Public Key Association

Werner Koch

g10 Code

<wk@gnupg.org>

16. Februar 2006

Public Key Association (PKA) ist a ein vereinfachtes Vertrauensmodell für OpenPGP und selbstsignierte X.509 Zertifikate. Es wird zur Erkennung gefälschter Absenderadressen sowie zu opportunistischer Verschlüsselung verwendet. Dieser Artikel beschreibt die Ideen und Grundlagen und stellt eine Implementierung vor.

1 Übersicht

Für viele Anwendungen sind die üblichen Vertrauensmodelle (Web-of-Trust, Direct, X.509) nicht oder nur schlecht anwendbar. Insbesondere stellt sich bei der Anwendung von OpenPGP die Frage, wie man sinnvoll ein Web-of-Trust aufbauen kann und ob es die notwendige Zuverlässigkeit bieten kann. Der Aufbau einer eigenen CA ist zwar firmenintern möglich, bedeutet aber immer noch einen hohen organisatorischen Aufwand und wird komplizierter bei der Kommunikation mit externen Benutzern.

Oft stellt sich ein Kryptogateway als einzige effiziente Lösung dar, da es eine transparente Migration hin zu verschlüsselter Kommunikation ermöglicht. Selbstverständlich sollte die Schlüsselverwaltung so einfach wie möglich geschehen — nach Möglichkeit ohne jeden manuellen Eingriff.

Das hier vorgeschlagene — und in GnuPG bereits implementierte — Verfahren delegiert die Entscheidung, ob einem Schlüssel vertraut werden kann an das DNS. Standard DNS ist selbst zwar nicht sicher; es kann aber erwartet werden, daß DNSSEC sich in den nächsten Jahren mehr und mehr etabliert und so nach und nach eine sichere Infrastruktur zur Namensauflösung aufgebaut wird. PKA partizipiert dann direkt daran.

Desweiteren kann PKA benutzt werden, um opportunistische Verschlüsselung, also ohne explizites Wissen um die Verschlüsselung, auf dem Desktop zu erreichen. Dies wird erreicht durch die Anbindung von Schlüsseln an Email-Adressen.

Konzeptionell basiert PKA auf 2 Teilen: Einer in jeder signierten Mail mitgesendeten Information, von welcher Mail-Adresse die jeweilige Nachricht stammt; sowie besonderen DNS Records, die eine Verbindung zwischen Mail-Adresse und Schlüssel sowie zu dem Fingerprint des Schlüssels schaffen.

2 Verschlüsselung und Schlüsselverwaltung

Traditionell wird häufig versucht, Verschlüsselung und digitale Signatur mit der Verwaltung und Validierung der benutzten Schlüssel in einen Topf zu werfen. Diese fehlende Trennung unterschiedlicher Bereiche führt immer wieder zu fehlerhafter Anwendung und zu komplizierten Spezifikationen.

Hier ist vor allem X.509, also die Beschreibung der von S/MIME verwendeten Schlüssel, zu nennen. In diesem Standard wird von einer untrennbaren Verbindung von Public Key Infrastruktur (PKI) und dem Aufbau der Schlüssel ausgegangen. Ein Standard[1] beschreibt den technischen Aufbau der Schlüssel (Zertifikate) als auch deren Verhältniss zu anderen Schlüsseln sowie organisatorische Aufgaben wie CRLs. Dies ist recht unflexibel und führte dazu, daß aus praktischen Erwägungen sehr häufig sebstsignierte Zertifikate benutzt werden — obgleich dies von X.509 nicht vorgesehen ist.

Um sich diese Probleme zu ersparen, werden im OpenPGP Standard zwar technische Möglichkeiten zur Implementierung einer PKI definiert, der Aufbau dieser PKI is aber aus gutem Grund nicht festgelegt und kann anwendungsspezifisch definiert werden. Es ist z.B. auch möglich, den Endbenutzer die Art der verwendeten Infrastuktur selbst festlegen zu lassen (z.B. Verwendung des Web of Trust). Die Kommunikationsfähigkeit mit anderen Benutzern ist dadurch nicht beeinträchtigt.

Die Vorherrschaft von X.509 in vielen Gebieten, hat allerdings auch das Konzept anderer Protokolle stark beeinflusst. Die Vermischung von Schlüsseln und diese begleitende Infrastruktur ist heute leider gang und gebe.

Es wird nun ein System beschrieben, welches eine Validierung von Schlüsseln ermöglicht und dabei strikt zwischen der Infrastruktur und dem eigentlichen Signaturverfahren unterscheidet.

3 PKA

Die Idee von Public Key Association (PKA) ist recht trivial: Statt eine aufwendige Public Key Infrastructure (PKI) neu aufzubauen oder komplizierte hybride Verfahren wie DKIM zu verwenden, werden Standardprotokolle benutzt und lediglich einen neue Art der Verwendung geschaffen.

Alles was gemacht werden muß um die Absenderangabe einer Mail zu prüfen, ist eine Verbindung zwischen eben dieser Absenderangabe und einem Schlüssel für ein Public Key Verfahren zu definieren.

Der traditionelle Ansatz einer PKI funktioniert bestenfalls im kleinen Rahmen. Deswegen ist nach einer anderen Möglichkeit zu suchen, die eventuell nicht so sicher wie eine theoretisch sehr ausgebuffte PKI ist, aber hinreichend sicher um gefälschte Absenderangaben zu erkennen.

Welche bestehende Infrastruktur kann nun verwendet werden? Es hat sich gezeugt, daß DNS über seinen ursprünglichen Zweck hinaus weitere Aufgaben übernehmen kann. Hiermit sit nicht der Mißbrauch als Werbeträger gemeint, sondern die Verwendung als replizierte und verteilte Datenbank. Durch eine simple Ausnutzung von DNS konnten sich RBLs (Realtime Blacklists) schnell etablieren ohne

[1] In Realität sind es sogar mehrere Varianten eines Standards; „Profile“ genannt.

erst eine neue Infrastruktur aufbauen zu müßen. Auch ENUM macht hiervon direkten Gebrauch.

Zusammenfassend kann gesagt werden:

- DNS ist immer und überall verfügbar.
- DNS wird aktiv für viele Zwecke benutzt.
- Es existiert ein breit gestreutes und gutes Wissen um seinen Aufbau und die Organisation dahinter.
- DNS ist sowieso ein Single Point of Failure; ein Ausfall des DNS legt schon heute das gesamte Internet lahm. Da Email Transport sowieso den Zugriff auf das DNS verlangt (MX Records), verschärft sich das Problem durch PKA nicht.
- DNS wird eines schönen Tages auch gesichert sein; d.h. garantiert authentische Aussagen treffen können. Damit wird sich dann auch automatisch die Sicherheit von PKA erhöhen.

Aus diesem Grund **verbindet PKA Schlüssel mit dem DNS**.

4 Mit PKA die Authentizitat einer Mail prüfen

Um zu zeigen wie das System funktioniert, werden wir es jetzt anhand eines Beispiels erläutern.

Wir nehmen an, daß alice@example.net eine Nachricht an bob@acme.com senden möchte. Da sie Bob gar nicht kennt und die zu sendene Nachricht auch nicht als vertraulich ansieht, ist sie an einer Verschlüsselung oder digitalen Signatur der Nachricht nicht wirklich interessiert. Das PKA Verfahren ist für sie vollkommen transparent implementiert und bedarf keines Eingriff von ihrer Seite. Zur Beschreibung gehen wir allerdings davon aus, daß Alice alle Aktionen manuell durchführt.

Alice verfasst die Nachricht und setzt in die From: Zeile der Nachricht Ihre eigene Adresse ein. Die Nachricht mag so aussehen:

```
From:    "Allice Wonder" <alice@example.net>
To:      bob@acme.com
Subject: Anfrage

Lieber Bob,

Ich bin an Ihren Produkten zur Absicherung unserer Emails
interessiert.  Bitte übersenden Sie mir Ihre aktuelle Preisliste.

Viele Grüße,

   Alice
```

Um diese Mail zu versenden wird Alice sie nun mit Ihrem Schlüssel signieren. Diese Signatur hat eine Besonderheit: Neben dem Text wird die Signatur auch eine Information enthalten, wer diese erstellt hat. Die Verwendung der im Schlüssel enthaltenen Emailadresse ist hier nicht ausreichend, da Alice über mehrere Adressen

verfügt und je nach Aufgabe eine andere Absenderangabe benutzt (z.B. könnte sie als Externe für verschiedene Firmen arbeiten und möchte dies in Ihrem Schlüssel nicht direkt ausdrücken).

Hier kommt PKA ins Spiel: Mittels einer sogenannter Notation (OpenPGP) oder einem SignedAttribute (X.509) wird die verwendete Adresse mit unterschrieben. Um beim Beispiel OpenPGP zu bleiben, wird die Notation

```
pka-address@gnupg.org=alice@example.net
```

verwendet. Die Nachricht wird dann wie üblich versendet. Im Grunde sind die Modifikationen also auf Senderseite sehr gering.

Wenn Bob nun diese Mail erhält, so wird er feststellen, daß diese digital signiert ist. Wahrscheinlich hat er den Schlüssel zur Überprüfung der Signatur noch nicht. Er könnte Ihn dann (bei OpenPGP) von einem Keyserver beziehen und die Integrität der Mail so überprüfen. Was er nicht kann, ist feststellen, ob alice@example.net wirklich diese Mail geschrieben hat. Denn er kennt Alice nicht und hat auch keine PKI Struktur wie ein Web of Trust zur Verfügung um eine Aussage treffen zu können. Was er also macht ist, die Notation aus der Signatur zu betrachten und damit eine DNS Anfrage zu erstellen. Er wandelt den Wert der Notation dazu um in

```
alice._pka.example.net
```

und sucht damit einen TXT Record im DNS. Er wird dort diesen Record finden:

```
$ host -t txt alice._pka.example.net
alice._pka.example.net text \
  "v=pka1;fpr=47[...]A907;uri=finger:alice@example.org"
```

Der wichtige Teil hier ist der „fpr“ Teil. Bob wird den dort angegebenen Fingerprint mit dem des zur Überprüfung der Signatur verwendeten Schlüssels abgleichen. Wenn diese übereinstimmen, hat er gezeigt, daß Alice in der Tat eine Adresse in der Domain example.net ist. Denn nur dem Eigentümer dieser Domain sollte es möglich sein, Records in dieser Zone einzutragen.

Sollte der Fingerprint aber nicht übereinstimmen, kann davon ausgegangen werden, daß die Absenderangabe gefälscht ist. Dasselbe gilt wenn die Signaturprüfung negativ ausfällt.

Ist eine TXT Record nicht vorhanden, so kann keine Aussage getroffen werden.

Interessant aber optional ist der „uri“ Parameter des TXT Records. Falls dieser vorhanden ist, kann er benutzt werden um den Schlüssel zur Überprüfung der Mail selbst zu finden. In diesem Fall wird Bob natürlich zuerst die DNS Anfrage durchführen und erst dann die Signatur überprüfen. „uri“ soll den kanonische Speicherplatz des Schlüssels angeben; in unserem Beispiel eine Finger Abfrage an den Server example.org.

5 Vorläufige PKA Spezifikation

Dieser Abschnitt dient als Spezifikation der aktuellen PKA Implementierung. Die Verwendung von DNS TXT Records ist nicht optimal aber akzeptabel bis einige

Erfahrung mit dem Verfahren gesammelt werden konnte. Es ist angestrebt, später einen eigenen Resourcetyp zu definieren.

Für jede Mailadresse, die legitim Mail versenden darf, ist ein DNS Eintrag notwendig. Sinnvollerverweise wird hierzu eine eigene Zone delegiert:

```
$ORIGIN _pka.example.net.
@                   IN  SOA  ns1.example.net. ([...])
                    [...]
mailer-daemon       IN  TXT "v=pka1;fpr=79[...]7B"
alice               IN  TXT "v=pka1;fpr=F1[...]02"
alice.wonderland    IN  TXT "v=pka1;fpr=F1[...]02"
bob                 IN  TXT "v=pka1;fpr=B3[...]EA"
```

In diesem Beispiel sind 4 Adressen angelegt wobei zwei auf denselben Fingerprint zeigen (alice und alice.wonderland). Dies ist immer dann notwendig, wenn unter mehreren Absenderangaben Mails gesendet werden. `mailer-daemon` ist eine besondere Adresse: Falls lediglich die Überprüfung der Domain gewünscht ist, sollte dieser Name als PKA-Adresse benutzt werden. Ob dies azeptabel ist, ist eine lokale Policy Einstellung.

Optional kann noch eine Referenz auf den Schlüssel selbst im PKA Record untergebracht werden. Dies ist als drittes Key-Value Paar anzugeben. Beispiel:

```
alice  IN  TXT "v=pka1;fpr=F1[...]02;uri=finger:alice@example.net"
```

Der URI soll auf einen Ort verweisen, von dem er öffentliche Schlüssel direkt angefordert werden kann. Dies kann z.B. mittels des Kommandos "gpg –fetch-keys URL" geschehen.

Jede zu sendende Mail ist mit einer digitalen Signatur zu versehen wobei hier die Absenderangabe und der Body der Mail von der Signatur abgedeckt werden. Bei allen heutigen Signaturverfahren wird lediglich der Body signiert[2]. Als Absenderangabe wird lediglich die eigentliche Adresse benutzt (d.h. `local-part '@' domain`) und je nach Signaturverfahren in den zu signierenden Text mit eingebaut:

OpenPGP Hier wird das Konstrukt *Notation Data* verwendet (RFC2440, 5.2.3.15). Das erste Oktet der *flags* ist auf 0x80 zu setzen, `pka-address@gnupg.org` ist als *name* zu verwenden und die Absenderadresse (z.B. alice@example.net) als *value*. Mit gpg kann dies durch folgende Option erreicht werden:

```
--sig-notation "pka-address@gnupg.org=alice@example.net"
```

S/MIME Hier wird ein *SignedAttribute* verwendet.
Es ist durch die OID `1.3.6.1.4.1.11591.2.1.1` zu identifizieren und nimmt als Wert einen *IA5String* mit der Absenderadresse auf. Hier ein Beispiel als informelles ASN.1:

```
SEQUENCE {
  OBJECT IDENTIFIER
    pkaAddress (1 3 6 1 4 1 11591 2 1 1)
  SET {
    IA5STRING "alice@example.net"
```

[2] eine Signierung der Mail-Header ist nicht möglich, da diese sich während des Transports ändern können

```
    }
}
```

Der Empfänger einer Mail soll diese Signatur prüfen. Fehlt der entsprechende Schlüssel so kann im DNS nachgeschlagen werden ob im PKA Record der optionale URI Parameter vorhanden ist und auf diese Weise der Schlüssel importiert werden.

Ist die Signaturprüfung erfolgreich, so ist zu prüfen ob der Fingerprint es verwendeten Schlüssels dem Fingerprint in PKA Record entspricht. Desweiteren ist zu prüfen, ob die Absenderangabe im Mail Header mit der Angabe der PKA Adresse in der Signatur selbst übereinstimmt. Treffen beide Kriterien zu, so ist die Mail authentisch. Trifft dies nicht zu und ist ein PKA Record vorhanden, so handelt es sich um eine Mail mit gefälschter Absenderangabe.

6 Unterschied zu anderen Vorschlägen

Die Entwicklung von PKA geht auf die Entstehung des DKIM Internet Draft[3] zurück. Der dort gewählte Ansatz wurde von einigen mit der Materie befassten Entwicklern als überflüssig, fehlerhaft und allgemein als Rückschritt empfunden. Insbesondere werden bei DKIM alle Erfahrungen, die bei der Entwicklung anderer Email Sicherheitsprotokolle (OpenPGP, S/MIME) in der Vergangenheit gemacht wurden schlichweg ignoriert.

Hier ist unter anderem die Kanoninisierung der Nachricht zur Signaturerzeugung und -prüfung zu nennen: Obwohl OpenPGP hier sehr einfache Regeln hat (Abschneiden von „trailing white-spaces“ und Zeilenendekonvertierungen), ist dies heute noch eins der häufigsten Probleme, die neue Implementierungen haben. DKIM geht dann aber sogar dazu über, MIME Strukturen besonders zu behandlen, was sehr schwierig zu implementieren ist und in einer Spezifikation nur schwer zu fassen. Es konnte sogar schnell gezeigt werden, daß das von DKIM verwendete Verfahren unterlaufen werden kann und eine veränderter Inhalt einer ansonsten gültigen MIME Mail aufgeprägt werden kann.

Desweiteren verhindert DKIM eine Implemetierung in einer Pipeline (streaming) da hier zuerst die Signatur gesendet wird und dann erst die signierten Daten. Dies war eins der gravierendsten Probleme von PGP, die mit OpenPGP vor 8 Jahren gelöst wurden. Dies heute wieder einführen zu wollen zeugt von einer gewaltigen Portion Ignoranz.

Neben diesen technischen Fehlern hat DKIM, wie die meisten anderen MASS Verfahren, die konzeptionelle Schwäche, Signaturverfahren und Schlüsselverwaltung nicht zu trennen. Insbesondere wird hier ein neues Signaturverfahren und eine eng damit verbundene Schlüsselverwaltung vorgeschlagen. Beides sind aber Bereiche die sich zwar gegenseitig bedingen aber unabhangig betrachtet werden sollten. Andere etablierte Protokolle trennen diese Schichten (z.B. SMTP (RFC821) für Transport und RFC822 für den Inhalt einer Mail oder HTTP und HTML).

PKA dagegen setzt auf etablierte Signaturverfahren und definiert lediglich ein neues Vertrauensmodell.

[3] draft-allman-dkim-base-00.txt

Das Argument, die vorhandenen Signaturverfahren haben sich in der Praxis nicht durchgesetzt und deswegen ist ein neues Verfahren zu definieren, ist nicht sichhaltig: Auch DKIM, implementiert direkt in den Mailprogrammen des Benuzters, würde sich nicht durchsetzen — es ist aber auch gar nicht drauf ausgerichtet, sondern soll an zentraler Stelle automatisch und transparent funktionieren.

Nun, dies kann mit den bestehenden und ausgereiften Protokollen genauso gut und sogar wesentlich besser implementiert werden. Es existiert eine Vielzahl von Implementierungen die in einer Serverimplementierung von PKA benutzt werden können. Es sind lediglich geringe Änderungen notwending, wie wir im nächsten Abschnitt darlegen werden.

7 Implementierung für OpenPGP

GnuPG wird ab der Version 1.4.3 PKA implementieren. Im Subversion Repository ist PKA seit Ende 2005 implementiert (übrigens exakt 8 Jahre nach der ersten Veröffentlichung von GnuPG). Der Release Candidate 1.4.3rc1 implementiert PKA ebenfalls.

Obwohl serverbasierte Implementierungen interessant wären, ergab sich im Zuge der Neuportierung von Sylpheed-Claws auf Windows, die Gelegenheit PKA dort zu implementieren. Sylpheed (und auch die Variante Sylpheed-Claws) benutzt die Bibliothek GPGME um auf GnuPG zuzugreifen.

GPGME wurde desswegen um PKA Funktionen erweitert. Hierzu wurde die Sruktur mit den Informationen zu einer Signatur um die Felder `pka_trust` und `pka_address` erweitert. Dies ermöglicht eine Beurteilung der PKA Prüfung unabhängig von anderen Vertrauensmodellen.

Die Implemtierung der PKA Funktionalität wurde im Plugin `pgpcore` und dort in der Datei `sgpgme.c` vorgenommen:

```
272     if (sig->status != GPG_ERR_BAD_SIGNATURE) {
273           gint j = 1;
274           user = user ? user->next : NULL;
275           while (user != NULL) {
276                   g_string_append_printf(siginfo,
277                           _("                aka \"%s\"\n"),
278                           user->uid);
279                   j++;
280                   user = user->next;
281           }
282           g_string_append_printf(siginfo,
283                   _("Primary key fingerprint: %s\n"),
284                           sig ? sig->fpr: "?");
285   #ifdef HAVE_GPGME_PKA_TRUST
286           if (sig->pka_trust == 1 && sig->pka_address) {
287                   g_string_append_printf(siginfo,
288                      _("WARNING: Signer's address \"%s\" "
289                         "does not match DNS entry\n"),
290                      sig->pka_address);
291           }
292           else if (sig->pka_trust == 2 && sig->pka_address) {
293                   g_string_append_printf(siginfo,
294                      _("Verified signer's address is \"%s\"\n"),
```

```
295                     sig->pka_address);
296                   /* FIXME: Compare the address to the
297                    * From: address.  */
298           }
299   #endif /*HAVE_GPGME_PKA_TRUST*/
300     }
301     g_string_append(siginfo, "\n");
302     i++;
303     sig = sig->next;
```

Wie man sehen kann, ist die Änderung minimal und auch die wichtige aber noch fehlende Prüfung gegen die Absenderadresse wird nicht mehr als 5 Zeilen Code benötigen.[4]

8 Implementierung für S/MIME

Eine Implementierung für S/MIME existiert momentan noch nicht. Es ist allerdings beabsichgtigt, dies in Bälde für GnuPG 1.9 durchzuführen.

S/MIME basiert auf CMS (Cryptographic Message Syntax — vormals unter PKCS#7 bekannt) welches wiederunm X.509 Zertifikate benutzt. Um die Adresse des Absenders in einer CMS Nachricht unterzubringen, ist eine OID festgelegt worden. Diese bezeichnet in einem SignedAttribute einen IA5String mit der Adresse.

9 Anwendungen

Neben der direkten Unterstützung in Mailprogrammen bietet es sich an, PKA zentral in einem Gateway zu implementieren. Dort können Mails direkt mit einer Signatur versehen werden bzw. eine PKA Prüfung durchgeführt werden. Mit einem zusätzlichen Header in der Mail kann auch festgelegt werdern, daß es sich um eine automatische generierte Signatur handelt, die von einem Gateway auf Empfängerseite wieder komplett entfernt werden kann. Hierdurch kann ein vollständig transparentes System implementiert werden, welches keine Änderungen an den Mailprogrammen der Benuzter erfordert.

Desweiteren kann zur Vereinfachung der Administration ein globaler Schlüssel pro Domain festgelegt werden, der dann nur den Domainteil authentifiziert und den eigentlichen Absender nicht berücksichtigt.

Neben der Signaturfunktionalität bietet PKA auch einen Weg zur opportunistischen Verschlüsselung an: Mittels des optionalen URI Parameters im PKA Record kann für jeden Empfänger ein Public Key ermittelt werden und dieser automatisch (z.B. auch durch eine Gateway) benutzt werden.

10 Nachteile

Obgleich gefälschte Absenderadressen mittels PKA sicher festgestellt werden können und auf diese Weise die Erkennung von Spam erleichtert, gibt es auch Schattenseiten.

[4] Es war zu der Zeit noch keine definierte Möglichkeit vorhanden, um aus einem Plugin die From: Adresse zu ermitteln.

Die Verfügbarkeit der PKA Records im DNS erlaubt eine einfache Überprüfung auf gültige Mailadressen. MTAs erlauben dies aus gutem Grund heute nicht mehr; mittels PKA wird aber eine indirekte Prüfungsmöglichkeit wieder eingeführt. Es sei hier noch angemerkt, daß eine Auflisting aller Mailadressen einer Domain i.d.R. nicht möglich ist und so ein direktes Abernten von Adressen durch Spammer nicht möglich ist.

Dieses Problem tritt allerdings nur solange auf, wie PKA nicht im großen Maßstab eingesetzt wird — durch konsequenten Einsatz würde Spam theoretisch unmöglich gemacht...

... wenn es denn da nicht die Zombienetze (gekaperte Rechner an DSL) gäbe, über die heute ein Großteil des Spams versendet wird. Selbst wenn von diesen gekaperten Rechnern zwangsweise ein Smarthost des ISP benutzt werden muß, so kann dieser doch nicht unterscheiden, ob es eine legitime Mail ist und wird im Zwefelsfall eine gültige PKA Signatur hinzufügen. Immerhin kann aber so verlässlich und schnell festgestellt werden von welchem Rechner bzw. Mailaccount der Spam versandt wurde.

Da aus technischen Gründen PGP/MIME als Signaturverfahren verwendet werden soll, könnten – falls auf der Empfängerseite keine PKA Überprüfung mit Entfernung des Signatur durchgeführt wird — einige alte Mailprogramme Schwierigkeiten haben, derartig signierte Mails darzustellen.

11 Zusammenfassung

PKA bietet einen leicht zu implementieren Weg zur Authentisierung von Absenderadressen basierend auf existierenden Protokollen. Es werden starke und erprobte Kryptographieprotokolle eingesetzt. Der neu zu schreibende und pflegende Code ist minimal und fügt sich leicht in bestehende Strukturen ein. Stolpersteine und Fehler wie sie bei neuen Protokollen immer vorhanden sind werden vermieden. Die Funktionseinheiten sind klar getrennt.

Vielen Dank an Thomas Roessler, der die ursprünkliche Idee zu PKA hatte und einige wertvolle Anregungen gab; David Shaw und Simon Josefsson für Diskussionen zur exakten Anwendung des DNS und nicht zuletzt den Autoren des DKIM I-Ds, die den Anlass gaben, dieses Verfahren zu entwickeln.