
session_enctypes	Definiert den Verschlüsselungsstandard für angeforderte Session Keys des Principals
add_policy (options) *policy*	Fügt eine Passwort Policy namens policy zur Database hinzu
-maxlife *time*	Setzt die maximale Lebensdauer für ein Passwort fest
-histroy *number*	Setzt die Anzahl an alten Passwörtern die der Principal behält. Diese Option wird nicht im LDAP KDC Database Modul unterstützt.
-pw *password*	Setzt das Passwort auf den darauffolgenden String, hier password

Tabelle 2.3: Wichtige kadmin Befehle

verifizieren, was automatisiert geschieht ohne weiteres Zutun. Unter Umständen wird Alice durch mehrere KDCs weitergeleitet, bis sie den gewünschten Server erreicht, was als Cross-Realm bezeichnet, aber zunächst wird die Anmeldung im eigenem Realm behandelt.
Für diesen ersten Fall betrachten wir das Beispiel, in dem sich Alice und der geforderte Dienst im selben Realm befinden. Für den weiteren Verlauf wurden im RFC 4120 [15] feste Begrifflichkeiten festgelegt, die hier Verwendung finden:

Begriffe für die Authentifizierung laut RFC 4120:		
Nachricht von	**Nachricht an**	**RFC 4120 Syntax**
Client	Authentication Server	KRB_AS_REQ
Authentication Server	Client	KRB_AS_REP
Client	Ticket Granting Server	KRB_TGS_REQ
Ticket Granting Server	Client	KRB_TGS_REP
Client	Application Server	KRB_AP_REQ
Application Server	Client	KRB_AP_REP

Tabelle 2.4: RFC 4120

Schritt 1 Alice sendet eine KRB_AS_REQ an den Authentication Server (AS) im Klartext mit folgendem Inhalt: User ID, User IP Adresse, geforderter Service ID, geforderte Ticket Lifetime

Schritt 2 Der AS sendet eine KRB_AS_REP an Alice. Die Nachricht besteht aus zwei Teilen. Der erste Teil, der mit Alice Passwort verschlüsselt wurde enthält: Ticket Lifetime, Timestamp, TGS ID, Ticket Granting Server (TGS) Session Key. Der zweite Teil, der mit dem TGS Passwort verschlüsselt ist, enthält das Ticket Granting Ticket (TGT). Das Ticket enthält: User ID, User IP-Adresse, Ticket Lifetime, Timestamp und den TGS Session Key für Alice.

Die Tickets sind für den Client unzugänglich, da diese ansonsten manipuliert und eine Einfallstelle für Angriffe darstellen würden.

Schritt 3 Alice sendet eine KRB_TGS_REQ an den TGS. In diesem wird ein Ticket für den gewünschten Service angefordert. Zusätzlich befindet sich in der Anfrage

das TGT und eine Authentifikation, die verschlüsselt wird mit dem TGS Session Key.

Schritt 4 Der TGS entschlüsselt das TGT mit seinem Long-Term Passwort und erhält den TGS Session Key. Mit Hilfe des Session Keys entschlüsselt das TGS die Authentifizierung, um Alice zu verifizieren.

Schritt 5 Der TGS sendet ein KRB_TGS_REP an Alice. Die Nachricht besteht aus zwei Teilen. Der erste Teil, der mit dem TGS Session Key verschlüsselt wurde enthält: Ticket Lifetime, Timestamp, Session ID, Service Session Key. Der zweite Teil enthält das Service Session Ticket, welches mit dem Service Passwort verschlüsselt wird. Das Ticket enthält: User ID, User IP-Adresse, Ticket Lifetime, Timestamp und den Service Session Key für Alice.

Schritt 6 Alice entschlüsselt das KRB_TGS_REP mit dem TGS Session Key und erhält den Service Session Key. Zu dem Service Ticket hat Alice keinen Zugang.

Schritt 7 Alice sendet eine KRB_AP_REQ an den Service. In diesem befindet sich das Service Ticket und eine Authentifizierung, die mit dem Service Session Key verschlüsselt ist.

Schritt 8 Der Service sendet eine KRB_AP_REP an Alice. Diese enthält eine Authentifizierung des Service welche mit dem Service Session Key verschlüsselt wurde.

Schritt 9 Alice entschlüsselt die KRB_AP_REP um ihrerseits den Service zu verifizieren und kann so sicher sein, dass es sich wirklich um dem gewünschten Service handelt.

Für die Basis Authentifizierung in Kerberos müssen folgende Dinge gegeben sein:

1. der geforderte Dienst muss im KDC identifizierbar sein,

2. der Dienst muss eine eindeutige ID besitzen,

3. die ID des Dienstes muss im KDC vorliegen, ansonsten kann kein Ticket ausgestellt werden,

4. mit Active Directory (AD) können Nutzer sowie Computerobjekte registriert werden,

5. Kerberos macht keinen Unterschied zwischen Nutzer und Computer, sondern unterscheidet nur die Namen.

In Schritt 1 ist zu sehen, dass der erste Request des Nutzers im Klartext an den AS gesendet wird und dieser darauf folgend eine Response sendet, die der Nutzer mit seinem Passwort entschlüsseln muss. Um einer unnötigen Überlastung des AS entgegenzuwirken und dieser nicht jede Request mit einer vollständigen Response beantwortet, kann eine Pre-Authentifizierung vorgeschaltet werden, in der der AS zuvor eine Verifizierung des Nutzer verlangt. Diese Pre-Authentifizierung findet zwischen den beiden ersten Schritten der Standard Authentifizierung statt und sieht wie folgt aus.

Schritt 1 Der AS sendet eine KRB_AS_REP an Alice. Diese Nachricht enthält einen Error welcher verlangt, dass der Nutzer sich Pre-Authentifiziert.

Schritt 2 Alice sendet eine KRB_AS_REQ an den AS. Die Nachricht enthält den Timestamp verschlüsselt mit ihrem Passwort.

Schritt 3 Der AS entschlüsselt die KRB_AS_REQ mit Alices Passwort, welches der AS aus der Datenbank bezieht. Ist diese Entschlüsselung erfolgreich und der Timestamp wurde akzeptiert, wird mit dem Standard Protokoll fortgefahren.

Nachdem der Authentifizierungsvorgang erfolgreich abgeschlossen wurde, hat der Nutzer Zugang zu seinem geforderten Dienst für die Dauer der Session. Werden weitere Dienste benötigt, erfolgt der Erhalt des Session Keys schneller, da der AS alle Nutzer, die zurzeit einen Session Key verwenden, in seinem Cache speichert und somit die Verifizierung stark abgekürzt wird. Der Dienst, wie auch in der Grafik 2.3 schematisch dargestellt ist, hat während des gesamten Vorgangs kein einziges Mal direkten Kontakt zum KDC. Der Dienst ist nur indirekt mit dem KDC verbunden, da das KDC lediglich die registrierten Daten des Servers, beziehungsweise die Daten des darauf laufenden Dienstes aus der DB bezieht und diese zur Verschlüsselung nutzt. [21]

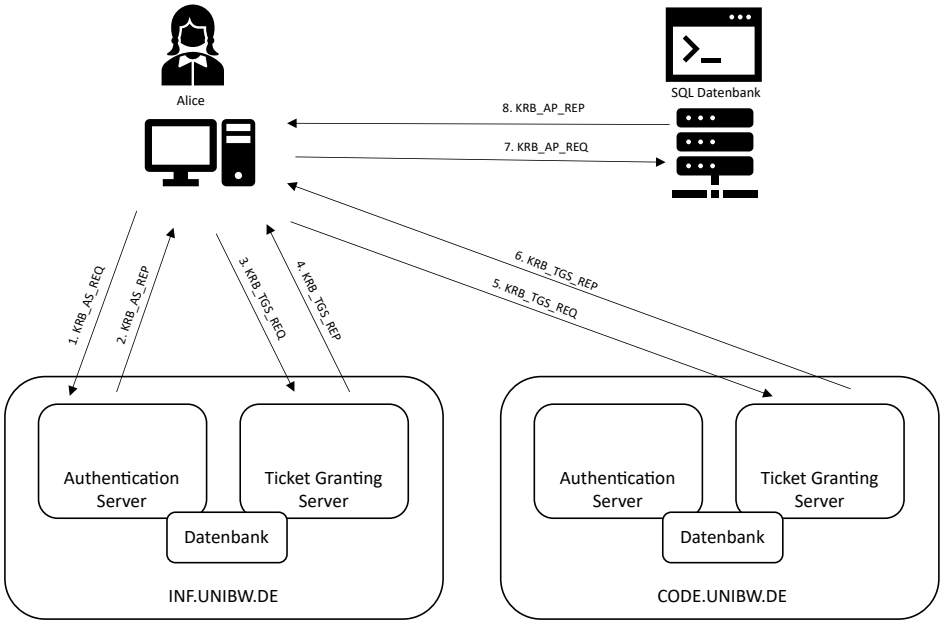


Abbildung 2.3: Darstellung einer Cross-Realm Authentifikation

2.3.2 Kerberos Cross-Realm Authentifizierung

Jedes KDC befindet sich bei Kerberos in seiner eigenen Realm, wobei eine Realm mehrere KDC verwalten kann, aber nicht umgekehrt. Eine Realm kann als Objekt betrachtet werden, welche als Attribute die Principals führt. Ein Realm funktioniert auf sehr ähnliche Weise wie eine Domain. Dies wird auch durch die Namensgebung der Realms gezeigt, es ist Konvention denselben Namen für den Realm zu wählen wie für die Domain, jedoch in Großbuchstaben. So wäre zum Beispiel die Domain unibw.de als Realm UNIBW.DE. Im obigen Beispiel haben wir gesehen wie Alice sich bei einem Dienst anmeldet, der sich im selben Realm befindet wie sie selbst. Im folgenden wird der Fall betrachtet, in dem Alice im INF.UNIBW.DE Realm registriert ist und der benötigte Dienst im CODE.UNIBW.DE. Hierfür bietet Kerberos eine weitere Funktion, das Cross-Realm. Den KDCs ist es möglich untereinander zu kommunizieren, wenn zuvor ein Trust zwischen diesen Realms hergestellt wurde. Dieser Trust würde in unserem Beispiel krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE im CODE Realm heißen oder krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE im INF Realm. Diese beiden Principals zeigen dem jeweiligen Realm an, dass es dem anderen Realm vertraut. Hierfür ist entscheidend, dass beide Principals dasselbe Passwort teilen (Abbildung 2.4). [3]

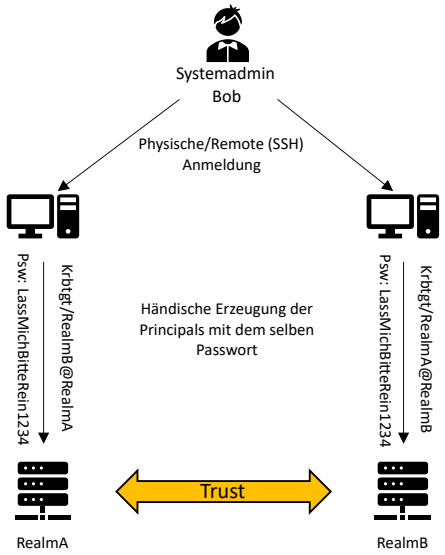


Abbildung 2.4: Erstellung eines Cross-Realm Trusts

Dieses betrachtete Beispiel zeigt einen beidseitigen Trust, in dem Nutzer oder Services aus beiden Realms auf den jeweiligen anderen zugreifen können. Dies ließe sich auch als einseitiger Trust darstellen, wodurch Nutzer aus dem Realm INF Zugriff auf Realm CODE hätten, aber nicht umgekehrt.
Es können beliebig viele Trusts von einem KDC ausgehen. Hierbei ist auch zu beachten, dass der Trust transitiv ist. Besteht ein Trust zwischen Realm A und B und ein weiterer zwischen B und C, vertraut somit A ebenfalls C, sowie umgekehrt.
Die ersten Schritte beim Cross-Realm sind exakt dieselben, wie wenn nur ein Realm beteiligt wäre. Der erste Unterschied findet erst statt, wenn Alice sich an das TGS wendet für die Service ID. In diesem Beispiel befindet sich Alice im INF Realm und möchte einen Dienst aus dem CODE Realm aufrufen.

Schritt 1 Alice sendet eine KRB_TGS_REQ an den TGS. In diesem wird ein Ticket für den gewünschten Service angefordert. Zusätzlich befindet sich in der Anfrage das TGT und eine Authentifikation, die verschlüsselt wird mit dem TGS Session Key.

Schritt 2 Der TGS entschlüsselt das Ticket mit seinem Passwort und erhält so den TGS Session Key. Mit dem TGS Session Key entschlüsselt der TGS die Authentifizierung und kann so Alice verifizieren.

Schritt 3 Der TGS stellt bei der Überprüfung fest, dass der angeforderte Services sich außerhalb des eigenen Realms im CODE Realm befindet. Der TGS sendet eine KRB_TGS_REP an Alice. Die Nachricht besteht aus zwei Teilen. Der erste Teil, der mit dem TGS Session Key verschlüsselt wurde enthält: Ticket Lifetime, Timestamp, Cross-Realm TGS ID, Cross-Realm TGS Session Key. Der zweite Teil, der mit dem geteilten Cross-Realm TGS Passwort verschlüsselt ist enthält das CODE TGT. Das Ticket enthält: User ID, User IP-Adresse, Ticket Lifetime, Timestamp und den TGS Session Key für Alice.

Schritt 4 Alice entschlüsselt die KRB_TGS_REP mit dem TGS Session Key und erhält so den Cross-Realm TGS Session Key.

Schritt 5 Alice sendet eine zweite KRB_TGS_REQ, dieses Mal jedoch an das KDC im CODE Realm. In diesem wird ein Ticket für den gewünschten Service angefordert. Zusätzlich befindet sich in der Anfrage das CODE TGT und eine Authentifikation, die verschlüsselt wird mit dem TGS Session Key.

Schritt 6 Der CODE TGS entschlüsselt das Ticket mit dem Cross-Realm TGS Passwort und erhält so den Cross-Realm TGS Session Key. Mit dem Cross-Realm TGS Session Key entschlüsselt der CODE TGS die Authentifizierung und kann so Alice verifizieren. Hier ist es nicht nötig, dass Alice im CODE Realm registriert ist, da die Verschlüsselung ausreicht um sie als Principal zu verifizieren.

Schritt 7 Der TGS sendet eine KRB_TGS_REP an Alice. Die Nachricht besteht aus zwei Teilen. Der erste Teil, der mit dem Cross-Realm TGS Session Key verschlüsselt wurde, enthält: Ticket Lifetime, Timestamp, Cross-Realm Service ID, Service Session Key. Der zweite Teil, der mit dem Service Passwort verschlüsselt ist enthält das Service Ticket. Das Ticket enthält: User ID, User IP-Adresse, Ticket Lifetime, Timestamp und den Service Session Key für Alice.

Schritt 8 Alice entschlüsselt die KRB_TGS_REP mit dem Cross-Realm TGS Session Key und erhält so den Service Session Key.

Schritt 9 Alice sendet eine KRB_AP_REQ an den Service. In diesem befindet sich das Service Ticket und eine Authentifizierung, die mit dem Service Session Key verschlüsselt ist.

Schritt 10 Der Service entschlüsselt das Ticket mit seinem Passwort und erhält so den Service Session Key. Mit dem Service Session Key entschlüsselt der Service die Authentifizierung und kann so Alice verifizieren.

Schritt 11 Der Service sendet eine KRB_AP_REP an Alice. Diese enthält eine Authentifizierung des Service welche mit dem Service Session Key verschlüsselt wurde.

Schritt 12 Alice entschlüsselt die KRB_AP_REP und kann so den Service ihrerseits verifizieren und kann so sicher sein, dass es sich wirklich um dem gewünschten Service handelt.

Für Cross-Realm sind folgende Voraussetzungen nötig:

1. Netzwerkzugang vom Client zu jedem angesprochenem KDC in den Realms.

2. Adresszuordnungen (Mapping) der Netzwerke zu den jeweiligen Realm Namen

3. Eine Menge von gemeinsamen Cross-Realm Pricncipal und zugehörige Long-Term Keys

2.3.3 Kerberos Protokoll Analyse

Da es sich bei Kerberos um ein Protokoll handelt, welches ausschließlich auf symmetrische Verschlüsselung setzt, bietet es sich an, dieses Protokoll in ein Netz zu implementieren welches quantensicher werden soll, da die symmetrische Verschlüsselung zuweilen als die mit der größten Resistenz gegen Quantenalgorithmen gilt. Zudem bietet Kerberos viele Möglichkeiten, es gezielt durch die Verwendung von Quantenschlüsseln zu verstärken und so für weitere Sicherheit des Netzwerks zu sorgen. Durch gezielte Plugins, die den ursprünglichen Quellcode nur erweitern, ohne diesen in seiner Funktion zu modifizieren, ließen es zu, dass an der bewährten Sicherheit des Protokolls keine ungewollten Lücken entstehen. So müsste auch kein Nachweis für den Bestand der Sicherheit erbracht werden. Ziel bei der Erweiterung muss es dabei vorrangig sein, das Protokoll kompatibel zu alten, beziehungsweise nicht mit Quanten modifizierten Kerberos Versionen zu lassen, da dieses Netz in bisher bestehende Infrastrukturen eingebunden werden können soll.

Der KDC Server hat erweiterten Schutzbedarf und muss deshalb vor physischem Zugang geschützt stehen, da dort die Daten aller Principals samt Passwörtern in der Datenbank aufbewahrt werden. Im Zuge der Sicherheit ist es durchaus sinnvoll, den AS und den TGS auf physikalisch verschieden Geräten zu implementieren und zusätzlich einen Slave Server zu erstellen, um Verfügbarkeit und Redundanz zu erhöhen.
Bei Kerberos ist die Synchronisation der Zeit auf den Servern ein wichtiger Aspekt. Da alle Response, Requests und Tickets mit einem Zeitstempel versehen werden und die Tickets zusätzlich über eine Lebensdauer verfügen, ist es essenziell, dass die Zeit auf den Servern synchronisiert ist. Andernfalls könnte es sonst mit der Validierung der Tickets Probleme geben, da diese abgelehnt werden würden, wenn der Zeitunterschied zu groß wäre zwischen Zeitstempel und der aktuell auf dem Server laufenden Zeit. Anstelle von Passwörtern sendet Kerberos Tickets durch das Netz, welche mit Hilfe der Keys verschlüsselt werden. Die Kerberos Principals verfügen alle über Long-Term Keys, die im regulären Betrieb deutlich seltener erneuert werden als sie eigentlich sollten, gerade im Bereich der Nutzer.
Durch die Transitivität der Trusts zwischen den KDCs bildet Kerberos die Möglichkeit, die Reichweite des QKD zu erhöhen. Da die kabelgebundene Reichweite des QKD begrenzt ist, kann die Reichweite durch die Eigenschaft des Cross-Realm ohne weiteres erhöht werden.

Potentielle Schwachstellen

Kerberos hat sich in den letzten Jahren als durchaus sicher erwiesen. Es liegt als Open-Source Code vor und gewährleistet somit, dass jeder transparent das Verfahren des Protokolls nachvollziehen und bei Verbesserungen mitwirken kann. Jedoch sind auch hier einige Schwachstellen bekannt, die es zu beleuchten gilt.

Kerberoasting

In dem Standard Authentifizierungsverfahren von Kerberos sendet der Nutzer dem AS eine unverschlüsselte Request. Diese Request beantwortet der AS mit einer Response, die mit dem Passwort des Nutzers verschlüsselt wurde. Sollte es einem Angreifer gelingen sich als ein registrierter Nutzer auszugeben, erhält dieser somit das gehashte Passwort des Nutzers. Mit Hilfe eines Offline Angriff auf das Passwort kann nun das

Passwort berechnet werden. Sowie dies gelungen ist, kann der Angreifer erneut eine Request an den AS senden und erhält mit dem Passwort Zugang in das Netz.

Golden Ticket

Bei diesem Angriff versucht Alice zunächst, einen Zugang in das Netzwerk zu finden, beispielsweise mit Phishing-E-Mails. Von diesem aus wird auf Bobs Administrator Konto Zugriff erlangt. Sowie Alice es erreicht hat dieses Konto zu kompromittieren, kann sie das KDC derart manipulieren wodurch sie sich ein Golden Ticket erstellen kann. Dieses ermöglicht es ihr, ohne die Aufmerksamkeit auf sich zu lenken, sämtliche Computer, Daten und sonstige Informationen, die sich in diesem Netz befinden zu befallen. Dies ist möglich, da das Protokoll aufgrund des Golden Tickets Alice gewähren lässt und sie für diesen Vorgang autorisiert hält.

Denial of Service

Sollte der Server jeden Request mit einem Response beantworten, bietet dies eine weitere Schwachstelle, die es auszuschließen gilt. Sollte der Angreifer den Server mit Requests überhäufen, würde dieser entweder wegen Überlastung den Dienst einstellen oder aber wirkliche Anfragen nicht mehr bearbeiten können. So könnte der Dienst in dem Netzwerk eingestellt werden.

Angriff auf Zeitsynchronisation

Sollte es einem Angreifer gelingen, die Quelle für die Synchronisation der Zeit zu manipulieren, besteht die Möglichkeit, dass der Angreifer alte bereits abgelaufene Tickets wieder verwenden kann. Somit wäre es nicht einmal nötig, an das Passwort eines Nutzers zu gelangen, da es genügen würde, eine Nachricht abzufangen, mit der er dann erneut nach Umstellung der Zeit auf dem Server Zugriff erhalten würde.

Schutzmaßnahmen

Wie an den vorangegangenen Beispielen gezeigt, weist Kerberos einige Schwachstellen auf. Die meisten Angriffe lassen sich jedoch durch leichte administrative Maßnahmen beseitigen.

Pre-Authentifizierung

Angriffe wie Kerberoasting oder DoS lassen sich mit Hilfe der Pre-Authentifizierung vermeiden. Hierbei sendet Bob zusätzlich zu dem unverschlüsselten Request an den Authentication Server einen Zeitstempel, der mit seinem Passwort verschlüsselt wurde. Der Authentication Server prüft zunächst den verschlüsselten Teil der Nachricht und prüft, ob dieser sich mit dem Passwort des vermeintlichen Nutzers entschlüsseln lässt. Erst wenn dies der Fall ist beginnt er mit der Erstellung der Response. Somit ist eine Überflutung mit Requests ausgeschlossen und es wird zuvor schon verifiziert, dass es sich um den tatsächlichen Nutzer handelt und niemand versucht unrechtmäßig an das Passwort zu gelangen.[21]

Public Key Kryptography für initiale Authentication in Kerberos (PKINIT)

Ein weiterer Ansatz zur Vermeidung des Kerberoasting ist das PKINIT. Das Problem bei der Pre-Authentifizierung liegt in der Verschlüsselung mit dem Passwort des Nutzers. Diese Passwörter wurden von Menschen festgelegt und sind daher bedeutend schwächer als die restliche Verschlüsselung, die im KDC stattfindet. Daher ist jedes System lediglich so stark wie das schwächste Passwort der Nutzer. Um dafür zu sorgen, dass sich nicht jeder Nutzer ein 26-stelliges Passwort mit zufälligen ASCII Symbolen merken muss, kann eine asymmetrische Verschlüsselung eingeführt werden. Hierfür muss ein Schlüsselpaar zwischen Client und dem KDC verteilt werden. Wurde dies bewerkstelligt, muss der Client anschließend an den Request an den Authentication Server nur noch einen Teil anhängen, der mit Hilfe des Secret Keys verschlüsselt wurde und verifiziert sich dadurch bei dem KDC. [7]

Server Synchronisation

Der Zeitsynchronisation der Server sollte besonders viel Aufmerksamkeit gewidmet werden. Wenn diese Funktion nicht richtig arbeitet, wird das KDC unbrauchbar, da es sämtliche Anfragen der Nutzer abweist, sollte die Zeit zwischen den beiden über fünf Minuten abweichen. Diese Zeitspanne ist der Default Wert, der in den Einstellungen festgelegt ist und lässt sich bei Bedarf konfigurieren. Um die Synchronisation zu erreichen, gibt es verschiedene Methoden. Diese sollte aber auf einem besonders geschützten Rechner implementiert werden, im besten Fall auf demselben, auf dem das

KDC installiert ist. Mit einer funktionstüchtigen Zeitsynchronisation beschränkt man die Gefahr der Ausnutzung alter Tickets. [2]

2.3.4 Fazit Kerberos

Kerberos ist seit mehreren Jahrzehnten im Gebrauch und hat sich als sicher erwiesen. Zudem ist es seit Windows 2000 das Standard Verfahren zur Authentifizierung auf Windows Servern. Auf Unix Betriebssystemen ist es ebenso weitverbreitet und genutzt. Wenn die Pre-Authentifizierung im Server eingestellt ist und der Server physisch sicher verwahrt ist, ist es äußerst unwahrscheinlich, dass Angreifer eine Bedrohung darstellen. Einziges bestehendes Problem könnten schwache, vom Menschen festgelegte Passwörter darstellen.

In diesem Kapitel wurden die kryptographische Verfahren betrachtet und wie solche verwaltet werden. Eines dieser Verwaltungssysteme ist das QKD. Es wurde gezeigt weshalb dieses so relevant ist für die Sicherheit zukünftiger Systeme. Außerdem wurde das Kerberos Protokoll ausführlich in seiner Funktion dargestellt und weshalb dieses bis Heute einen hohen Sicherheitsstandard vorweist. Im nächsten Kapitel werden mögliche Fallbeispiele gezeigt, in denen das QKD mit Kerberos zusammengeführt werden.

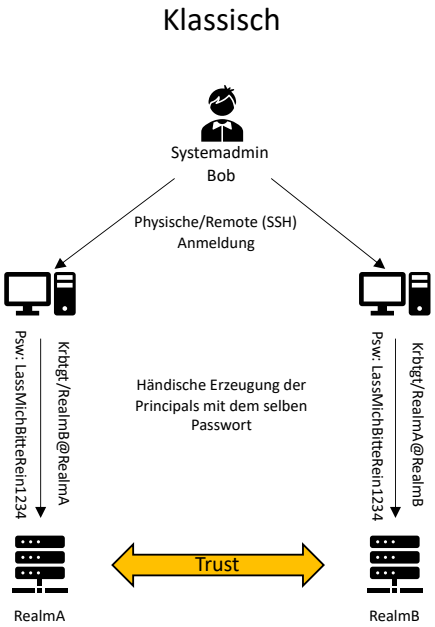
Kapitel 3

Konzepte zur Anwendung des QKD

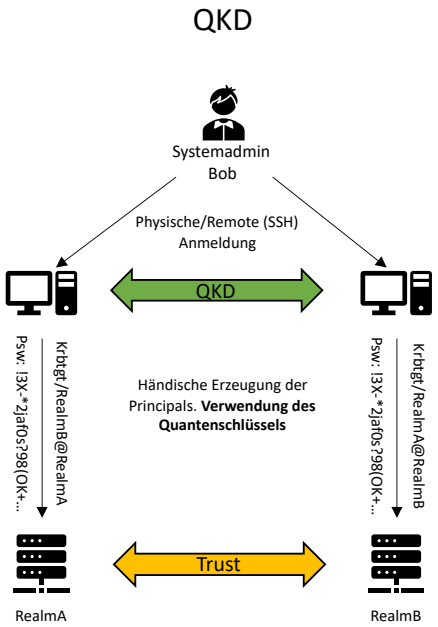
Im Vorangegangenen wurde gezeigt, wie QKD und Kerberos aufgebaut sind und welche Funktionen diese bieten. Bei der Analyse wurden Schwachstellen erörtert und wie diese vermieden oder vollends verhindert werden können. In diesem Kapitel wird betrachtet, wie die Nutzung von Quantenschlüsseln sinnvoll in Kerberos genutzt werden kann. Hier werden einige spezifische Abschnitte aus dem Protokoll betrachtet und aufgezeigt, wodurch die Einbringung des QKD ein Mehrgewinn an Sicherheit erzielt werden kann. Bei der Einbindung des QKD ist darauf zu achten, am Kerberos Protokoll selbst keine Änderungen vorzunehmen, sondern lediglich Erweiterungen zu entwickeln, die den ursprünglichen Quellcode unverändert lassen, um somit nicht den Beweis der Sicherheit des bestehenden Protokolls zu verlieren.

3.1 QKD Cross-Realm Trust

Aufgrund der Erzeugung des beidseitigen Trusts durch die Cross-Realm Principals krbtgt/REALMB@REALMA und krbtgt/REALMA@REALMB wird eine Kommunikation zwischen beiden Realms ermöglicht. Diese Principals haben entweder vom Menschen eingegebene Schlüssel oder von /dev/[u]random durch Pseudo Random Number Generator erzeugte Zufallszahlen. Diese Schlüssel sind in Kerberos als Long-Term Keys angelegt und werden auch als solche behandelt. Damit der Trust zustande kommt, müssen beide Principals über den selben Schlüssels verfügen. Dadurch ist es notwendig, diesen in den fremden Realm zu übertragen. Die asymmetrische Verschlüsselung sollte



(a) Klassische Erstellung eines Cross-Realm Trusts

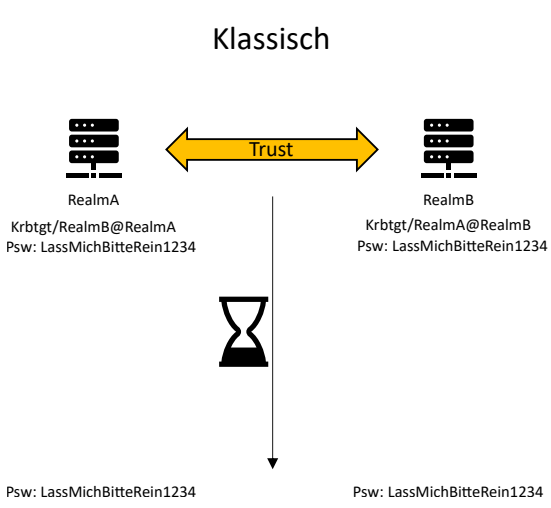


(b) QKD Erstellung eines Cross-Realm Trusts

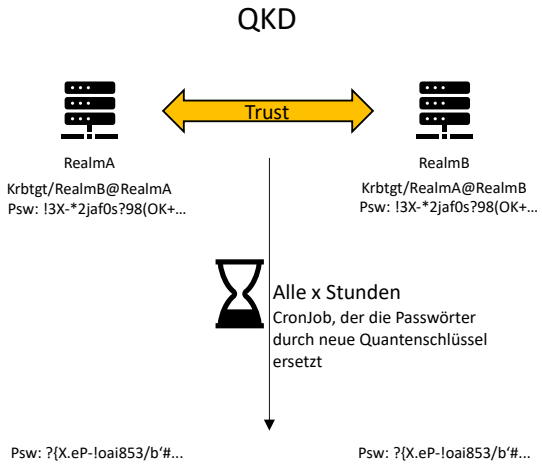
vermieden werden, da dieses Verfahren potenziell angreifbar ist und die Übertragung zum Beispiel per Mail ebenfalls Gefahren birgt, muss hier ein anderes Verfahren genutzt werden. Zu diesem Zeitpunkt der Implementierung liegt jedoch noch kein symmetrischer Schlüssel vor, weshalb eine Alternative gefunden werden muss. Durch die Nutzung von QKD bestünde die Möglichkeit, dieses Problem zu beheben. Durch das Quantenprotokoll ließe sich ein identischer Schlüssel auf beiden Seiten übertragen ohne Gefahr zu laufen, dass dieser abgefangen oder mitgelesen worden sein könnte.

Die Cross-Realm Principals werden vom Administrator wie in Abbildung 3.1a gezeigt, manuell erstellt und fordern von diesem die Eingabe des selben Passworts für beide Principals in den Realms. Üblicherweise wird der Principal im fremden Realm durch eine SSH-Verbindung erstellt, oder von dem dort zuständigen Admin des Realms selbst. Somit muss man ein asymmetrisches Verfahren verwenden oder dem Admin des Realms auf andere Art das Passwort für den Principal zukommen lassen, was wieder ein Risiko darstellt.

Durch die Etablierung des QKD wie in Abbildung 3.1b dargestellt, lässt sich bei der Erzeugung der Principals der Schlüssel sicher über den Quantenkanal übertragen. Da die Schlüssel in den Realms vorrätig gespeichert und verwaltet werden, wird diese sofort



(a) krbtgt Passwort Behandlung



(b) Dynamischer inter-KDC QKD Schlüssel

und ohne Umwege in beiden Realms verfügbar, ohne Nutzung von asymmetrischer Verschlüsselung oder sonstiger Übertragung der Schlüssel im Netz. Der Schlüssel lässt sich ohne weiteres bei beiden Cross-Realm Principals einsetzen, um so einen Trust zu erzeugen der die Vorteile eines Quantenschlüssels verwendet.
Die Passwörter der Trust Principals sollen wie in der ersten Anwendung bereits durch Quantenschlüssel ersetzt werden. Im weiteren Verlauf werden wir die periodische Erneuerung dieses betrachten.

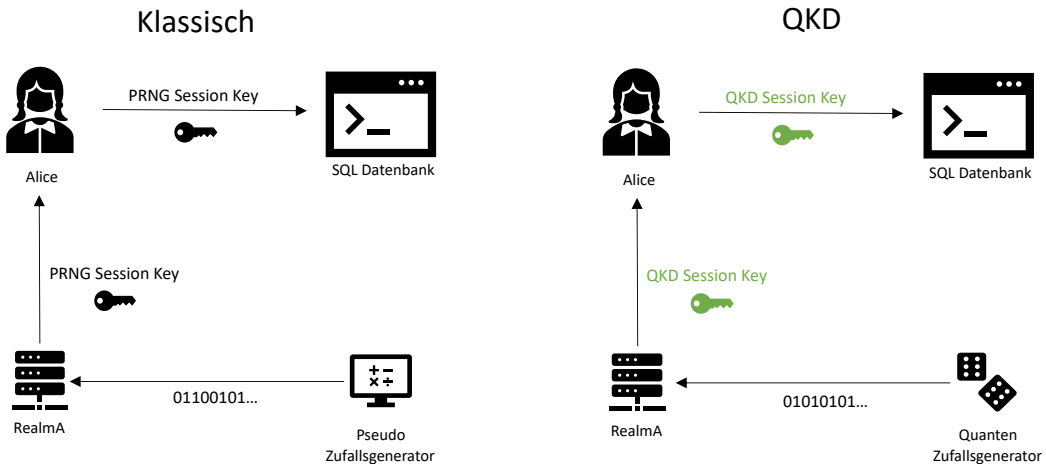
3.2 Trust Erneuerung

Die Passwörter der inter-KDC Principals sind als Long-Term Passwörter ausgelegt und haben laut Empfehlung des MIT mindestens 26 zufällige Zeichen aus dem ASCII-Zeichensatz.[13] Ist der Trust einmal hergestellt wird, wie in Abbildung 3.2a gezeigt, an diesen Principals für längere Zeit nichts mehr geändert. Mit steigender Lebensdauer erhöht sich jedoch die Gefahr eines Angriffs und der Entschlüsselung eines solchen Passworts. Somit ist es durchaus notwendig, diese Passwörter in regelmäßigen Abständen zu erneuern. Meist geschieht dies durch die händische Eingabe des Systemadministrators, da die Passwörter wieder an beide betroffene Realms übertragen und eingefügt werden müssen.
Durch die Einführung eines CronJobs der sich ca. alle 12 Stunden ausführen würde,

könnte man regelmäßig diese Long-Term Keys erneuern. Aufgrund der vorrätigen Speicherung von Schlüsseln im QKD lässt sich dies ohne weiteren Aufwand bewerkstelligen (Abbildung 3.2b). Um mathematisch beweisbare Sicherheit zu garantieren, muss ein Passwort nach jeder Verwendung erneuert werden, ein sogenanntes One-Time-Pad. Allerdings würde eine solche Implementierung auf einem Kerberos Server derart Ressourcen beanspruchen durch die ständige Erneuerung, dass es nicht mehr zielführend ist.
Da Kerberos die Passwörter nummeriert mit der Key version number (kvno) und die verfallenen Passwörter für eine bestimmte Zeit speichern kann, wissen beide Realms, dass sie aktuell mit der neusten Version des Trusts arbeiten. Dadurch ist es möglich, einen Session Key der noch unter dem vorherigen Trust Passwort zugewiesen wurde, seine Gültigkeit behalten kann und es nicht nötig werden würde, diese durch den CronJob ebenfalls zu erneuern. Zusätzlich wird die Zeit der KDC Server synchronisiert, wodurch ein Zeitversatz in den CronJobs ausgeschlossen werden kann und es garantiert ist, dass beide KDCs zum selben Zeitpunkt den Trust erneuern.
Eine alternative Variante der Erneuerung ist, nach einer gewissen Anzahl von Nutzungen den Trust neu zu setzen. Hierfür wird es nötig werden die Abrufung des Passworts zur Verschlüsselung abzufangen, damit die Erneuerung eingeleitet werden kann. Vor diesem Hintergrund ist es einfacher umsetzbar, das Passwort in festen regelmäßigen Abständen zu erneuern.

3.3 QKD Session Key

Die Session Keys in Kerberos dienen zur Verifikation des Nutzers gegenüber dem Dienst oder umgekehrt. Jeder in Kerberos erzeugte Session Key hat eine maximale Lebensdauer, die vom Administrator im KDC festgelegt wird. Somit wird verhindert, dass diese Schlüssel für einen dauerhaften Zugang im System missbraucht werden. Nach Ablauf der Session ist der Nutzer oder Dienst gezwungen, sich erneut bei seinem KDC zu authentifizieren und eine neue Session anzumelden. Um die Nutzung abgelaufener Session Keys zu verhindern, ist es zwingend notwendig, dass das KDC einer Zeitsynchronisation unterliegt und falls ein Cross-Realm Ticket angefordert wird beide beteiligten KDCs denselben Zeitgeber heranziehen. Für das Format des Schlüssels bietet Kerberos viele der gemeinhin geläufigen Verfahren an, wir werden aber hier wie für alle QKD Anwen-



(a) Session Key (b) QKD Session Key

dungen das AES-256 wählen. Im herkömmlichen Betrieb werden die Keys in Kerberos mit dem -randomkey Befehl erzeugt, womit die Cross-Realm Passwörter sowie auch die Session Keys mit dem selben Pseudo Random Number Generator erstellt werden (Abbildung 3.3a). Diese mit dem krbtgt/INF.UNIBW.DE@INF.UNIBW.DE Principal generierten Keys sind zwar lediglich als Short-Term Key angelegt, werden aber dennoch maschinell erzeugt und unterliegen deshalb auch keinem physikalisch echtem Zufall. Dieser durch Pseudozufall entstandene Keys könnte ebenso durch die Nutzung von QKD entschärft werden. Durch die Anweisung an das KDC den Schlüssel aus einer anderen Quelle zu beziehen, könnte ohne großen Aufwand ein hohes Maß an Sicherheit hinzugewonnen werden.

In der Kommunikation zwischen den Principals werden die Session Keys verwendet, hier ist auch das KDC selbst mit eingeschlossen. Diese Short-Term Keys haben den Vorteil, dass sie nur für die Dauer einer Sitzung gültig sind, im Normalfall circa zehn Stunden. Nach Ablauf dieser Zeitspanne wird der Nutzer gezwungen diesen zu erneuern, falls weiterer Bedarf besteht. Zudem fungieren die Session Keys als Authentifikation des Nutzers durch die reine Nutzung der Session Keys, da sie diese nur aus dem KDC selbst erhalten und somit eine vorangegangene erfolgreiche Authentifizierung voraussetzen. Nach der Authentifizierung in seinem KDC erhält der Principal einen Session Key für den von ihm geforderten Dienst. Dieser Key wird im KDC, genauer vom TGS, erzeugt. Hierfür wird der krbtgt Principal herangezogen, welcher bewirkt, dass durch den Pseudo Random Number Generator ein Schlüssel erzeugt und an den Principal

weitergeleitet wird. Mit diesem Key erhält der Principal Zugang zu dem Dienst. Dieser authentifiziert sich in der Regel ebenfalls beim Principal, um eine Verwechslung auszuschließen (Abbildung 3.3a).
Das KDC generiert diese Session Keys im Pseudo Random Number Generators welcher Pseudozufallszahlen ausgibt. Da diese Zahlen keinem echtem Zufall unterliegen sind sie aus mathematischer Sicht nicht beweisbar sicher. Man kann dem KDC somit echten physikalischen Zufall zukommen lassen, indem man hier das QKD anwendet (Abbildung 3.3b). Es genügt dem KDC die Quelle der Zufallszahlen neu zuzuweisen und ihm die Quantenschlüssel im benötigten Format bereitzustellen. Da wir den Anspruch größtmöglicher Sicherheit erreichen möchten, wird in diesem Fall das aes256-cts Format bevorzugt, welches standardmäßig von Kerberos akzeptiert wird. Der Advanced Encryption Standard (AES) mit einer Länge von 256 Bits weist bis zum heutigen Stand keine bekannte Angriffsmöglichkeit auf.

3.4 KDC Master-Slave Synchronisation

Das Kerberos Key Distribution Center ist wie andere KDCs ein Server mit erhöhtem Schutzbedarf und als solches ist es durchaus sinnvoll zur Erhöhung der Redundanz und der Fehlervermeidung ein Slave KDC einzurichten, welches beim Ausfall des Masters die Arbeit übernimmt, bis der Fehler behoben werden konnte. Die Einrichtung eines Slaves ist in wenigen Schritten umgesetzt und lässt dem Systemadministrator die Möglichkeit zu entscheiden, wie häufig die Synchronisation zwischen Master und Slave stattfindet. Wie alle anderen Vorgänge in Kerberos wird auch für den Master, den Slave, sowie für die Synchronisation ein Prinicpal mit Passwort benötigt. Wodurch die Synchronisation zwischen Master und Slave eine C2C (Client zu Client) Verbindung darstellt. Im herkömmlichen Betrieb wird hier ein Script erstellt, welches anschließend durch einen CronJob in regelmäßigen Abständen ausgeführt wird. Hierfür werden durchgehend dieselben Principals mit dauerhaftem Passwörtern herangezogen. Durch QKD wäre es möglich parallel auf beiden Seiten Principals im Vorrat zu generieren und diese mit Quantenschlüsseln zu versehen, so dass jeder Principal lediglich einmal zur Synchronisation genutzt werden könnte. Anschließend könnten diese gelöscht und durch einen neuen Principal ersetzt werden. So würde die Synchronisation zwischen Master und Slave deutlich geschützter möglich werden.

Kapitel 4

Implementierung des Quanten-Kerberos

Nun werden Möglichkeiten zur Umsetzung der im vorherigen Kapitel angegebenen Konzepte betrachtet. Es werden für dieses Beispiel zwei bereits implementierte Kerberos KDCs angenommen. Falls Bedarf zur Umsetzung eines solchen Systems besteht, bitte ich dieses in einschlägiger Literatur nachzulesen. Für die Darstellung des Cross-Realm werden die Realms „INF.UNIBW.DE“ sowie „CODE.UNIBW.DE“gegeben sein. Die genutzte Datei, die die Schlüssel bereitstellt wird von den ID Quantique Geräten produziert und anschließend von den Steuerkonsolen, die mit einem Unix System laufen, bereitgestellt. Zwingende Voraussetzung ist ein sicherer Kommunikationskanal über den ein Schlüssel ID Vergleich stattfinden kann.

4.1 QKD Principal Trust

Zunächst betrachten wir den herkömmlichen Ablauf der Erstellung eines Cross-Realm Principals. Im ersten Schritt muss sich bei dem kadmin angemeldet werden, die „-r“ Option bezeichnet hier, dass man sich direkt im Realm anmelden möchte als Administrator. Nach der Bestätigung wird das KDC Administrator Passwort zur Verifizierung verlangt. Im kadmin wird der „add_prinicipal“ Befehl ausgeführt, auf dem der Principal Name folgt, der aufgebaut sein muss wie im Beispielcode zu sehen. Anschließend wird die zweimalige Eingabe des Passworts verlangt. Sowie dies geschehen ist, ist der

Principal angelegt und bereit für die Einrichtung des Trusts.
Der Vorgang für die Erstellung der Cross-Realm Principals sieht in Linux Syntax wie folgt aus:
Realm INF.UNIBW.DE:

```
[root@inf ~]# kadmin -r INF.UNIBW.DE
kadmin: add_principal krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE
Enter password for principal "krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE":
Re-enter password for principal "krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE":
Principal "krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE" created.
quit
```

Realm CODE.UNIBW.DE:

```
[root@code ~]# kadmin -r CODE.UNIBW.DE
kadmin: add_principal krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE
Enter password for principal "krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE":
Re-enter password for principal "krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE":
Principal "krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE" created.
quit
```

Die Erstellung des Quanten-Cross-Realm Principals erfolgt durch eine Automatisierung des Vorgangs. Zunächst muss zu dem fremden Realm eine SSH-Verbindung hergestellt werden. Mit Zugang zu beiden Realms, wird folgendes beispielhaftes Programm gestartet, welches hier in Bash realisiert ist. Beim Start wird zunächst die Eingabe des fremden Realm Namens und das kadmin Passworts verlangt für den Zugang zum KDC. Unterdessen liest das Programm den eigenen Domain Namen aus und erstellt mit der Nutzereingabe den fertigen Realm Namen, welcher in einer Text Datei abgespeichert wird für spätere Zwecke. Zusätzlich wird der Quantenschlüssel aus der Datei, die in beiden Realms vorliegt, ausgelesen und in seine Tupel aufgeteilt.

```
#Creating the CR-Principals
read -p "Bitte geben Sie den Namen der fremden Realm ein: " realm2
realm1=$(hostname)
krbtgt="krbtgt/${realm2^^}@${realm1^^}"

#Saving the CR-Principals name for future CronJobs
echo $krbtgt > /usr/local/bin/krbName.txt

#Password prompt for kadmin access
```

```
echo "Bitte geben Sie Ihr kadmin Passwort ein: "
read -s adminpsw

#Reads the key from file
read ckey < /usr/local/bin/QuantumKeys.key
key=`echo  $ckey | awk -v FS=" " '{print $2}'`
id=`echo  $ckey | awk -v FS=" " '{print $1}'`

#Calling of the expect part
./Expect $adminpsw $krbtgt $key $realm1
```

Mit den gewonnenen Daten können anschließend in beiden Realms die Principals parallel angelegt werden. Dies erfolgt durch die Automatisierung mittels Expect und Send Befehlen. Nach erfolgreicher Erstellung, meldet sich das Programm beim kadmin ab und gelangt wieder auf den üblichen Konsolen Bildschirm.

```
#Variable for kadmin password
set adpsw [lindex $argv 0]

#Variable for Principal name
set trust [lindex $argv 1]

#Variable for key
set key [lindex $argv 2]

#Variable for real name, is needed for kadmin login
set realmName [lindex $argv 3]

#Kadmin login and creating of the new CR-Principal.
#The quantum key is used for password.
spawn kadmin -r $realmName
expect "Password"
send -- "$adpsw\r"
send -- "add_principal $trust\r"
expect  "Enter"
send  "$key\r"
expect "Re-enter"
send  "$key\r"
send  "quit\r"
expect eof
```

4.2 CronJob für regelmäßige Trust Erneuerung

Die händische Änderung eines Principal Passworts geschieht wie folgt. Zunächst erfolgt die Anmeldung im kadmin mit dem KDC Passwort. Ist der Nutzer verifiziert kann er mittels des „change_password -keepold“ Befehls dem darauf folgenden Principal ein neues Passwort zuweisen. Die Option weist dem KDC an, das alte Passwort zu behalten. Diese Option macht hauptsächlich bei den krbtgt Principals Sinn, da ansonsten alle aktuell authentifizierten Nutzer und Dienste gezwungen wären, sich mit dem erneuerten Passwort einen neuen Session Key ausstellen lassen zu müssen.
Die Befehlseingabe für diesen Vorgang sieht im Linux Betriebssystem wie folgt aus:
CronJob in Realm INF:

```
kadmin -r INF.UNIBW.DE
change_password -keepold krbtgt/CODE.UNIBW.DE@INF.UNIBW.DE
```

CronJob in Realm CODE:

```
kadmin -r CODE.UNIBW.DE
change_password -keepold krbtgt/INF.UNIBW.DE@CODE.UNIBW.DE
```

Durch die Einrichtung eines CronJobs mit Administrator Berechtigung, lässt sich dieser Vorgang in regelmäßigen Abständen automatisch wiederholen. Hier ein Beispiel eines CronJobs, der alle 12 Stunden um 0 und 12 Uhr das changepw.sh aufruft.

```
crontab -e
0 0,12 * * * /usr/local/bin/changepw.sh (Beipiel Dateipfad)
```

Durch die Einrichtung eines CronJob der mit administrativen Rechten arbeitet, könnte zur gleichen Zeit in den KDCs das Passwort erneuert werden, da es für die Server ohnehin essenziell ist ihre Zeit zu synchronisieren bestünde auch keine Gefahr das eine Abfrage dieses Passwortes geschehen würde außerhalb der Abarbeitung des Auftrags. Zur Umsetzung muss ein Service Principal angelegt werden, der durch die Access Control List (ACL) Zugang zu den kadmin Befehlen erhält. Der CronJob beginnt die Anmeldung mit diesem Principal und der „-kt“ Option, welche angibt, dass man sich als bestimmter Principal bei dem kadmin anmelden möchte, aber anstelle der Passworteingabe die keytab gelesen werden soll. Bevor das Passwort geändert werden kann, wird der Name des benötigten CR-Principals aus der Text Datei ausgelesen, die bei der Erstellung des Principals angelegt wurde. Der neue Schlüssel wird wie bei der Erstellung aus der Quantenschlüsseldatei ausgelesen.

```
#Reads the Principal name from file
read princ < /usr/local/bin/krbName.txt

#Reads key from file
read ckey < /usr/local/bin/QuantumKeys.key
key=`echo  $ckey | awk -v FS=" " '{print $2}'`
id=`echo  $ckey | awk -v FS=" " '{print $1}'`

#Calling of the expect part
./TrustPswChanger $key $princ
```

Wie bei der CR-Principal Erstellung erfolgt die Erneuerung des Schlüssel automatisiert mit Expect und Send Befehlen, die die zuvor gesammelten Daten verwenden um den Trust zu erneuern. Mit dem „change_password“ Befehl wird der neue Schlüssel an den Principal übertragen. Die „-keepold“ Option sorgt dafür, dass das vorige Passwort nicht verworfen wird sondern in einem Zwischenspeicher behalten wird. Dies verhindert, dass Principals, die zurzeit einen Session Key auf Grundlage dieses Passwortes nutzen nicht gezwungen werden sich erneut zu authentifizieren, sondern den Session Key bis zum Ende der Session weiter verwenden können.

```
#Variable of the new key
set key [lindex $argv 0]

#Variable for Principal name
set princ [lindex $argv 2]

#Kadmin login
spawn kadmin.local
sleep 1

#Renewing Trust password
sleep 1
send "cpw -keepold $princ\r"
sleep 1
expect  "Enter"
send  "$key\r"
sleep 1
expect "Re-enter"
send  "$key\r"
```

```
sleep 1
send "quit\r"
expect eof
```

4.3 Random Device umfunktionieren

Die Ersetzung des Pseudo Random Number Generator Session Keys, durch einen Quantenschlüssel könnte für den größten Zuwachs an Sicherheit innerhalb eines Netzes sorgen. Wie in folgendem Quellcode des MIT zur Implementierung des Kerberos Protokolls zu sehen, wird der Schlüssel über /dev/urandom erzeugt.

```
/* This file implements a PRNG module which relies on the
* system's /dev/urandom device.  An OS packager can select
* this module given sufficient confidence in the operating
* system's native PRNG quality.
*/
#include "crypto_int.h"
#define DEVICE "/dev/urandom"
static int fd = -1;
int
k5_prng_init(void) {
        /* Try to open the random device read-write;
        * if that fails,
        * read-only is okay. */
        fd = open(DEVICE, O_RDWR, 0);
        if (fd == -1)
        fd = open(DEVICE, O_RDONLY, 0);
        if (fd == -1)
        return errno;
        return 0;
}
void
k5_prng_cleanup(void) {
        close(fd);
        fd = -1;
}
krb5_error_code KRB5_CALLCONV
```

```
krb5_c_random_add_entropy(krb5_context context,
unsigned int randsource, const krb5_data *indata) {
        krb5_error_code ret;
        ret = krb5int_crypto_init();
        if (ret)
        return ret;
        (void)write(fd, indata->data, indata->length);
        return 0;
}
krb5_error_code KRB5_CALLCONV
krb5_c_random_make_octets(krb5_context context,
                                krb5_data *outdata) {
        char *buf = outdata->data;
        size_t len = outdata->length;
        ssize_t count;
        while (len > 0) {
                count = read(fd, buf, len);
                if (count == 0) /* Not expected
                        * from a random device. */
                return KRB5_CRYPTO_INTERNAL;
                if (count == -1)
                return errno;
                buf += count;
                len -= count;
        }
        return 0;
}
krb5_error_code KRB5_CALLCONV
krb5_c_random_os_entropy(krb5_context context,
                        int strong, int *success) {
        return 0;
}
```

Durch die gezielte Umlenkung des Device wird es möglich, dass dieser die Schlüssel aus den ID Quantique Geräten bezieht. Für diese Änderung des Daemons wird aber ein Linux Device Driver Programming nötig. Da ein solches den Rahmen dieser Arbeit deutlich übersteigen würde, kann dies in zukünftigen Projekten realisiert werden.

Kapitel 5

Fazit

In dieser Arbeit wurde gezeigt, dass mit Quantenschlüsseln neue Möglichkeiten zur Verbesserung der Sicherheit eines Netzwerkes erreicht werden können. Mit der Einführung einer Quantum Communication Infrastructure zwischen zwei Einrichtungen der Universität der Bundeswehr, wird der erste Baustein in einer nachhaltig sicheren Kommunikation in diesem Netzwerk gewährleistet. Durch Erweiterungen an weitere Institute kann dieses Netz sukzessive weiter ausgebaut und in bestehende Netze eingebunden werden. Wie gezeigt wurde, kann das Kerberos Protokoll in einem quantensicheren Netzwerk eingebracht werden. Die Tatsache, dass es ausschließlich auf symmetrischer Verschlüsselung beruht und Tickets anstelle von Passwörtern durch das Netz sendet, machen es hervorragend geeignet für diese Aufgabe. Die bekannten Schwachstellen wurden im Laufe der Zeit mit Patches behoben, oder es existieren bekannte Lösungen, um die Schwachstellen zu umgehen. Da der Quellcode des Protokolls unberührt bleibt, wird eine Einbindung in Netzwerke ohne Quantenschlüssel ebenso möglich bleiben. Da Kerberos die verwendeten Schlüsselformate benutzt, wie die vom QKD geforderten aes-256, lassen sich diese dort leicht verwenden und einbringen. Wie bei der Implementierung gezeigt, lassen sich die Cross-Realm Principals und eine regelmäßige Erneuerung des Trusts in wenigen Schritten realisieren. Einzig wichtige Voraussetzung ist ein sicherer Kanal, der den Abgleich der Schlüssel ID ermöglicht. Für die synchrone Verwendung der Schlüssel sind die Steuerkonsolen an den Servern der jeweiligen Realms verantwortlich.
Mit dem QKD wurde eine Möglichkeit entwickelt, symmetrische Schlüssel sicher an zwei

Parteien zu übermitteln. Für diese Übertragung wird keine vorhergehende asymmetrische Verschlüsselung nötig, um einen sicheren Kanal zu erstellen, sondern wird alleine durch die physikalischen Eigenschaften der Quanten erreicht. Da diese Technik kabelgebunden sowie über die Luft durch Laser genutzt werden kann, bietet sie flexible Einsatzmöglichkeiten in verschiedensten Bereichen. Die Schwächen des QKD werden mit voranschreitender Entwicklung abnehmen. Schon heute ist absehbar, dass interkontinentale Netzwerke eingerichtet werden können, die annehmbare Datenübertragungsraten erreichen werden.
Schlussendlich zeigt sich, dass das Kerberos Protokoll mit dem QKD sinnvoll erweitert und verstärkt werden kann und somit ein quantenresistentes Netzwerk ermöglicht wird.

5.1 Zukünftige Anwendungen

Wie schon in der Implementierung gezeigt, lassen sich die Session Key durch Quantenschlüssel ersetzen, was einen deutlichen Mehrgewinn an Sicherheit bedeutet. Des Weiteren ist es möglich eine Master-Slave Synchronisierung durch das QKD zu bewerkstelligen. Dadurch ist ein kontinuierlicher Abgleich von Master und Slave möglich. Somit sind beide Server durchgehend auf dem aktuellsten Stand der Daten, wodurch eine Erhöhung der Redundanz und Sicherheit erreicht werden kann.

Literatur

[1] MTG AG. *What is a Key Management System?* [Online; Stand 28. Februar 2021]. 2021. URL: https://www.mtg.de/en/crypto-key-management-system/info-key-management-system/.

[2] Red Hat Enterprise. *19.5. Konfigurieren eines Kerberos 5-Servers.* [Online; Stand 11. Februar 2021]. 2021. URL: https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-de-4/s1-kerberos-server.html.

[3] Red Hat Enterprise. *SETTING UP CROSS-REALM KERBEROS TRUSTS.* [Online; Stand 21. Januar 2021]. 2021. URL: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/using_trusts.

[4] ETSI. *Implementation Security of Quantum Cryptography.* [Online; Stand 21. Januar 2021]. 2018. URL: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp27_qkd_imp_sec_FINAL.pdf.

[5] ETSI. *Quantum Safe Cryptography and Security.* [Online; Stand 21. Januar 2021]. 2015. URL: www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf.

[6] Daniel Gottesman u. a. „Security of quantum key distribution with imperfect devices“. In: *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* [Online; Stand 11. Februar 2021]. IEEE. 2004, S. 136. URL: https://arxiv.org/pdf/quant-ph/0212066.pdf.

[7] Network Working Group. *Public Key Cryptography for Initial Authentication in Kerberos (PKINIT).* [Online; Stand 11. Februar 2021]. 2006. URL: https://tools.ietf.org/html/rfc4556#section-1.

[8] Mart Haitjema. „A survey of the prominent quantum key distribution protocols“. In: (2007). URL: https://www.cse.wustl.edu/~jain/cse571-07/ftp/quantum.pdf.

[9] Hoi-Kwong Lo, Marcos Curty und Kiyoshi Tamaki. „Secure quantum key distribution“. In: *Nature Photonics* 8.8 (2014). [Online; Stand 11. Februar 2021], S. 595–604. URL: https://arxiv.org/pdf/1505.05303.pdf.

[10] Andrew Shield/Toshiba Research Europe Ltd. *Performance Limits for Quantum Key Distribution Networks*. [Online; Stand 11. Februar 2021]. 2019. URL: https://www.itu.int/en/ITU-T/Workshops-and-Seminars/2019060507/Documents/Andrew_Shields_Presentation.pdf.

[11] Dipl.-Ing. (FH) Stefan Luber. *Definiton Kerberos*. [Online; Stand 21. Januar 2021]. 2019. URL: https://www.security-insider.de/was-ist-kerberos-a-887891/.

[12] Marco Lucamarini u. a. „Overcoming the rate–distance limit of quantum key distribution without quantum repeaters“. In: *Nature* 557.7705 (2018), S. 400–403.

[13] MIT. *Cross-realm Authentication*. [Online; Stand 29. Januar 2021]. 2021. URL: https://web.mit.edu/kerberos/krb5-1.5/krb5-1.5.4/doc/krb5-admin/Cross_002drealm-Authentication.html.

[14] MIT. *Kerberos: The Network Autentication Protocol*. [Online; Stand 21. Januar 2021]. 2020. URL: http://web.mit.edu/kerberos/www///.

[15] MIT. *The Kerberos Network Authentication Service (V5)*. [Online; Stand 21. Januar 2021]. 2005. URL: https://tools.ietf.org/html/rfc4120#section-1.1.

[16] ID Quantique. *True random number generation exploiting quantum physics*. [Online; Stand 28. Februar 2021]. 2021. URL: https://www.idquantique.com/random-number-generation/overview/.

[17] Masahide Sasaki. „Quantum key distribution and its applications“. In: *IEEE Security & Privacy* 16.5 (2018), S. 42–48.

[18] Valerio Scarani u. a. „The security of practical quantum key distribution“. In: *Reviews of modern physics* 81.3 (2009), S. 1301.

[19] Dipl.-Ing. (FH) Stefan Luber / Peter Schmitz. *Definition Kryptographie*. [Online; Stand 21. Januar 2021]. 2017. URL: https://www.security-insider.de/was-ist-kryptographie-a-642288/.

[20] Sebastian Schreiner. „Freiraumoptische Quantenkryptographie“. [Online; Stand 21. Januar 2021]. Magisterarb. URL: https://xqp.physik.uni-muenchen.de/publications/files/theses_diplom/diplom_schreiner.pdf.

[21] SAS Institute Inc. Stuart J. Rogers. *Kerberos Cross-Realm Authentication: Unraveling the Mysteries*. [Online; Stand 21. Januar 2021]. 2017. URL: https://support.sas.com/resources/papers/proceedings17/SAS0623-2017.pdf.

[22] Jeffrey Uhlmann. „Considerations on Quantum-Based Methods for Communication Security“. In: *CoRR* abs/1808.08591 (2018). arXiv: 1808.08591. URL: https://dblp.org/rec/journals/corr/abs-1808-08591.bib.

[23] Wikipedia. *No-Cloning-Theorem — Wikipedia, Die freie Enzyklopädie*. [Online; Stand 28. Januar 2021]. 2021. URL: https://de.wikipedia.org/w/index.php?title=No-Cloning-Theorem&oldid=207658510.

[24] Wikipedia. *Quantenschlüsselaustausch — Wikipedia, Die freie Enzyklopädie*. [Online; Stand 12. Januar 2021]. 2021. URL: https://de.wikipedia.org/w/index.php?title=Quantenschl%C3%BCsselaustausch&oldid=207558582.

„Hiermit versichere ich, die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, die Zitate ordnungsgemäß gekennzeichnet und keine anderen, als die im Literatur/Schriftenverzeichnis angegebenen Quellen und Hilfsmittel benutzt zu haben.“ Ferner habe ich vom Merkblatt über die Verwendung von studentischen Abschlussarbeiten Kenntnis genommen und räume das einfache Nutzungsrecht an meiner Bachelorarbeit der Universität der Bundeswehr München ein. / nicht ein.

..... (Unterschrift)