Das Beste aus zwei Welten

DANE und S/MIME

Markus Klause, IT Security Freelancer
https://www.klausecloud.de

DANE in a Nutshell

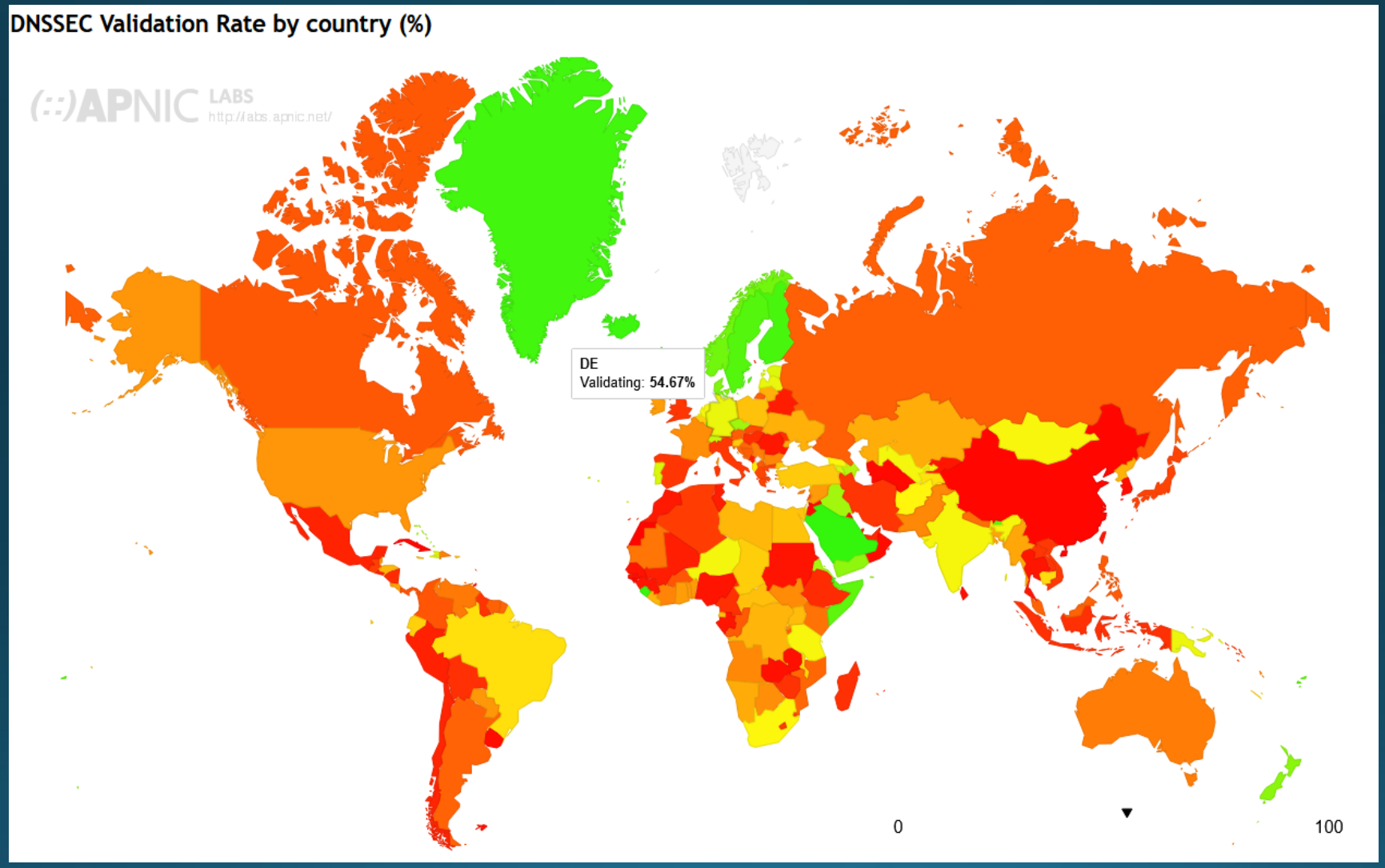
- DNS-based Authentication of Named Entities (DANE) ist ein standardisiertes Verfahren, mit dem sich Zertifikate von Servern zur Verschlüsselung von Webtraffic oder E-Mails per Domain Name System prüfen lassen
- DNS Server können neben IP Adressen also auch den zu einer Email Adresse zugehörigen öffentlichen S/MIME Schlüssel liefern

  → Hierfür wurde (leider) ein eigener RR Typ über die IANA reserviert: **SMIMEA** (https://tools.ietf.org/html/rfc8162)
- Erfordert DNSSEC

DNSSec

- Die DNS Security Extension (DNSSec) ergänzt das Domain Name System (DNS) um einen kryptographisch gesicherten Integritätsschutz, basierend auf asymmetrischer Kryptographie.
- Es schützt jedoch nicht vor Denial of Service und bietet auch keine Vertraulichkeit

Nutzt jemand DNSSEC?



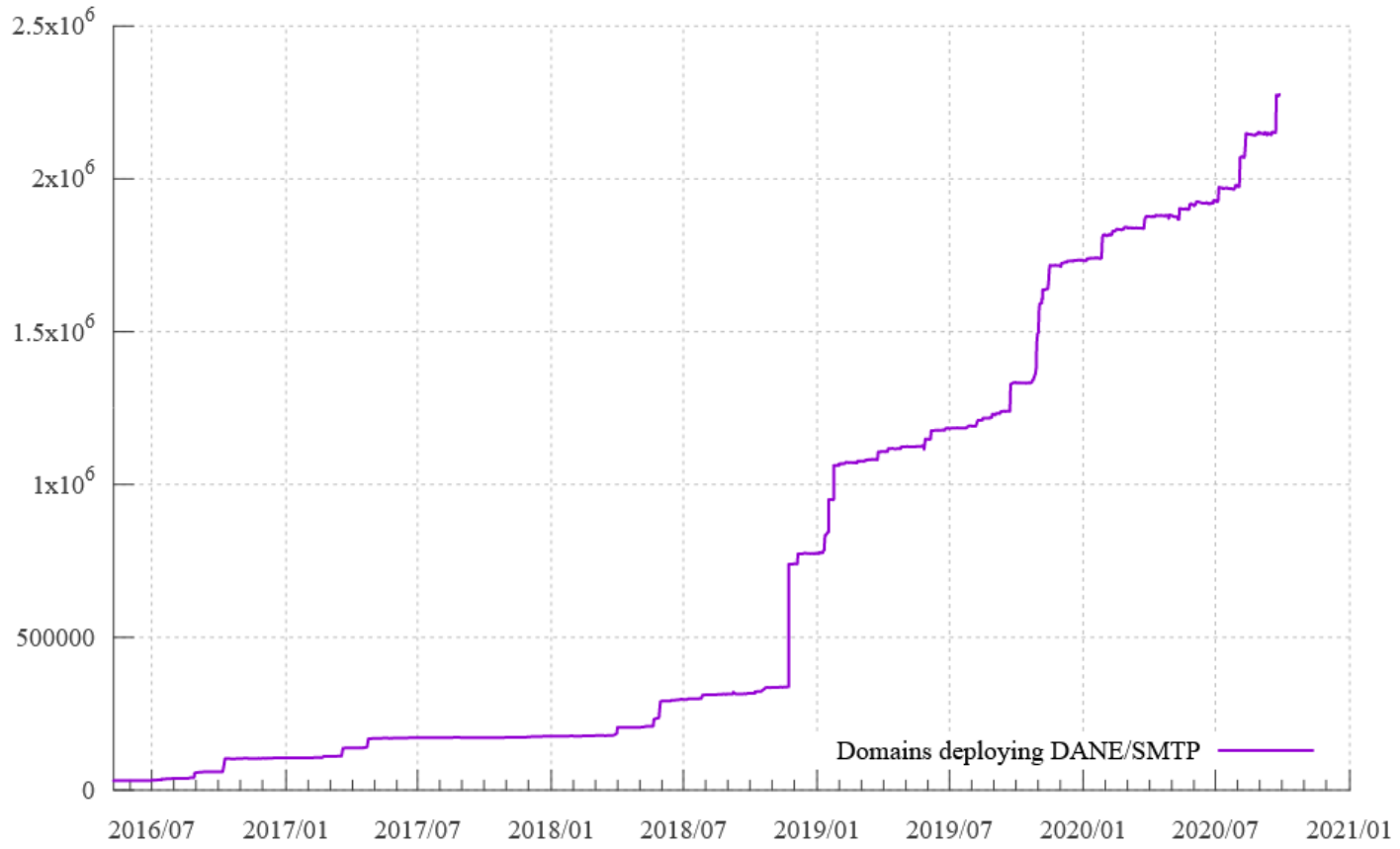
https://stats.labs.apnic.net/dnssec/

Und was ist mit DANE?

DANE Trend graphs

Domains with signed MX and DANE records

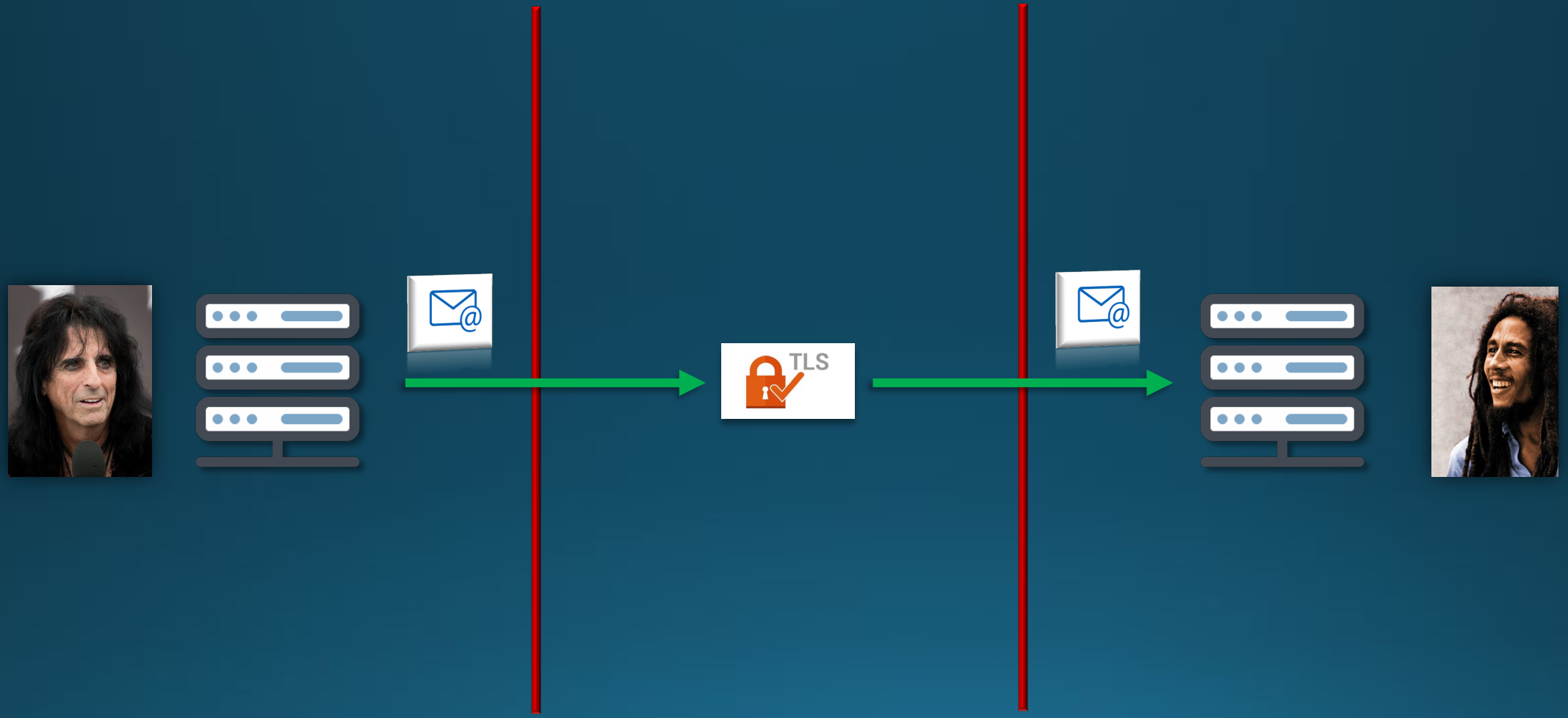
The following graph depicts the number of domains that have deployed DANE/SMTP. Specifically, their zone is signed, their MX records all point to hosts that have DANE TLSA records.

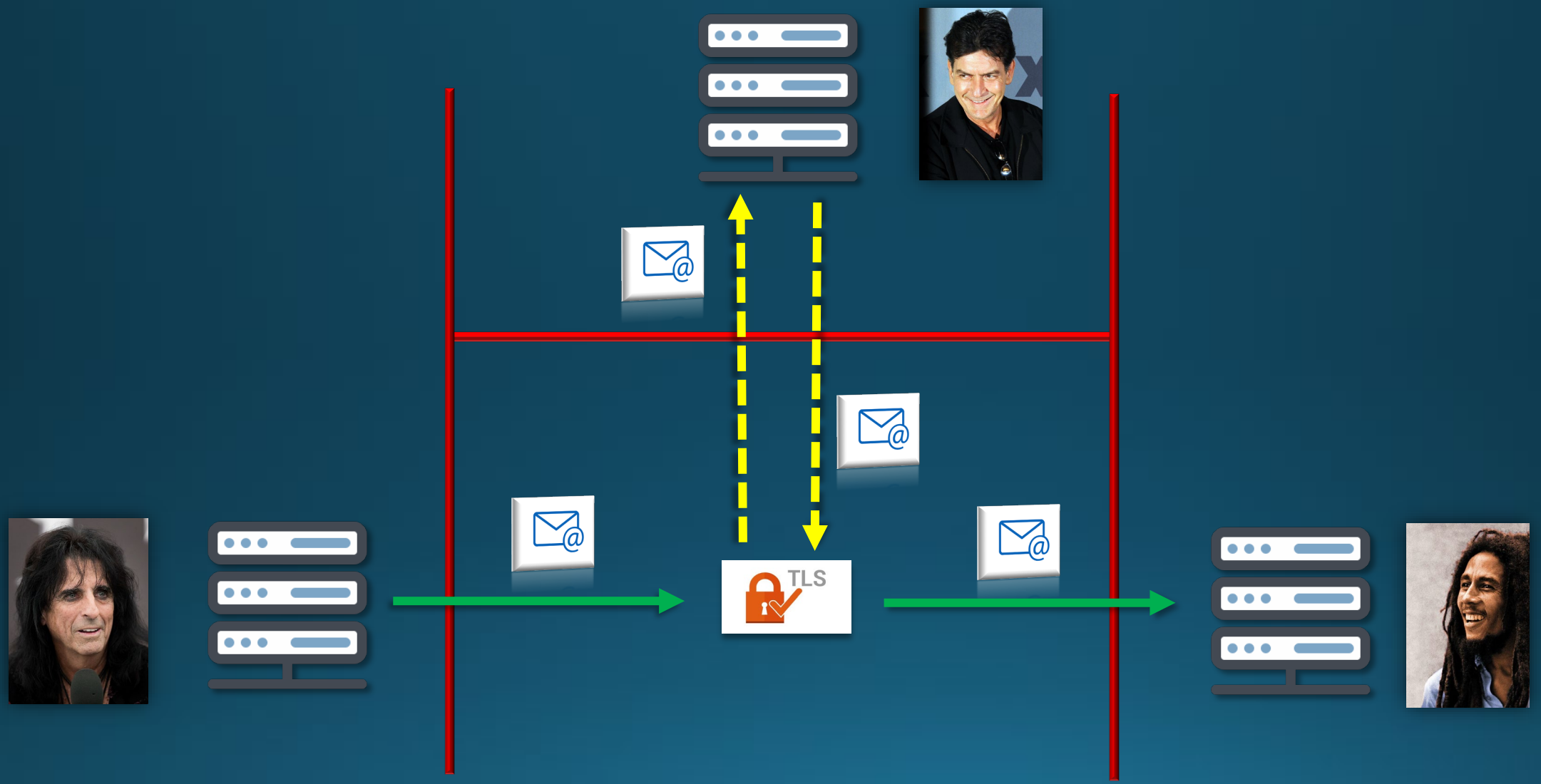


https://stats.dnssec-tools.org/

Motivation DANE

Warum sollte ich mich überhaupt mit DANE beschäftigen?


TLS


TLS

Was ist denn nun DANE?

Vereinfacht gesagt:

Bei DANE vertraut der Empfänger nicht den Certification Authorities, sondern dem Verwalter der Domain und dem **kryptografisch abgesicherten** DNS, also damit letztlich den Schlüsselverwaltern der DNS-Root-Zone.

DANE löst…

….das Problem der mangelnden Transportsicherheit

….das Problem nicht vertrauenswürdiger CA‘s

Motivation S/MIME und DANE

Warum sollte ich mich mit DANE im Zusammenhang mit S/MIME beschäftigen?

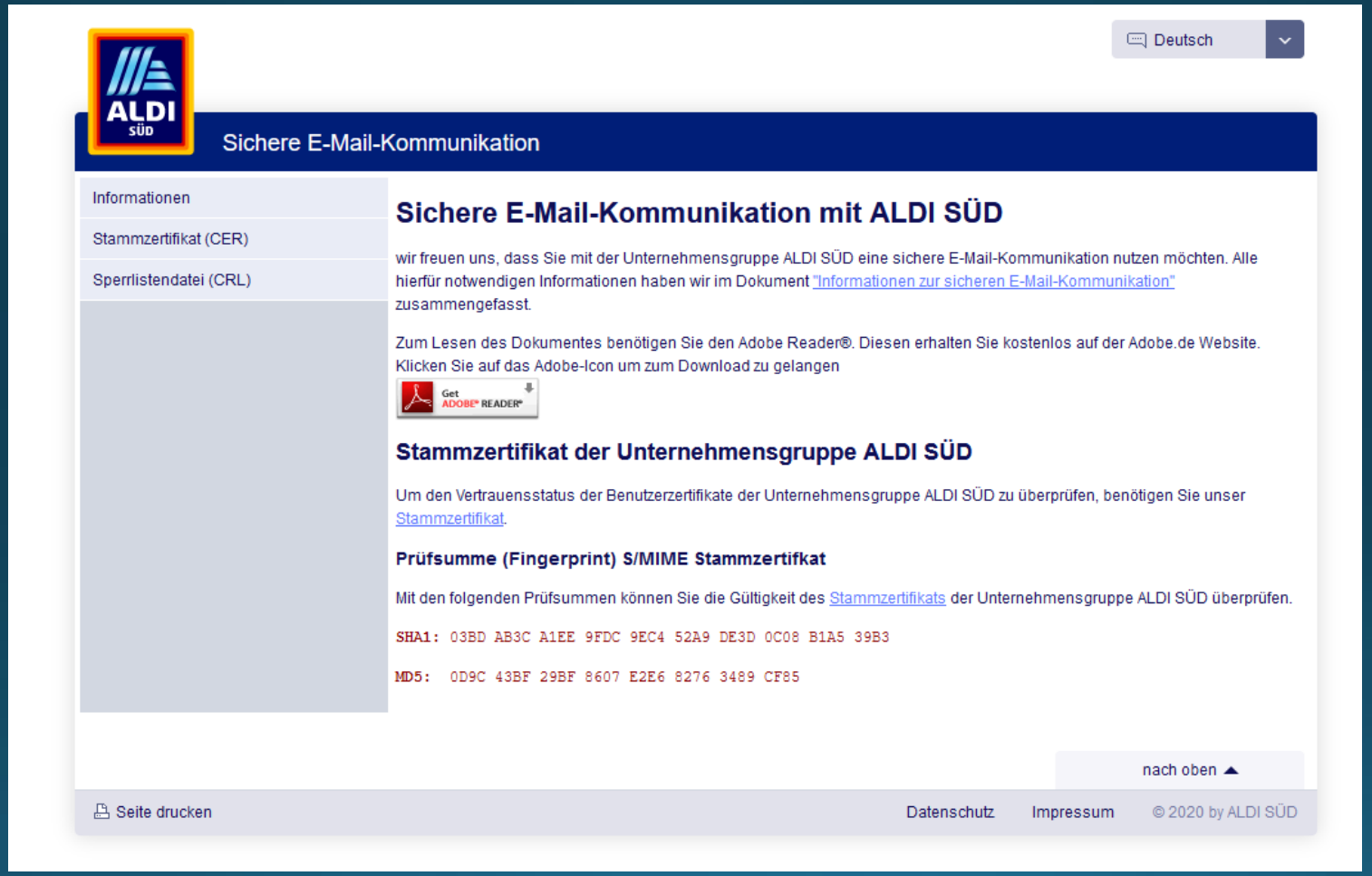
- Einfache Antwort: Das CA System ist kaputt!
- Eine etwas umfangreichere Antwort folgt jetzt…

Das CA Dilemma

- Jede CA (Certification Authority) der Welt kann für beliebige Domains (und damit auch Email Adressen) signierte Schlüssel herausgeben
- Eine CA muss als vertrauenswürdig eingestuft werden, damit ein Automatismus klappt und damit eine Signatur eine Aussagekraft hat. Das wird durch diverse negative Vorfälle immer schwerer
- Das System kostet den CA Betreibern viel Geld
- Zertifikate die über eine standardmäßig vertrauenswürdigen CA kommen, kosten daher auch Geld. Es gibt leider noch kein „LetsEncrypt“ für S/MIME
- Letztlich obliegt es dem Hersteller einer Software/Hardware, welche Zertifizierungsstellen "vertrauenswürdig" sind (schonmal ein altes Android genutzt?)
- Teils können sich die CA’s staatlicher Einflussnahme nicht gänzlich entziehen (Lawful Interception) oder werden gehackt (z.B. DigiNotar).

Dann machen wir das halt einfach selber...!

Beispiel: Aldi Süd



https://www.aldi-sued.com/cert/

Vertrauen sie uns!

- Auf einer Webseite wird ein zu vertrauendes Root Zertifikat angeboten
- Dieser PKI Stelle vertrauen wir mal… ist ja Aldi…
  Diese Aldi PKI könnte aber nun auch Zertifikate für jede andere URL (vielleicht ihr Homebanking) ausstellen.

Das machen die schon nicht!?

→ Sicher nicht vorsätzlich…

Kurzer Ausflug: Emailverschlüsselung

Emailverschlüsselung

Es gilt zunächst zwischen **Verschlüsselung** und **Signierung** zu unterscheiden:

- **Verschlüsseln** stellt sicher, dass Daten ohne privaten Schlüssel nicht gelesen werden können
- **Signieren** stellt sicher, dass Daten nicht modifiziert wurden und hilft die Identität des Absenders zu verifizieren (je nach verwendetem Zertifikatstyp ist die Aussagekraft hier unterschiedlich)

S/MIME oder PGP

- S/MIME nutzt ursprünglich Certificate Authorities (CAs) zum validieren
- Es gibt verschiedene Klassen (Class 0 - Class 4):

Häufig verwendet wird das **Class 1** Zertifikat:
Das verifiziert den Absender nur insoweit, dass der Besitzer des privaten Schlüssels ursprünglich sehr wahrscheinlich Zugriff auf die Mailbox der im Zertifikat genutzten Email Adresse hatte.

S/MIME oder **PGP**

- PGP nutzt das dezentrale „Web of trust“,
  d.h. jeder Teilnehmer kann die Schlüssel anderer signieren.
  (als PGP populär wurde, gab es noch keine Infrastruktur mit Zertifizierungsstellen).

- Es gibt keine übergeordnete Vertrauensstelle

- Praktisch gesehen muss man dem PGP-Key selber vertrauen, idealerweise indem man ihn persönlich oder aus einer vertrauensvollen Quelle erhält

- In öffentlichen PGP Key Servern können die öffentlichen Schlüssel hinterlegt werden. (Dummerweise aber von jedem)

Wann nehme ich was?

Geht es um Verschlüsselung oder Signierung?
(Idealerweise sollte es immer um beides gehen)

Generell sind beide Verfahren gleichwertig.

Bei der Signierung spielt die Frage nach der gewünschten Aussagekraft aber eine Rolle.

Geht es nur um Verschlüsselung spielen die Unterschiede keine so große Rolle mehr. Der Teufel liegt aber im Detail.

Ich persönlich würde immer eher zu S/MIME greifen.

Signieren

Eine Email wird

- Mit dem **privaten** Schlüssel des Absenders signiert (**Versand**)
- Mit dem **öffentlichen** Schlüssel des Absenders validiert (**Empfang**)

Eine gültige Signatur stellt sicher, dass der Inhalt nicht verändert wurde, und dass der Absender Zugriff auf den privaten Schlüssel hat.

Dies kann auch gefälschte Absender entlarven.

Verschlüsseln

Eine Email wird

-Mit dem **öffentlichen** Schlüssel des Empfängers verschlüsselt (**Versand**)
-Mit dem **privaten** Schlüssel des Empfängers entschlüsselt (**Empfang**)

Ohne privaten Schlüssel kann die Email nicht gelesen werden.
Der private Schlüssel passt dabei zu dem öffentlichen Schlüssel.

Starke Vereinfachung

Alice gibt Bob ein offenes Schloss und behält den passenden Schlüssel.

Bob sendet Alice dann Daten und sichert diese mit dem abgeschlossenen Schloss.

Da Alice den Schlüssel hat, kann er nun das Schloss öffnen. Und nur er.

Es sei denn er hat eine gute Flex (oder eher eine Quantenflex)

Warum nutzen immer noch so wenig Firmen S/MIME oder PGP?

„Das ist total kompliziert“

„Unser Admin ist eh schon überlastet“

„Die ganzen Zertifikate sind viel zu teuer“

„Unnötig, wir haben nix zu verbergen“

„TLS reicht doch“

Falsche Vorstellungen von Lösungen und von der Gefahrenlage sowie der möglichen Auswirkungen auf das eigene Unternehmen.

Aber ebenso von den Vorteilen wie vertrauenswürdigere Kommunikation, Spamschutz, eindeutige Zuordnung der Geschäftspartner…

Warum ein S/MIME Zertifikat im DNS?

- Wenn ich signiert sende, muss der Empfänger prüfen ob die Signatur korrekt ist. Dafür benötigt er den öffentlichen Schlüssel. Dieser sollte eigentlich von einer *vertrauensvollen* CA stammen.

→ Stattdessen veröffentliche ich jetzt meinen öffentlichen Schlüssel DNSSEC gesichert über DNS.

Ergebnis: Unabhängigkeit von einer kostenpflichtigen PKI Struktur und ein im ohnehin genutzten System eingebautes Vertrauen

Warum ein S/MIME Zertifikat im DNS?

- Wenn ich verschlüsselt versende, benötige ich den öffentlichen Schlüssel des Empfängers.

  → Bei erstem Kontakt kann der öffentliche Schlüssel

  nun automatisch über DNS angefragt werden.

Ergebnis: Kein Henne-Ei Problem mehr und die Vertrauensstellung ist intrinsisch.

Technische Umsetzung

Wie sieht ein SMIMEA Eintrag aus?

08485C1A4C662F161D312664F3E15FBF5C434EB1549BBEE001D21627._smimecert .klausecloud.de SMIMEA 0 0 1 xxxxxxxxxxxxxxxxxxxxxxxxxx

Aufbau:

Damit die Email Adresse nicht sichtbar im DNS liegt, wird diese als Hashwert hinterlegt

Es wird ein ._smimecert hinzugefügt

Es folgt die Domain

Dann kommt der RR Typ SMIMEA

Anschließend der Payload (TLSA entsprechend)

1. Certificate Usage (https://tools.ietf.org/html/rfc6698#section-7.2)

2. Selector
0 = komplettes Zertifikat
1 = Nur SubjectPublicKeyInfo

3. Matching Type (Details siehe https://tools.ietf.org/html/rfc6698#section-7.4)
0 = Exakte Übereinstimmung
1 = Sha 256 hash
2 = Sha512 Hash

Und das komplette öffentliche Zertifikat (DER SCHLÜSSEL ☺)

Delegation im DNS

Da nach jedem Hash Wert noch ein _smimecert folgt, kann dies einfach delegiert werden.

Daraus ergeben sich einige Vorteile:

- Unterschiedliche Zoneneinstellungen (TTL etc.)

- Unterschiedliche Administratoren

  Bei vielen Systemen lässt sich pro Zone die Administration steuern.
  Das ist wichtig, wenn der "Mail-Admin" die SMIME-Zertifikate publizieren soll aber eben nicht die A-Einträge der Firmendomäne

- Unterschiedliche DNS-Server

  Das hilft, wenn die Stammzone beim Hostingprovider für die Webseite liegt und man die SMIME-Einträge aber auf einem DNS-Server eines anderen Providers (oder bei sich selber) ablegen möchte.

Vorteile

- Mit SMIMEA lassen sich einfach selbstsignierte und dennoch vertrauenswürdige S/MIME-Schlüssel verwenden.
  Jeder von einem Domain-Verwalter im DNS hinterlegter S/MIME-Schlüssel gilt durch die Gesamtheit des Ablaufs als vertrauenswürdig

→ Unabhängigkeit von teuren CA Zertifikaten.

→ Einfache und automatische Verteilung der öffentlichen Schlüssel

Nachteile

- Benötigt zwingend DNSSEC

→ Für DNSSEC gibt es aber ebenfalls sehr gute Gründe

- Direkte Empfänger benötigen einen entsprechenden Email Client

→ Aber nur, wenn kein Gateway verwendet wird.

- Ohne SMIMEA fähiges Gateway ist das ebenfalls nicht nutzbar

→Hierfür gibt es bspw. einen SMIMEA Milter: https://github.com/sys4/smilla

- Durch den neuen RR Typ dauert die Verteilung länger

→ TXT hätte gereicht

- Größere Firmen benötigen ein geeignetes Provisioning, um die Zertifikate zu extrahieren und für alle Benutzer im DNS zu veröffentlichen

Probleme aus der Praxis

- Signaturen für den falschen Zweck ausgestellt:
  Die Annahme, dass ein Absender signierter Emails auch etwas mit einer verschlüsselten Email anfangen kann, muss nicht immer stimmen
- Oftmals eigene Gateways für Verschlüsselung/Signierung welche zusätzlich zum eigentlichen Email Gateway in den Mailflow gesetzt werden
- Der Abruf von kompletten Zertifikaten per DNS erzeugt größere Datenmengen, die u.U. nicht mehr in eine UDP-Antwort passen

→ DNS/TCP nutzen

Diskussion + Fragen

https://www.klausecloud.de

markus@klausecloud.de