# Verteilung von Benutzerzertifikaten auf Mobilgeräte

Gunnar Jacobson

Secardeo GmbH
gunnar.jacobson@secardeo.com

### Zusammenfassung

Zur Ende-zu-Ende Verschlüsselung und digitalen Signatur von E-Mails oder Dateien werden digitale Zertifikate und private Schlüssel am Client benötigt. Ein Benutzer muss auf seinem PC sowie an all seinen Mobilgeräten Zugriff auf seinen privaten Schlüssel sowie die Zertifikate der Partner haben, um E-Mails entschlüsseln oder signieren und verschlüsseln zu können. Damit dies für die IT-Anwender in einer Organisation komfortabel und benutzertransparent erfolgen kann, müssen private Schlüssel und Zertifikate automatisiert und sicher aus einem zentralen Schlüsselarchiv auf alle Geräte verteilt werden. Zur Verschlüsselung müssen zudem die Zertifikate von internen und externen Empfängern aus globalen Zertifikatsverzeichnissen abgerufen und dem Mobilgerät bereitgestellt werden.

# 1 Geschäftlicher Einsatz von Mobilgeräten

## 1.1 Mobile Plattformen

Mobilgeräte (Mobile Devices) wie Smartphones oder Tablets werden zunehmend auch für geschäftliche Zwecke eingesetzt. Sie werden in Unternehmen über ein Mobilgeräteverwaltungssystem (Mobile Device Management System, MDM) zentral verwaltet. Im privaten Bereich dominieren Android-Geräte, jedoch führt im geschäftlichen Einsatz Apple iOS. Der Aktivierungsanteil von iOS wird mit 66% gegenüber 31% von Android angegeben [Good15]. Bei Android Geräten ist Samsung mit über 60% Marktanteil führend. Andere Plattformen wie Blackberry oder Windows Phone spielen dabei eine zunehmend verschwindende Rolle.

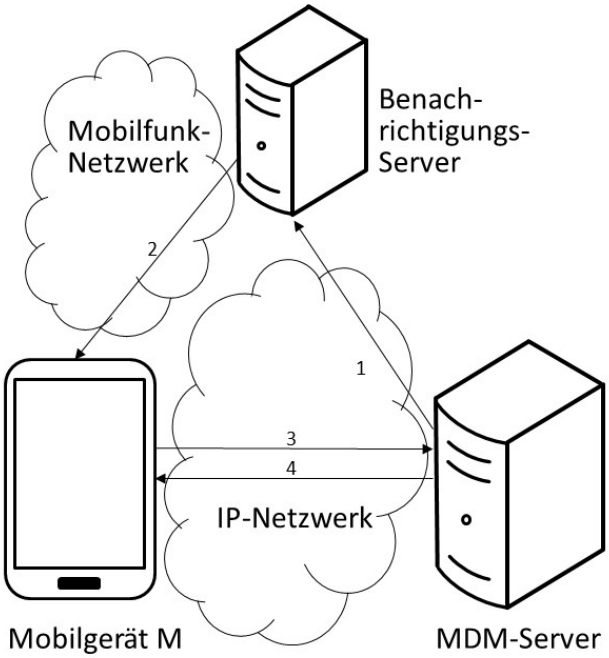
## 1.2 Mobile Device Management

Die zentralisierte Verwaltung von Mobilgeräten erfolgt in Unternehmen über ein Mobilgeräteverwaltungssystem (Mobile Device Management System, MDM), auch als Enterprise Mobility Management (EMM) bezeichnet. Die Verwaltung umfasst die Inventarisierung von Geräten, die Software- und Datenverteilung, sowie den Schutz der Daten auf diesen Geräten. Auch Aufgaben der drahtlosen Verwaltung via Mobilfunk/WLAN gehören dazu. Zur Verwaltung dienen Konfigurationsprofile, die betriebssystemspezifisch sind. Die Verteilung der Konfigurationsprofile vom MDM System auf die betreffenden Geräte erfolgt über MDM Protokolle, die sich je nach Hersteller und Betriebssystem unterscheiden. Diese MDM Protokolle verwenden zum Schutz kryptografische Mechanismen zur Authentisierung und Verschlüsselung. In Abbildung 1 ist der prinzipielle Ablauf abgebildet. Der MDM-Server sendet über das Internet eine Nachricht an einen Benachrichtigungsserver die signalisiert, dass er Daten an das Mobilgerät M sen-

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 98-106.

den möchte. Der Benachrichtigungsserver sendet daraufhin eine Wecknachricht über das Mobilfunknetz an das Mobilgerät M. Das Mobilgerät fragt dann über ein IP Netzwerk beim MDM-Server an. Der MDM-Server sendet dann Anweisungen, beispielsweise zum Löschen von Daten auf dem Gerät oder zum Installieren eines Konfigurationsprofils, an das Mobilgerät. Der genaue Protokollablauf ist abhängig vom Hersteller des Mobilgerätes sowie des MDM Systems. Im Folgenden werden die wichtigsten Protokolle vorgestellt.

**Abb. 1:** Mobile Device Management




Das Open Mobile Alliance (OMA) Device Management (DM) Protocol basiert auf der Synchronization Markup Language (SyncML), die eine Untermenge von XML darstellt [OMA16]. OMA DM spezifiziert die Provisionierung, Gerätekonfiguration, Software Upgrades sowie das Fehlermanagement für Mobilgeräte. Hierfür wird ein Anfrage/Antwort-Protokoll verwendet, das eine gegenseitige Authentisierung umfasst.

Microsoft verwendet in ihrem Mobile Device Management Protocol eine Untermenge des OMA-DM Protokolls [MicM15]. Die damit zu verwaltenden Mobilgeräte müssen zuvor mit dem Mobile Device Enrollment Protocol [MS-MDE] registriert worden sein [MicE15].

Das Microsoft Mobile Device Enrollment Protocol führt die Ausstellung von Zertifikaten gemäß Oasis WS-Trust und den Microsoft WS-Trust X.509v3 Token Enrollment Extensions durch [MicW15]. Damit können Zertifikatsanforderungen in den gängigen Formaten PKCS#10 (RFC2986), PKCS#7 (RFC3852), oder CMC (RFC2797) mittels dem SOAP Protokoll übermittelt werden.

Für Android wurde die Android Device Administration API spezifiziert und unter developer.android.com veröffentlicht [Goog16]. Diese kann von Apps verwendet werden, um Administrationsaufgaben durchzuführen. Mit der Version Android for Works erfolgt eine Trennung privater und geschäftlicher Bereiche auf dem Mobilgerät. Hiermit können auch Profile von einem MDM/EMM System verwaltet werden. Dazu dient eine Work Policy Controller App, wel-

che Anfragen an das MDM System sendet und die verfügbaren Policies über die Device Administration API auf dem Gerät aktiviert. Welches Protokoll zwischen der Work Policy Controller App und dem MDM System abgewickelt wird ist nicht spezifiziert und kann vom MDM Hersteller individuell, beispielsweise gemäß SyncML, implementiert werden.

Apple legt in ihrer „Mobile Device Management Protocol Reference“, für registrierte Benutzer zugänglich unter developer.apple.com, fest, wie iOS Geräte verwaltet werden [Appl11]. Mit dem Protokoll werden Kommandos zur Geräteverwaltung an das Gerät gesendet. Insbesondere können damit von einem MDM System aus Konfigurationsprofile inspiziert, installiert oder gelöscht werden. Dies sind XML-Strukturen im Property List Format (.plist), die beispielsweise zur Konfiguration von E-Mail- oder Netzwerk-Einstellungen dienen und die in der „Apple Configuration Profile Reference“ unter developer.apple.com spezifiziert sind. Das iOS Gerät wird über den Apple Push Notification Service (APNS) „aufgeweckt“, um dann vom MDM System Kommandos abzurufen. Mit dem Kommando InstallProfile wird das Mobilgerät angewiesen das angegebene Profil zu installieren. Die MDM-Nutzdaten (Payload) sind in einem solchen Konfigurationsprofil enthalten. Es kann zu einem Zeitpunkt nur eine MDM Payload auf einem Gerät installiert sein. Diese können vom MDM signiert und für das Gerät mit einem Identity Certificate verschlüsselt werden. Dies geschieht im CMS Format gemäß RFC 5652 [Hous09].

## 1.3 Bedrohungen und Risiken

Eine zunehmende Bedrohung für Unternehmen ist die Industriespionage durch Abgreifen und Mitlesen von Daten. Dabei muss mit mächtigen Angreifer wie Geheimdiensten und professionelle Industriespionen gerechnet werden, die nicht nur in der Lage sind, Überseekabel anzuzapfen sondern ihre Werkzeuge auch innerhalb von Unternehmensnetzwerken zu platzieren. Mobilgeräte sind über unterschiedliche, wechselnde Netzwerke angeschlossen. Daher ist die Ende-zu-Ende Verschlüsselung von Daten eine unternehmenskritische Anforderung.

Ende-zu-Ende Verschlüsselung (end-to-end encryption, E2EE) bedeutet, dass eine Nachricht an seiner Quelle verschlüsselt wird und dass sie nicht entschlüsselt werden kann bis sie ihr endgültiges Ziel erreicht hat, wo sie dann entschlüsselt wird [Shir07].

# 2 Verschlüsseln am Mobilgerät

## 2.1 S/MIME Unterstützung

Das S/MIME Format hat sich zur Verschlüsselung und digitalen Signatur von E-Mails durchgesetzt [Rams04]. Dazu benötigt der Absender einen privaten Schlüssel sowie die öffentlichen Schlüssel der Empfänger. Diese sind in digitalen Zertifikaten nach ITU-T X.509 enthalten, die von einer Zertifizierungsstelle (Certification Authority, CA) ausgestellt werden.

S/MIME wird am PC von Standardanwendungen wie Outlook, Notes oder Thunderbird unterstützt. Die Unterstützung von S/MIME an Mobilgeräten hängt vom Gerätehersteller ab.

Apple iPhones und iPads unter iOS beinhalten eine native E-Mail App, die S/MIME und digitale Zertifikate unterstützt. Die iOS Mail App verschlüsselt für den Benutzer völlig transparent: Sobald ein Zertifikat für einen Empfänger im Gerät verfügbar ist, wird die E-Mail vor dem Versand verschlüsselt.

Das führende Android Derivat von Samsung bietet eine native S/MIME Unterstützung. Bei den meisten anderen Herstellern sucht man hiernach vergeblich. Für solche Androids sowie auch

für iOS gibt es S/MIME Apps anderer Hersteller, die dann zusätzlich installiert werden können. Google bietet mit Android for Work eine Plattform zur Trennung geschäftlicher und privater Daten und Apps. Hier wird auch eine Mail-App mit S/MIME Unterstützung angeboten.

Die von den S/MIME Apps benötigten kryptografischen Operationen zum Ver- und Entschlüsseln und zum digitalen Signieren können mit Hilfe von Software-Bibliotheken des Betriebssystems werden. Alternativ könnte dazu auch spezielle Hardware wie Smartcards genutzt werden. Smartcards werden von den nativen Mail Apps aber nicht unterstützt. Die Anbindung über spezielle mobile Lesegeräte ist teuer und die Nutzung ist auf speziell programmierte Apps beschränkt.

## 2.2 Zertifikate und Schlüsselspeicher

Bei S/MIME können zur Ver- und Entschlüsselung sowie zur digitalen Signatur getrennte Schlüsselpaare und Zertifikate (S/MIME Encryption Certificate, S/MIME Signature Certificate) oder ein gemeinsames (S/MIME Multipurpose) Zertifikat verwendet werden.

Die Krypto-Bibliotheken benötigen Zugriff auf:

1. Die privaten Schlüssel des Benutzers
2. Die zugehörigen digitalen Zertifikate mit den öffentlichen Schlüsseln des Benutzers
3. Die digitalen Verschlüsselungs-Zertifikate der Kommunikationspartner

Dabei werden 1. und 2. zusammen oft auch als digitale Identität des Benutzers, bestehend aus seinem privaten Schlüssel und zugehörigem digitalem Zertifikat bezeichnet. Als Austauschformat wird üblicherweise die PKCS#12 Transfersyntax verwendet [MNP+14].

Die Betriebssysteme stellen für die Speicherung von digitalen Identitäten einen eigenen abgesicherten Speicherbereich zur Verfügung. In Windows werden private Schlüssel und Zertifikate im Windows Certificate Store verwaltet.

Bei Apple iOS steht für die Speicherung von digitalen Identitäten die iOS Keychain und unter Android der Keystore zur Verfügung [Appl14, GooK16].

Eine iOS App besitzt nur Zugriffsrechte auf Keychain-Einträge innerhalb der eigenen Keychain Access Group über die Keychain Services API. Mit der Funktion SecPKCS12Import kann eine iOS App eine digitale Identität zur eigenen Access Group hinzufügen. Auf die Einträge der Apple Keychain Access Group können nur Apple Apps wie iOS Mail oder Safari zugreifen. Solche digitale Identitäten können aus einem PKCS#12 Container unter iOS durch Download mit dem Web-Browser Safari, durch Öffnen als E-Mail-Anhang oder durch den Import eines Konfigurationsprofils installiert werden. Als Dateiendung wird .p12 verwendet und diese kann von keiner anderen App verwendet werden. So installierte digitale Identitäten werden der Apple Keychain Access Group zugeordnet.

Android unterstützt verschiedene Keystores zur Speicherung von digitalen Identitäten. Diese unterscheiden sich je nach Hersteller und weisen unterschiedliche Sicherheitsmerkmale auf [CoRP14]. Unter Android kann jede App die Zugriffsberechtigungen für Schlüssel bei deren Generierung oder deren Import in den Android Keystore persistent festlegen. Es können auch systemweite Schlüssel von Apps über die Android Keychain API verwendet werden. Dazu muss aber der Benutzer zuvor über einen Systemdialog die Berechtigung erteilen.

# 3 Verteilung digitaler Identitäten von Benutzern

## 3.1 Anforderungen

Damit ein Benutzer seine verschlüsselten E-Mails sowohl auf seinem Windows PC, beispielsweise mit Outlook, als auch auf seinem Mobilgerät öffnen kann, müssen dort dieselben privaten Schlüssel verfügbar sein. Ein Benutzer könnte nun selbsttätig seinen privaten Schlüssel aus seinem Windows- Zertifikatsspeicher im PKCS#12-Format exportieren und in seinem Mobilgerät importieren. Dies ist aufwändig und erfordert Fähigkeiten, die man von einem gewöhnlichen Benutzer nicht erwarten kann. In vielen Fällen ist aber sogar die Exportfunktion gesperrt. Daher muss eine automatisierte Verteilung von digitalen Identitäten bereitgestellt werden. Ein Ansatz hierfür ist das zentrale Schlüsselarchiv, das in einer Windows Infrastruktur bereitgestellt werden kann.

## 3.2 Automatisierte Verteilung auf nicht-verwaltete Geräte

Private Schlüssel von Benutzern werden üblicherweise verschlüsselt in einem Schlüsselarchiv gesichert. Bei Windows geschieht dies mittels so genannter Key Recovery Agent (KRA) Zertifikate. Zur Wiederherstellung wird der verschlüsselte Datensatz mit dem privaten KRA-Schlüssel entschlüsselt und kann dann mit einem starken Passwort verschlüsselt in einem PKCS#12 Container bereitgestellt werden. Der verschlüsselte PKCS#12 Container kann per E-Mail an den Benutzer gesendet werden. Dieser empfängt den PKCS#12 Container mit seiner Mail App auf seinem Mobilgerät. Das zum Import erforderliche Passwort wird dem Benutzer in einer S/MIME E-Mail verschlüsselt übermittelt. Der Benutzer kann diese E-Mail mit seinem E-Mailclient auf seinem Windows-PC öffnen und das enthaltene Passwort auf seinem Mobilgerät eingeben. Dieser Prozess kann entweder manuell durch einen KRA-Mitarbeiter oder automatisiert durch eine abgesicherte Serveranwendung (Key Recovery Server) abgewickelt werden, siehe Abbildung 2. Der Import der digitalen Identität auf diesem Weg ist auf allen Geräten des Benutzers möglich mit Ausnahme von Apple Geräten, die durch ein MDM System verwaltet werden.

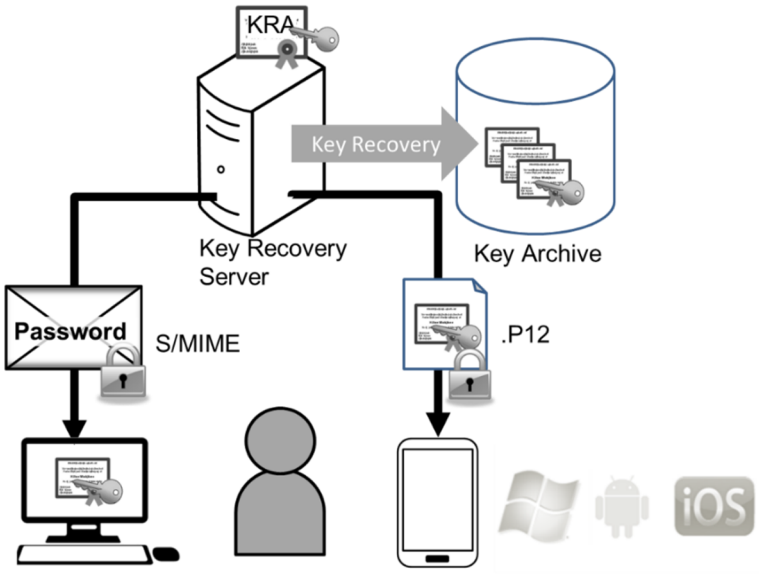



**Abb. 2:** Schlüsselverteilung

## 3.3 Einschränkungen bei MDM-verwalteten iOS Geräten

Bei Apple iOS Geräten, die durch ein MDM System verwaltet werden, ergeben sich grundlegende Einschränkungen, so dass das zuvor beschriebene Verfahren hier nicht verwendet werden kann. Im Apple MDM-Protokoll werden Kommandos zur Geräteverwaltung an das iOS Gerät gesendet. Das Apple MDM-Protokoll verwendet Nutzdaten, die in der Apple Configuration Profile Reference definiert sind. Die Certificate Payload, kann einen PKCS#12 Container sowie das zugehörige Passwort enthalten. Für ein mittels MDM verwaltetes iOS-Gerät muss das Passwort für eine digitale ID über das MDM Protokoll auf das Gerät gelangen. Dazu müssen die PKCS#12 Container sowie die zugehörigen Passwörter auf dem MDM System gespeichert und von diesem übertragen werden. Jeder andere Weg wird durch Apple verhindert. Auch ein manueller Import durch den Benutzer ist nicht möglich.

Ein solches MDM System, in dem PKCS#12-Container mit privaten Schlüsseln samt Passwort aller Mobilanwender eines Unternehmens gespeichert sind, ist ein lohnendes Angriffsziel zur Industriespionage, da der Angreifer anschließend alle abgefangenen verschlüsselten Daten entschlüsseln könnte. Da ein MDM System vom Internet aus erreichbar sein muss, ist es damit einer Vielzahl von Bedrohungen ausgesetzt [RhJW12]. Ein weiteres Risiko stellt der Administrator des MDM Systems dar, welcher sich ebenfalls Zugriff auf alle privaten Schlüssel verschaffen kann. Bei vielen MDM Systemen muss der Administrator die PKCS#12-Container samt Passwörtern der mobilen Benutzer manuell in das MDM System importieren. Das ist zudem mit hohem Aufwand verbunden. Der Einsatz eines MDM Systems zur Verteilung privater Benutzerschlüssel stellt somit ein untragbares Risiko für ein Unternehmen dar.

## 3.4 MDM-Proxy zur sicheren Verteilung digitaler Identitäten

Als Lösung für dieses Sicherheitsproblem wird ein MDM-Proxy eingeführt, vgl. Abbildung 3. Dieser hat die Aufgabe, digitale Identitäten für Benutzer aus dem zentralen Schlüsselarchiv zu holen und an das Mobilgerät zu übermitteln. Hierzu gibt sich der MDM-Proxy im MDM-Protokoll gegenüber dem Mobilgerät als MDM System aus und gegenüber dem MDM System als Mobilgerät. Der MDM Proxy prüft, ob in einem übermittelten Profil Nutzdaten enthalten sind, welche digitale Identitäten eines Benutzers enthalten können, beispielsweise eine Mail- oder Exchange Payload. Um die benötigten digitalen Identitäten des Benutzers in den Nutzdaten zu ergänzen, sendet der MDM-Proxy eine Anfragenachricht an einen Dienst zur Schlüsselwiederherstellung (Key-Recovery Server). Dieser befindet sich üblicherweise in einem anderen, internen Netzwerk welches über eine Firewall abgesichert ist. Der Key-Recovery Server holt aus dem zentralen Schlüsselarchiv den privaten Schlüssel des Benutzers und erzeugt eine mit Passwort geschützte digitale ID im PKCS#12 Format und sendet diese mit dem Passwort über eine, beispielsweise mittels TLS verschlüsselte Verbindung an den MDM-Proxy zurück. Der MDM-Proxy fügt die für die Nutzdaten, beispielsweise Exchange Payload, benötigten digitalen Identitäten inklusive Passwort als Certificate Payload vom Typ com.apple.security.pkcs12 in die PLIST-Struktur des Profils ein. Ferner wird eine Referenz auf die jeweilige Certificate Payload, beispielsweise SMIMEEncryption-CertificateUUID, in die vorhandenen Nutzdaten (z.B. Exchange Payload) eingefügt. Die neu entstandene PLIST-Struktur des Profils wird dann dem Mobilgerät zugesendet. Das Mobilgerät speichert die erhaltenen Konfigurationsdaten, wobei die darin enthaltenen digitalen Identitäten ID-Ui in die System-Key-Chain abgespeichert werden.

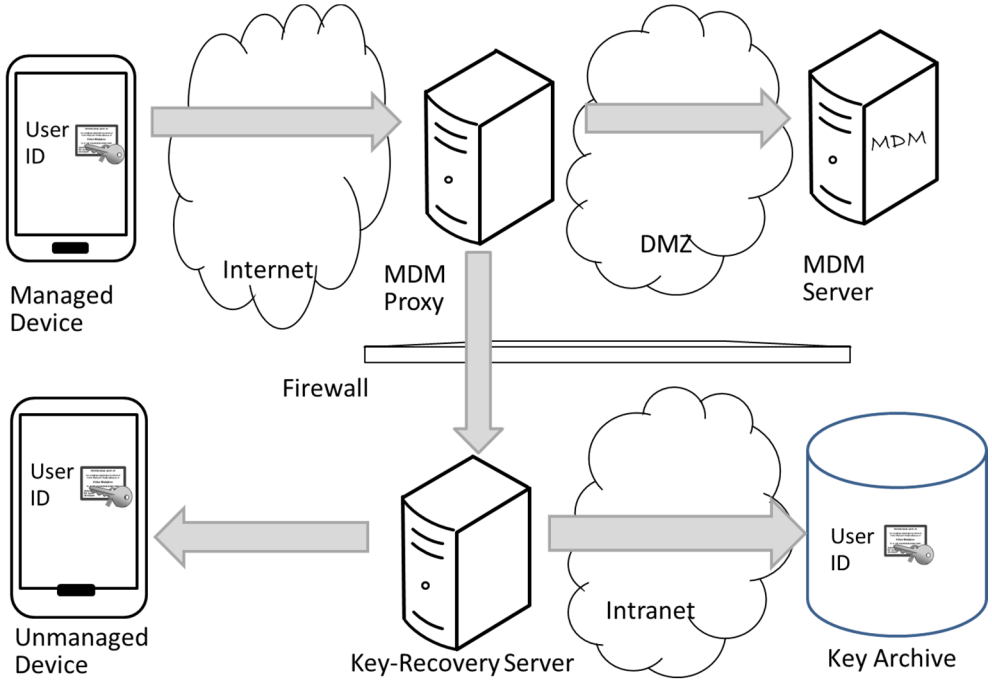



**Abb. 3:** MDM-Proxy

Die Sicherheit kann noch weiter erhöht werden, indem die Exchange Payload samt der Certificate Payload an den Key-Recovery Server übergeben und von diesem mit dem Gerätezertifikat des Mobilgerätes verschlüsselt wird. Mit der so erzeugten EncryptedPayload kann eine Ende-zu-Ende Verschlüsselung der digitalen ID vom Key-Archive bis zum Managed Device erreicht werden.

Mit diesem Verfahren erhält der Benutzer völlig transparent und automatisiert seine privaten Schlüssel und Zertifikate auf seine mittels MDM verwalteten Mobilgeräte und kann beispielsweise mit der iOS Mail App ver- und entschlüsseln sowie digital signieren.

## 3.5 EAS-Proxy zum globalen Abruf von Partnerzertifikaten

### 3.5.1 Zertifikatsverzeichnisse

Unternehmen veröffentlichen die Verschlüsselungszertifikate der Mitarbeiter in einem Zertifikatsverzeichnis, das über LDAP abrufbar ist [Jaco06]. Gängige Desktop-Clients wie Outlook, Notes oder Thunderbird verwenden zur Suche nach Zertifikaten das LDAP Protokoll. Ein Certificate Broker lokalisiert das Verzeichnis, ruft die Zertifikate ab und validiert diese optional [JaNe09]. Wenn für eine Empfängeradresse kein Zertifikat gefunden wird, können für diese auch ad-hoc Zertifikate ausgestellt werden. Der Certificate Broker liefert die Empfängerzertifikate an den Client zurück.

### 3.5.2 Exchange ActiveSync Einschränkungen

Die Anbindung gängiger Mobilgeräte an Serversysteme erfolgt über das Exchange Active-Sync (EAS) Protokoll [MicX15]. Zur Zertifikatssuche werden spezielle Resolve Nachrichten des EAS Protokolls verwendet. Der Server führt die Suche dann im lokalen Verzeichnis, meistens Active Directory, durch und liefert dort gefundene Zertifikate an das Mobilgerät zurück. Damit wird die Zertifikatssuche auf die internen Kollegen im Unternehmen beschränkt.

### 3.5.3 Globaler Zertifikatsabruf

Die Lösung für dieses Problem liefert ein EAS-Proxy zwischen Mobilgerät und Server, vgl. Abbildung 4. Der EAS-Proxy leitet die Resolve Nachrichten an den ActiveSync Server weiter.

Dieser erzeugt eine Resolve-Response Nachricht, welche die lokalen Zertifikate jeweils im Attribut Certificates enthält und sendet die Nachricht an den Proxy. Der Proxy ermittelt die Empfängeradressen, für die kein Zertifikat zurück geliefert wurde und sendet eine LDAP Suchanfrage für diese an der Zertifikats-Broker. Der Zertifikats-Broker leitet die spezifischen LDAP Suchanfragen an die jeweils für einen Adressraum zuständigen externen Directory-Server weiter. Sobald der Zertifikats-Broker alle Antwortnachrichten empfangen hat, sendet er eine LDAP Suchantwort an den EAS-Proxy, die alle gefundenen externen Zertifikate enthält. Der Proxy ergänzt die Resolve-Response Nachricht, welche dann die empfangenen internen und externen Zertifikate im Attribut Certificates enthält und sendet diese an den Client zurück.

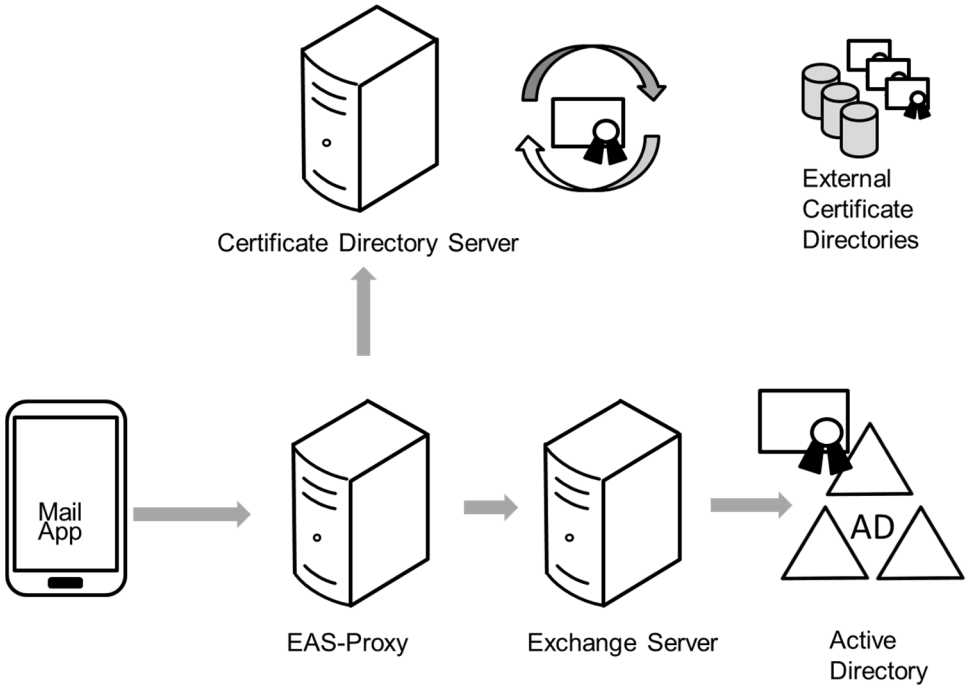



**Abb. 4:** EAS-Proxy

Die Mail App kann nun völlig transparent für den Benutzer an alle internen und externen Empfänger verschlüsseln, für die ein Zertifikat gefunden wurde.

# 4 Ausblick

Es wurde gezeigt, wie die privaten Schlüssel eines Anwenders sowie die Public Key Zertifikate von externen Partnern transparent und sicher auf allen Geräten des Anwenders bereitgestellt werden können. Dort können sie von E-Mail Apps oder anderen Anwendungen zur Gewährleistung von Ende-zu-Ende Sicherheit in der Kommunikation mit beliebigen Partnern verwendet werden. Die somit geschaffene Anwenderfreundlichkeit und Automatisierung von IT-Prozessen zur Schlüsselverteilung sind wichtige Erfolgsfaktoren zur Maximierung der IT-Sicherheit in Organisationen.

## Literatur

[Appl11] Apple: Mobile Device Management Protocol Reference, developer.apple.com, Apple 2011.

[Appl14] Apple: Keychain Services Programming Guide, https://developer.apple.com/library/mac/documentation/Security/Conceptual/keychainServConcepts/01introduction/introduction.html, Apple 2014.

[CoRP14] T. Cooijmans, R. de Ruiter, E. Poll: Analysis of Secure Key Storage Solutions on Android, SPSM '14 Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, 2014.

[Good15] Good Technologies: Mobility Index Report Q3 2015.

[GooK16] Google: „Android Keystore System“,https://developer.android.com/training/articles/keystore.html, Google 2016.

[GooD16] Google: API Guides: Android Device Administration API, https://developer.android.com/guide/topics/admin/device-admin.html, Google 2016.

[Hous09] R. Housley: RFC 5652 Cryptographic Message Syntax (CMS), IETF 2009.

[Jaco06] G. Jacobson: Anbindung von Zertifikatsverzeichnissen. In: D-A-CH Security 2006, syssec (2006).

[JaNe09] G. Jacobson, A. Neppach: Zertifikatsverzeichnisse für „öffentliche“ Public Keys, DuD 7, 2009.

[MicM15] [MS-MDM]: Mobile Device Management Protocol v20151016, https://msdn.microsoft.com/en-us/library/dn392112.aspx, Microsoft 2015.

[MicE15] [MS-MDE]: Mobile Device Enrollment Protocol v20151016, https://msdn.microsoft.com/en-us/library/dn409494.aspx, Microsoft 2015.

[MicW15] [MS-WSTEP]: WS-Trust X.509v3 Token Enrollment Extensions v20151016, https://msdn.microsoft.com/en-us/library/dd340609.aspx, Microsoft 2015.

[MicX15] [MS-EAS]: Exchange Server Protocols, https://msdn.microsoft.com/en-us/library/cc425499(EXCHG.80).aspx, Microsoft 2015.

[MNP+14] K. Moriarty, M. Nystrom, S. Parkinson, A. Rusch, M. Scott: RFC 7292: PKCS#12: Personal Information Exchange Syntax v1.1, IETF 2014.

[OMA16] OMA Device Management V2.0, http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-device-management-v2-0, OMA 2016

[Rams04] B. Ramsdell: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification, Request for Comments: 3851, IETF (2004).

[RhJW12] K. Rhee, W. Jeon, D. Won: Security Requirements of a Mobile Device Management System, International Journal of Security and Its Applications Vol. 6, No. 2, April, 2012.

[Shir07] Shirey: RFC 4949, Internet Security Glossary, Version 2, IETF 2007.