1. PGP – warum, was und wie?

- Ziele: unsere und die der Bullen
- Knappe Theorie
- Verschlüsseln
- Signaturen
- Das Web of Trust
- Praktikalitäten
- Praxis

2. Warum PGP?

Auswertung von E-Mails ist ein Routineverfahren bei Polizei und Geheimdiensten. Wegen der relativ geringen Datenmengen und der direkten Verarbeitbarkeit (im Gegensatz zu Telefonie oder Videos) von E-Mails ist flächendeckende Speicherung und z.B. Schlüsselwortsuche technisch trivial.

Viele Wege führen zur Mail: „Normales“ Abhören geschieht in der BRD gemäß TKÜV und StPO bzw. PolGs in der Regel nach Mail-Adresse überwacht. Globales Scannen macht der BND nach seinem Gesetz. Bei GMX, Google usf. liegende Mails sind mehr oder minder Freiwild, sie sind *nicht* vom TK-Geheimnis geschützt (auch wenn die Unternehmen mittlerweile idR immerhin etwas auf die Plausibilität der Anordnungen schauen).

PGP ist ein offener, interoperabler Standard. Das heißt, dass es eine offen gelegte Spezifikation gibt und jedenfalls im Prinzip jedeR Software schreiben kann, die PGP versteht. Tatsächlich ist das Programm, das ihr wohl alle verwenden werdet, GnuPG, eine alternative Implementation zu dem originalen PGP.

PGP ist relativ einfach verständliche Krypto. Mensch will natürlich nicht nur Mails verschlüsseln, sondern auch Chat, HTTP, Telefonie usf. Aus historischen, kommerziellen, und teilweise auch in der Natur der Sache liegenden (z.B. die forward secrecy von OTR) Gründen wird da eher selten PGP eingesetzt. Die Verfahren sind aber ähnlich, wenn auch idR schwerer zu verstehen (ganz krass bei HTTPS, das auf einem Monstrum namens X.509 aufbaut). Krypto, die mensch nicht versteht, ist fast wirkungslos. Ein Verständnis für PGP hilft, auch über die anderen Verfahren nachzudenken und zu beschließen, wie viel mensch ihnen vertrauen will.

Oh: Es gibt für E-Mail auch ein Verfahren namens S/MIME. Das teilt all die Probleme von HTTPS im Hinblick aufs Schlüsselmanagment und ist quasi PGP, wie es sich Chefs wünschen. Schaut, dass ihr drumrumkommt.

Mails verschlüsseln ist praktische Solidarität. Und zwar nicht nur krass politische Mails. Denn wenn nur die verschlüsselt sind, sagt ist die Tatsache, dass mensch verschlüsselt, der Gegenseite ziemlich viel. Das hat sich schon etwas geändert. Und muss sich noch mehr ändern.

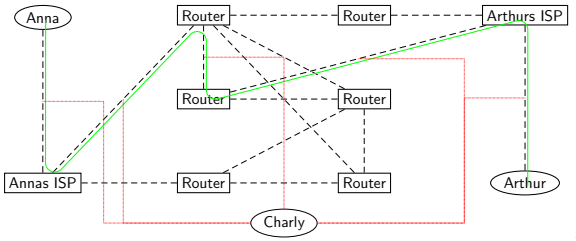

Fig. 1

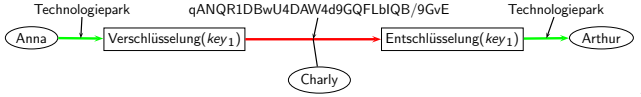

Fig. 2

3. Wo wird Netzverkehr abgehört?

(cf. Fig. 1)

„Bedarfsträger“ haben an fast jeder Stelle Zugriff auf im Netz transportierte Daten.

Das gilt natürlich nicht nur für Mails, sondern genauso für jede andere Kommunikation am Netz.

Das Problem der Verkehrsdaten („wer mit wem?“) behandeln wir hier nicht, obwohl es speziell bei der Telefonie derzeit viel gravierender ist als das der Inhalte. Es gibt auch dabei technische Mittel, aber lasst uns erstmal die Basics klarkriegen.

Was die Grafik auch nicht enthält: Wenn Charly bei einem Webmailer sitzt, kann die Polizei unter sehr geringen Voraussetzungen auf die Mails zugreifen (technisch: Sicherstellung, wenn das Unternehmen kooperiert, Beschlagnahme, wenn nicht). Das solltet ihr also besser vermeiden. Die Rechtslage bei ungelesenen Nachrichten, die aufs POPen warten ist unklar, aber wahrscheintlich deutlich besser.

4. Symmetrische Verschlüsselung

(cf. Fig. 2)

Anna und Arthur vereinbaren eine Funktion, die aus normalem Text („Technologiepark“) scheinbar sinnlose Zeichen („qAN...“) macht. Die Funktion kann umgekehrt werden (d.h. aus den sinnlosen Zeichen wird wieder der Text) – aber zur Berechnung der Funktion braucht mensch den Schlüssel key_1.

Ist die Funktion gut gewählt, kann Charly ohne Kenntnis von key_1 den Klartext nicht (mit vernünftigem Aufwand) rückgewinnen.

Ein simples Beispiel für die Funktion könnte sein: „Wandele jedes Zeichen in die Position seines ersten Vorkommens auf einer zufällig gewählten Seite eines Buchs.“ Der Schlüssel ist dann der Titel des Buchs.

Probleme:

1. Übermittlung des Schlüssels
2. Wer verschlüsseln kann, kann auch entschlüsseln

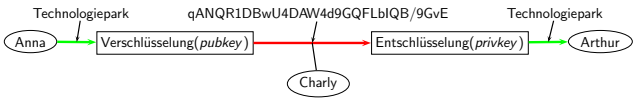

Fig. 3

3. Gute Wahl der Verschlüsselungsfunktion
4. Sicherheit der Leitungen zwischen Anna und der Verschlüsselung bzw. zwischen der Entschlüsselung und Arthur

5. Asymmetrische Verschlüsselung

(cf. Fig. 3)

Zumindest die ersten beiden Probleme lösen asymmetrische (public-key) Verschlüsselungsverfahren.

Idee: Verschlüsselungsfunktion braucht nur einen Teil des Schlüssels, (den *öffentlichen Schlüssel*) zum Entschlüsseln braucht mensch den vollen Schlüssel (den *privaten Schlüssel*).

(Gute) Metapher: Briefkästen – jedeR kann einwerfen (der public key: aus einer offenen Nachricht wird eine versteckte, weil sie im Kasten liegt), nur wer einen Schlüssel hat, kommt an den Inhalt ran (der private Schlüssel: Nur er kann den Brief wieder ans Tageslicht bringen). Die Gefahr, dass Leute mit Nachschlüsseln oder roher Gewalt an den Inhalt herankommen, ist bei aktuellen Verfahren und verantwortungsvollem Umgang mit den Schlüsseln praktisch vernachlässigbar.

Probleme:

1. Sicherheit des Schlüssels
2. Sicherheit bis zum Verschlüsseln („Staatstrojaner“)
3. „Vertauschen der Namenschilder“

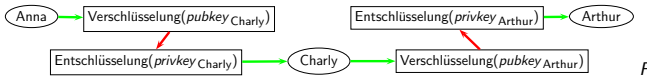

Fig. 4

6. Man in the middle

Wenn Charly Anna vortäuscht, Arthur zu sein und umgekehrt Arthur, er, Charly, sei Anna, kann Charly mitlesen:

(cf. Fig. 4)

Man-in-the-middle attack

Solche MITM-Angriffe lassen sich im Wesentlichen nur verhindern, wenn Arthur eine Möglichkeit hat, sich von der Authentizität von Annas Schlüssel zu überzeugen.

Krypto ohne Schlüsselverifikation stinkt.

Mit anderen Worten: Wenn ihr irgendeine Verschlüsselung macht, und ihr müsst euch keine Gedanken darüber machen, ob der Schlüssel eurer PartnerIn auch wirklich ihr gehört, seid ihr vermutlich (zumindest) gegen einen MITM-Angriff verletzlich (und es ist nicht unwahrscheinlich, dass sich auch gleich jemand Nachschlüssel machen kann). Der Punkt mit der (relativ) leichten Verständlichkeit von PGP ist, dass PGP eben ein recht explizites Schlüsselmanagement hat.

Die Abhilfe für MITM geht über die digitale Signatur. Sie ist letztlich ein Wert, den nur berechnen kann, wer Annas privaten Schlüssel als auch ein paar Daten hat. Wenn jetzt Arthur die Daten, diesen Wert und den Annas *öffentlichen* Schlüssel hat, kann er sehen, ob Anna auch wirklich die Daten unterschrieben hat.

Dass es eine Funktion gibt, die so einen Wert berechnet, scheint zunächst unwahrscheinlich, sie ist aber ein relativ unproblematisches Abfallprodukt üblicher asymmetrischer Verschlüsselungsverfahren.

Wenn die Daten jetzt ein öffentlicher Schlüssel sind, kann jemand öffentlich sagen: „Ich habe mich überzeugt, dass dieser Schlüssel tatsächlich zu einer Person oder ein Gruppe gehört“.

Beispiele für unterschriebene Schlüssel:

- In euren Browsern sind die öffentlichen Schlüssel von Firmen (z.B. verisign) enthalten, mit denen nachgeprüft wird, ob z.B. die Webseite von Microsoft auch wirklich von Microsoft kommt.
- In den neuen Personalausweisen sind Schlüssel, die vom Innenministerium unterschrieben werden.

Natürlich wollen wir uns in unserer Kryptografie nicht auf das Innenministerium (das mit Sicherheit früher oder später Nachschlüssel anfertigen wird) oder auf große Firmen (die viel Geld für ihre Dienste verlangen und auch nicht zwingend unabhängig von allerlei Diensten sind) verlassen. PGP verwendet daher eine andere, soziale Möglichkeit zur Verifizierung von Schlüsseln.

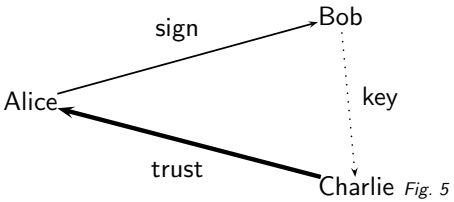

Fig. 5

7. Das Web of Trust

- Leute signieren die öffentlichen Schlüssel anderer Leute und bestätigen damit ihre Authentizität
- Leute vertrauen anderen Leuten, dass diese wirklich nur das signieren, was sie geprüft haben

Es ergibt sich ein Netz des „Kennens“, über das sich im Idealfall die Identitäten klären lassen. Im einfachsten Fall sind daran drei Leute beteiligt.

Hier kann Charlie Bobs Schlüssel vertrauen, weil er Alices Unterschrift trägt und Charlie Alice vertraut.

(cf. Fig. 5)

Dabei vertraut Charlie Alice (voll), d.h., er akzeptiert einen von Alice unterschriebenen Schlüssel, als hätte er ihn selbst geprüft. Außerdem hat Alice Bobs Schlüssel unterschrieben und hat ihn Bob zurückgegeben und auf einen Keyserver hochgeladen.

Wenn nun Charlie den Schlüssel von Bob bekommt (oder von einem Keyserver holt), sieht Charly die Unterschrift von Alice und weiß, dass Alice glaubt, der Schlüssel gehöre Bob. Weil Charlie Alice vertraut, wird sie den Schlüssel als authentisch akzeptieren.

In Wirklichkeit ist das ganze etwas subtiler. Ihr könnt in der Regel den „Grad“ des Vertrauens in Personen angeben und auch die Sicherheit, mit der ihr den Schlüssel überprüft habt. Die PGP-Programme berechnen daraus dann einen „Grad“ des Vertrauens, das sie in einen Schlüssel haben.

So schön diese Idee – eine soziale Lösung für ein technisches Problem, ganz im Gegensatz zu den versuchten Lösungen sozialer Probleme durch technische Maßnahmen, denen wir in dieser Gesellschaft so oft ausgesetzt sind – auch ist, das weite Web of Trust ist vorerst noch Utopie, insbesondere, weil die dahinterstehenden Konzepte doch etwas diffizil sind. Eine populäre Alternative sind „grass-roots“-Signierkampagnen (der Heiseverlag macht sowas), die aber leider an Unsinn wie Personalausweisen hängen, und Keysigning-Partys.

8. Für die RH

Zentrales Problem des WoT: Die Unterschriften müssen öffentlich sein, und damit kann jedeR einen sozialen Graph malen („wer hat mit wem zu tun?“).

In der RH deshalb: Es gibt (derzeit) einen Schlüssel, der alle Organisations-Schlüssel unterschreibt. Damit hat die Staatsgewalt keine Info über die offensichtliche hinaus, dass eine Adresse mit der RH zusammenhängt. Also:

1. Ihr holt den Fingerprint des Datenschutzgruppen-Schlüssels aus der RHZ.
2. Ihr holt den Schlüssel von datenschutzgruppe@rote-hilfe.de vom Keyserver.
3. Ihr verifiziert den Schlüssel und stellt *danach* das Vertrauen auf dem Schlüssel auf „voll“.

Danach könnt ihr die Vertrauenswürdigkeit jedes RH-Schlüssel überprüfen.

Mit Thunderbird und Enigmail sieht das so aus (modulo Sprache):

1. *Enigmail* → *Key Management*
2. *Keyserver* → *Search for Keys*
3. datenschutzgruppe@rote-hilfe.de suchen, den 4096er-Schlüssel nehmen
4. Rechtsklick auf Schlüssel, *Key Properties*
5. Fingerprint mit dem in der RHZ vergleichen
6. Bei *You rely on certifications* auf *Change* klicken
7. *You trust fully*
8. Danach könnt ihr bei jedem Schlüssel einfach Rechtsklicken und bei *Key Properties* die *Validity* prüfen.

Im Idealfall würdet ihr in den Einstellungen unter Sending *Only trusted keys* anklicken. Damit das Spaß macht, brauchen wir aber wahrscheinlich noch etwas mehr Routine (allerdings: Die Voreinstellung meines Mailclients (mutt) war von jeher, nicht unterschriebene Schlüssel nur nach Rückfrage zu benutzen. . .)

9. Schlüsselwechsel

Manchmal will mensch einen neuen Schlüssel haben:

- Leute, die den Schlüssel haben, werden unheimlich
- Mensch will einen längeren Schlüssel
- Mensch will beschränken, wie viel verschlüsseltes Material es zu einem Schlüssel gibt (das ist, so, wie PGP gemacht ist, aus unserer Sicht kein Grund zur Sorge)

Wenn Leute ständig Schlüssel wechseln und die anderen die einfach importieren, haben MITM leichtes Spiel.

Deshalb: Macht ein Transition Statement und schickt es an alle@.

Näheres: http://datenschmutz/gc/html/sicherumziehen.html.

Oh, und natürlich: Lasst uns den Fingerprint zukommen, normalerweise via ODT und BuVo.

10. PGP/MIME

Noch eine wichtige Kleinigkeit:

In den alten Zeiten gab es keine Anhänge und keine deutschen Sonderzeichen.

Heute gibts das, und es funktioniert nicht gescheit, wenn ihr nicht PGP/MIME auswählt. Alte Enigmails haben das eher schwer gemacht.

Heute: (Rechtsklick auf Mailbox) → *Properties*, *OpenPGP Security*, *Use PGP/MIME by default*.

11. Zum Ende

https://www.datenschmutz.de