Implementing Cryptography

We could protect and authenticate communication within our group by using *encryption* and *digital signatures.* We might choose either the simple *OpenPGP* standard, used by individuals and small groups, or *S/MIME,* popular in the corporate world – both well supported by cryptography software and e-mail clients – or we might adopt the *TOFU* model used in modern cryptographic messaging applications.

We could also use it to protect and authenticate communications to and from people outside our own group, such as our professional advisers or beneficiaries, if they were willing.

This paper discusses the choice of software, message format and cryptographic key management. The companion paper *Pros and Cons of Cryptography* summarises the benefits, drawbacks and costs of cryptography; *Background to Modern Cryptography* sets out the basic principles of Public Key Cryptography, with a brief account of how we got here and where we may be going; and there is also a *Glossary of Cryptographic Terms.*

Prerequisites for Cryptography

The parties setting up cryptography for secure communication must each:

1. Choose, obtain and install the application software that they wish to use for communication, e.g. e-mail client or Instant Messaging platform (see Appendices II and III).

2. Agree on and adopt the same standards for message format and key management. These standards are discussed later in this paper.

3. Create cryptographic keys. Depending on the choice of software and key management, they may do this themselves through the software, or they may use a facility provided by a *Certification Authority,* or the keys may be created automatically by the software.

4. Exchange *Public Keys* with one another, making sure the keys they receive have really come from their purported owners. This is discussed below and in *Background to Modern Cryptography* section Public Key Cryptography.

Identity in the Virtual World

For secure communication, you need to be sure of the *identity* of the person you are dealing with. When you send confidential information, you want to know it has gone to someone whom you trust to keep it secret and not misuse it. When you receive information or advice that you need to rely on, you want to know it was sent in good faith and is not a confidence trick.

What 'identity' means depends on the circumstances. The internet has a tradition of anonymity[*], so you may never meet your contact in person but, instead, know them on the internet by an online *persona*. In an e-mail based system, this is commonly their e-mail address; in a social media based system it may be their mobile telephone number or username.

When you communicate with your contact using cryptography, it uses their Public Key (see Public Key Cryptography in the *Background to Modern Cryptography* paper) to encrypt outgoing messages from you to them and to verify signatures on incoming messages from them to you. From the cryptography software's point of view, the key represents your contact in cyberspace. It is bound to their persona.

This ensures that an encrypted message you send can be read by only one person, the owner of the key; and that an incoming message signed with that key can only have come from that same person; but it does not tell you who the person *is.* You need something more: a link between your contact's key and their true identity. By this, we mean sufficient details to track them down if needed – for example, their real name and that of the organisation they represent (if any), and their real-world address – which banks typically ask of personal bank account applicants.

In an e-mail based system, you might find this embedded in the OpenPGP key, authenticated through person-to-person exchange or Web of Trust; or in the S/MIME certificate, validated through Certification Authorities and Public Key Infrastructure; though such systems often provide less information than that suggested above. In social media-based cryptography, TOFU leaves gathering and validation of such information to you, the user. All three rely on the same cryptographic algorithms, but differ in the way *key owners* distribute and authenticate their Public Keys to the *key users* who will rely on those keys.

The next three sections compare these standards. You can go straight to the Conclusions on page 10 if you wish to skip the supporting arguments.

[*] Cartoon, originally from the *New Yorker*, reproduced in Wikipedia: https://en.wikipedia.org/wiki/On_the_Internet,_nobody_knows_you're_a_dog

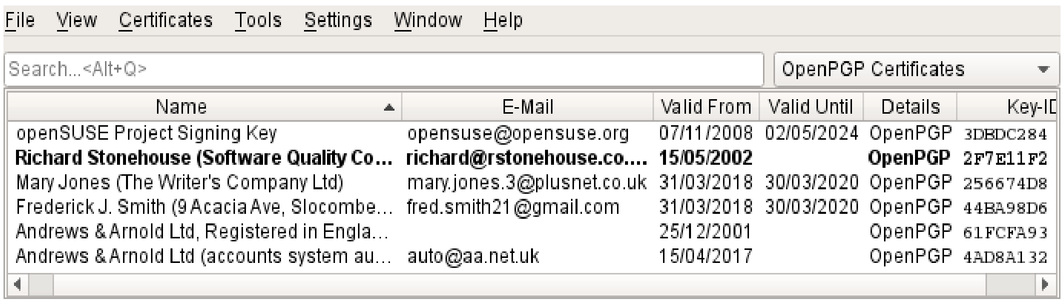
OpenPGP

OpenPGP[*], based on Phil Zimmermann's 'Pretty Good Privacy' program (PGP), provides good identification within small groups of people. They can set it up themselves without the need to involve any external third party.

Person-to-person Key Exchange

The parties may exchange and authenticate Public Keys person-to-person. For example a key owner may hand you a copy of their Public Key at a face-to-face meeting; or they may send it via e-mail and confirm it by a *fingerprint* which they hand to you, send in a letter or dictate by telephone. You can give the other party a copy of your Public Key in the same way. Person-to-person exchange between people who know one another well authenticates their ownership of their Public Keys to a high standard.

You import the Public Keys you receive into your own key store. Each contains its owner's name, e-mail address and usually some description; for example[†]:



Name	E-Mail	Valid From	Valid Until	Details	Key-ID
openSUSE Project Signing Key	opensuse@opensuse.org	07/11/2008	02/05/2024	OpenPGP	3DBDC284
Richard Stonehouse (Software Quality Co…	**richard@rstonehouse.co.…**	**15/05/2002**		**OpenPGP**	**2F7E11F2**
Mary Jones (The Writer's Company Ltd)	mary.jones.3@plusnet.co.uk	31/03/2018	30/03/2020	OpenPGP	256674D8
Frederick J. Smith (9 Acacia Ave, Slocombe…	fred.smith21@gmail.com	31/03/2018	30/03/2020	OpenPGP	44BA98D6
Andrews & Arnold Ltd, Registered in Engla…		25/12/2001		OpenPGP	61FCFA93
Andrews & Arnold Ltd (accounts system au…	auto@aa.net.uk	15/04/2017		OpenPGP	4AD8A132

When you send an *encrypted* message, the cryptography software encrypts it using the Public Key of the intended recipient; by default, the one whose entry in the E-mail column above matches the e-mail address (persona) you are sending the message to. They will decrypt the message using their own *Private Key.*

When you open a *digitally signed* message that you have received, the OpenPGP software tells you who sent it by displaying a signature verification report that contains the sender's identity details. These come from the sender's Public Key[‡], as in the following example:

---

[*] See RFC 4880: OpenPGP Message Format.

[†] Personal details in this example are fictitious.

[‡] The sender will have signed the message using their Private Key, but the identity details come from its Public Key counterpart, a copy of which is in your key store.

```
Good signature from: Richard Stonehouse (Software Quality
Consultant)
+<richard@rstonehouse.co.uk>
                created: Tue 04 Apr 2017 19:48:05 BST
```

In the case of well-known organisations, it is not always necessary to go through a full person-to-person key exchange – and sometimes may not be feasible. In the example above, openSUSE and Andrews & Arnold are widely-known and reputed companies, and I have been their customer for several years; their websites are secure and traceable to them; and the keys were downloaded from those sites via a secure internet connection. I think this authenticated them well enough.

Key Servers and Webs of Trust

A drawback of person-to-person key exchange is that it does not scale up well. As the group expands, the number of key exchanges grows rapidly. To solve this, the key owner may upload their Public Key to a publicly-accessible *key server*; anyone can download keys from there, as needed, to their own key store.

A more difficult problem is: if the parties live too far apart to meet, or are not familiar enough to identify one another positively, how can the would-be user of the key be sure that it belongs to the person they think it does? The key server makes it easy to download the key, but does not authenticate it.

OpenPGP allows the parties to authenticate the key via a trusted third-party intermediary known to both of them. The intermediary vouches for the key owner's identity by signing the key; the would-be key user accepts and relies on this assurance. In more complex cases there may be a chain of intermediaries: for example the key user knows and trusts intermediary A; A knows and trusts B; and B knows and trusts the key owner. But the more intermediaries, the more doubtful is the authentication; people differ in the criteria on which they judge identity and the diligence with which they apply those criteria, so there may be weak links in the chain.

This is a very simple example of a *Web of Trust* (WoT). It is a distributed structure without central top-down control; authentication criteria are not strictly defined but left largely to individual judgement. There is much theory about WoT but, while subsets are in common use, it is not clear to me that people are successfully using the full system as described in the literature.

S/MIME

S/MIME[*] is designed for the corporate world. People who use it may not know one another and may be geographically remote, so it relies on the formal business relationships that exist in that world, not the personal knowledge common in OpenPGP. Public Keys are distributed in digitally signed electronic documents called *certificates,* which also authenticate the keys:

- **Whose key is this?** The certificate states the identity of its owner;
- **Who says so?** The issuer of the certificate signs it to certify that the Public Key and identity details belong to the same person; and
- **Why believe them?** Usually because the issuer is a *Certification Authority* (CA) with an established brand name and a reputation to protect. It is perfectly possible for anyone to create their own *self-signed certificate,* but only CA-signed certificates are widely trusted.

The CAs effectively control the system. A would-be *Certificate owner* creates a *Certificate Signing Request* (CSR) and sends it to a CA for validation and signature, or uses one of the web-based systems (see Appendix I). Certificate owners are customers of the CAs for this service.

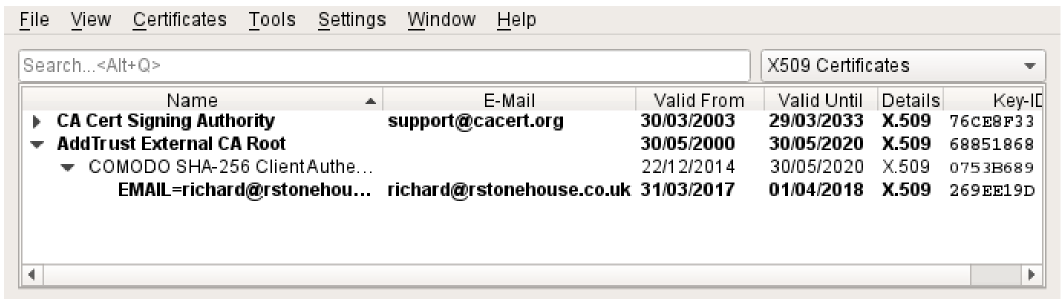
A certificate has a limited validity period, typically 6 to 12 months, after which its owner must renew it. Also, a CA may *revoke* a certificate early, for example if it has been compromised or the owner is in breach of its terms. *Certificate users* should not trust expired or revoked certificates. Their cryptography software will warn them if the certificate they are about to use has expired and should be set up to do a *revocation status check* on certificates too.

Public Key Infrastructure

In S/MIME, *root CAs* are the source of all trust. Your certificate might be signed by a root CA, or by a *subordinate CA* granted the right to do so by a root CA or a higher-level subordinate CA. These can form a chain, a bit like OpenPGP's WoT. Users need to have your certificate, and also the root and subordinate certificates, in their key stores; but they need not install the latter themselves – the common ones (but not CAcert) come pre-installed in most systems. Users are, in effect, placing their trust in their software suppliers, perhaps without being aware of it.

[*] See RFC 8551: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification

The following example shows a certificate store with two certificate chains. The first (**CA Cert Signing Authority**) is rolled up; the second is expanded to show my certificate **EMAIL=richard@rstonehouse.co.uk** signed by **COMODO SHA-256 Client Authentication and Secure Email CA** which, in turn, is signed by the root certificate **AddTrust External CA Root**:



Name	E-Mail	Valid From	Valid Until	Details	Key-I
▸ **CA Cert Signing Authority**	**support@cacert.org**	**30/03/2003**	**29/03/2033**	**X.509**	76CE8F33
▾ **AddTrust External CA Root**		**30/05/2000**	**30/05/2020**	**X.509**	68851868
▾ COMODO SHA-256 ClientAuthe...		22/12/2014	30/05/2020	X.509	0753B689
EMAIL=richard@rstonehou...	**richard@rstonehouse.co.uk**	**31/03/2017**	**01/04/2018**	**X.509**	269EE19D

Such a structure is known as a *Public Key Infrastructure* (PKI).

Personal Certificates

S/MIME Personal certificates for individual certificate owners are usually issued free-of-charge or at low cost.

These certificates bind their owner's Public Key to their e-mail address (persona) but give no other identity details. This deficiency is reflected in the verification report you see when you receive a message from them – for example:

```
Good signature from:
+1.2.840.113549.1.9.1=#7269636861726440727374 6F6E65686F7573652E
636F2E756B
aka: <richard@rstonehouse.co.uk>
created: Fri 31 Mar 2017 20:11:05 BST
```

This is not enough to identify the sender or provide a starting point for further enquiries. The former method of finding the sender's true identity, by looking up the *domain* name component of their e-mail address in the *WHOIS* service, no longer works because the WHOIS output does not now show contact details.

A few CAs, e.g. CAcert (see Appendix I) and GlobalSign[*], offer certificates that give the owner's proper name as well as e-mail address. But I am not aware of any that meet the criteria in the *Identity in the Virtual World* section.

---

* See PersonalSign2 certificates at https://shop.globalsign.com/en/secure-email.

It has been suggested that the reason may be competitive pressures* on CAs to offer their customers low-cost certification, by limiting validation to an automated *e-mail ping*.

But inadequate owner details leave certificate users, who are not customers, at risk of being taken in by fake certificates. They may be reliant on the scanty details provided because they have no direct contact with, or knowledge of, the owner.

Enterprise Certificates

Enterprise certificates, intended for companies and large organisations, are more expensive than personal certificates.

These certificates can provide *Extended Validation* against official registers, to prove ownership of the internet domain, organisation identity and physical presence of the legal entity who owns the certificate. However unincorporated bodies such as ours, having no entry in the Companies Register, may not be eligible for this service. It is mainly used for SSL (secure web-sites) and its usage for S/MIME is not yet standardised.

Big companies and organisations may be able to set themselves up as subordinate certification authorities and issue certificates to their own staff. An external CA would issue a certificate to the company conferring the authority to do this; the certificates that the company issued to its staff and signed as sub-CA would be traceable back to the external CA. This would provide good authentication because the CA knows the company and the company knows its staff.

* See Hagai Bar-El: The Inevitable Collapse of the Certificate Model (https://www.hbarel.com/analysis/itsec/the-inevitable-collapse-of-the).

TOFU

Trust on First Use (TOFU) has been adopted as the trust model for social media-based cryptography. It is not exclusive to that; SSH* and GnuPG also use it.

In TOFU, mutual trust between a pair of correspondents is established online the first time they try to communicate. This differs from OpenPGP's Web of Trust and S/MIME's Public Key Infrastructure, where users are authenticated in advance using offline methods.

TOFU records this trust by binding the user's persona (such as phone number or username), by which you know them on the *platform,* to their Public Key. TOFU does not bind the user's key to their true identity (see the Identity in the Virtual World section); it knows only about keys, not the people they represent, so it leaves making this connection up to you.

First Contact

When you start a secure conversation with a contact for the first time, you make a kind of contract with them. It is sealed by an exchange of keys; the software on your device receives a copy of your contact's key from them and stores it in your key store, and vice versa.

Attackers may try to subvert your communication, either by *phishing* or by a *Man in the Middle* (MITM) exploit:

In a **phishing** attack, the person you are making contact with for the first time pretends to act in good faith, when in fact their aim is to deceive you by persuasion or false promises into serving their purposes.

To judge whether a contact is someone you can trust, you need to know who they are. Perhaps they appear to be someone you know well and you can confirm this, for example by telephoning them. If not, you might want to get an idea of their work and reputation by meeting them or from mailing lists or social media (if they have a presence there). All this takes time and you may be under pressure to decide in haste, but you should not just think "It'll probably be all right" and click OK; it might not be.

In a **Man in the Middle** attack, the person you are *trying* to make contact with is genuine, but the conversation has been hijacked by an interloper who pretends to be that person and feeds off your trust in them.

* See RFC 4251: The Secure Shell (SSH) Protocol Architecture

The initial communication is at risk from such an interloper, if they can send you a fake key during the short time window when you are expecting to receive a genuine key from your contact. You should check *safety numbers* after the key exchange to detect whether the key you received is the real one or a forgery.

Existing Contact

When you start another conversation with a previous contact, TOFU checks the key they have supplied now against the one you trusted and stored earlier and:

- if they are the same, all is well and the conversation can proceed; or

- if not, the contact may be a friend who has changed their key, or may be a Man in the Middle attacker. You should check safety numbers, images or fingerprints via an independent route such as phone.

Conclusions

1. The S/MIME and OpenPGP based implementations of Public Key Cryptography are well established, comply with open standards and so provide universal internet-wide secure e-mail communication. However they lack some modern features and are not very easy to use.

2. The strengths of S/MIME, relative to OpenPGP, are: its formal structure of certificates and certification authorities; its credible solution to the problems posed by a large user community who may not have face-to-face contact; and services, suited to the needs of big organisations, that build on arms-length relationships in the corporate environment.

3. Its weaknesses are: lack of good identity data in S/MIME personal certificates, which are the only sort available to bodies such as ours; reliance on Certification Authorities to authenticate our contacts, to the exclusion of our own often deeper knowledge; and the burden of dealing with S/MIME certificates and revocation lists.

4. If we want secure communication between ourselves, a simple OpenPGP setup (without Web of Trust) would suit us well. It offers stronger identification of certificate owners, at less cost and effort than S/MIME, to small groups of people who know one another. But, if we wished to extend our secure communication to the outside world (and if the outside world wanted to participate), we might come under pressure to adopt S/MIME as well as or instead of OpenPGP.

5. Cryptographic applications using TOFU-based authentication are under active development and already have a huge user base; which way they are leading is not clear.

6. TOFU looks simple, but this may be misleading. It places more of the responsibility for authentication onto the user than OpenPGP or S/MIME do. The user needs to have the time, knowledge and inclination to do this properly, otherwise they could be open to a phishing attack causing a major security breach.

Richard Stonehouse

Appendix I – Free Personal Certificate Providers

The information in this appendix has been tested by generating ***Comodo*** *and* ***CAcert*** *certificates and checking that they worked. Comodo has since been renamed or acquired by* ***Sectigo*** *and no longer issues free certificates.*

CAcert (http://www.cacert.org/)

CAcert is a not-for-profit Certification Authority.

Unlike many other CAs, CAcert are willing to sign certificates containing the certificate owner's real name as well as e-mail address. To get one of these, you must first prove your identity by attending two face-to-face meetings with CAcert assessors and producing official photo id documents to them.

CAcert's certificates are free of charge, except possibly a charge for assessors' travel expenses.

You need to become a CAcert member before you can obtain certificates; this does not impose heavy obligations. Apply online by using the form at:

https://www.cacert.org/index.php?id=1

To obtain a basic personal certificate containing just your e-mail address, for non-commercial use:

1. Click on the *Email Accounts* menu entry *Add* sub-menu entry then fill in and submit the online form that is displayed.

2. CAcert e-mails further instructions to your specified e-mail address. This acts as an *e-mail ping* check that the address is valid and you have access to it; if not, the e-mailed instructions will not reach you and you will be unable to proceed.

3. To generate the certificate, you need to run Microsoft Internet Explorer or Mozilla Firefox, with JavaScript and ActiveX Controls enabled, under Microsoft Windows Vista (or above) or Apple MacOS, and follow the instructions given in the e-mail. The browser stores the resulting certificate bundle in its certificate store.

Alternatively, you may use the Open SSL software. This is more complex but more flexible than the browser method. See:

https://wiki.cacert.org/EmailCertificates

Again, the end result is a bundle containing your Private Key and your Public Key certificate.

In either case, you copy the certificate bundle to a file. You may import this* into the certificate store of your e-mail client or take it to another system; you can install it in a wide range of mail clients, on any computer that supports them.

But, first, you will need to install CAcert's Class1 PKI Key root certificate; unlike other CAs' root certificates, this is not usually pre-installed in your system. Download it from:

http://www.cacert.org/index.php?id=3

If you are using an old version of GnuPG, you must mark the root certificate as trusted by using the `kleopatra` utility or by editing the trustlist.txt file.

You may make a file containing your Public Key certificate alone (not the full bundle) and distribute it to users. They will also need to install CAcert's root certificate.

* Not necessary if your browser is Internet Explorer and your e-mail program is Microsoft Outlook, as they share the same certificate store.

Appendix II – Software for OpenPGP and S/MIME

The following software supports OpenPGP and S/MIME on modern systems; some also works on older platforms (see Compatibility Summary). This information comes from product documentation and testing by me as noted in the text.

Desktop Computer Cryptography Software

GnuPG is Free Software and free-of-charge, but a small donation is requested. It supports OpenPGP, S/MIME and TOFU, with variants for different platforms:

– **Gpg4win** *(tested)* runs on Microsoft Windows. Gpg4win includes a plug-in *(not tested)* to integrate with Microsoft Outlook (*not* Outlook Express).

Main page:	`http://www.gpg4win.org/`
Download:	`http://www.gpg4win.org/download.html`

– **GPGTools** *(not tested)* runs on Macintosh and integrates with Apple mail.

Main page:	`http://gpgtools.org/gpgsuite.html`
Download:	click on the Download button in the above page.

– **gpg2** *(tested and used in operation)* runs on Linux, Unix, OpenBSD etc., and integrates with several e-mail programs for those systems.

Main page:	`http://gnupg.org/`
Download:	`https://www.gnupg.org/download/`

Desktop Computer Cryptography-aware e-mail Clients

The following is a selection of e-mail clients that support both S/MIME and OpenPGP key management.

Microsoft Outlook *(not tested – a 'must do' if we decide to adopt cryptography)* is part of the Microsoft Office suite. It supports S/MIME out of the box, and OpenPGP via Gpg4win and its plug-in. The Gpg4win plug-in can also support S/MIME, as an alternative to using Outlook's built-in feature, but you would probably not want to do this.

Note that Outlook Express is a completely different program from Outlook and does not work with Gpg4win.

The current version of Microsoft Office runs on modern Windows systems:

Rent or buy: https://www.microsoft.com/en-us/store/b/office?icid=Homepage_LeftNav_01_Office_en_US

Thunderbird *(tested on Windows)* comes from the Mozilla project, best known for the Firefox browser. It is popular and well-established. It supports S/MIME out of the box, and OpenPGP through the **Enigmail** add-on and GnuPG. It is a fully-featured mail client with a modern user interface and extra features, including calendar, contact management and chat, provided by add-ons.

The current version requires a modern PC (64-bit or 32-bit with SSE2) running Windows, Macintosh or Linux. It is Free Software:

Download: https://www.mozilla.org/en-US/thunderbird/

For the Enigmail add-on, see the *Add-ons* command in Thunderbird's menu.

Claws Mail *(tested on Windows)* originates from the Unix/Linux community. It supports both S/MIME and OpenPGP via GnuPG on Windows, Macintosh and Linux. It does all the basic things that a mail client has to do but lacks some of the extra features found in more sophisticated e-mail clients. It does not send HTML ('rich text') e-mail with fancy fonts and graphics, but can display such messages if received. The user interface is traditional and simple.

Claws-mail is relatively lightweight, and can run on older PC hardware and software. It is Free Software and free-of-charge:

Download: http://www.claws-mail.org/

eM Client *(tested)* is a new e-mail client for Microsoft Windows only. It supports S/MIME and OpenPGP out of the box. It is a fully-featured mail client with built-in calendar, contact management and chat functions. It has a modern user interface which may or may not be preferred to the traditional style.

eM Client is a commercial product and closed-source, but is free-of-charge for non-commercial use (subject to some restrictions which are not onerous). For further information and download, see:

Download: http://www.emclient.com/

Mobile Device Cryptography-aware e-mail Clients *(not tested)*

iOS Mail, the iPhone/iPad e-mail application, supports S/MIME out of the box.

User guide:	https://support.apple.com/en-us/HT202345

iPGMail, a low-cost cryptography app for the iPhone/iPad, integrates with iOS Mail to add OpenPGP support. It appears to be widely-used and well spoken of.

Description:	http://ipgmail.com/
Download:	click on the App Store button in the above page.
Tutorial:	http://www.youtube.com/watch?v=IDSIn0oLbf4

Appendix III – Social Media-based Cryptography Software

(under development – further research and testing needed)

These platforms provide Social Media-based cryptography, principally for mobile devices but also (subject to some limitations) for desktop computers.

Signal Messenger *(not tested)* is a well-regarded platform in the serious cryptography field. It supports Instant Messaging and Group Chat, plus voice and video calls. It is published by Signal Messenger LLC, owned by the non-profit Signal Foundation.

It provides full e*nd-to-end* encryption by default, with optional *forward secrecy.*

Hardware and Software requirements:

– Mandatory: a mobile device with mobile phone network/SMS connectivity; plus

– Optional: a desktop computer.

The telephone number of the mobile device, which must be the actual one on which you are running Signal, serves as your persona. You use it to register with the Signal platform and also to login. When you make contact with another user, you can verify that you are through to the real owner of the telephone number you called – and not a 'man in the middle' – by exchanging and checking *safety numbers* with them.

The Signal software is open source (so it can be inspected by anyone) and passed an independent security audit in 2016. There is a video description[*] of the protocol design. The Signal protocol is the same as that used by WhatsApp.

The software and services are licensed:	
Terms:	https://signal.org/legal/

The software is available free-of-charge:	
Download:	https://www.signal.org/download/

* See the video presentation at: https://www.youtube.com/watch?v=7WnwSovjYMs.

Keybase *(limited testing)* is a serious cryptography platform. It supports instant messaging and group chat. It is published by Keybase LLC which was acquired by Zoom Video Communications, Inc. in May 2020.

It provides full end-to-end encryption by default, with key synchronisation across the user's devices, and optional forward secrecy. It supports the *Networking and Cryptographic Library (NaCl)*[†] (also known as 'salt'), which provides Elliptic Curve Cryptography (ECC) keys; also OpenPGP keys keys (but not integrated e-mail) which enables you to create encrypted and/or digitally signed documents that you can send as e-mail attachments.

Hardware and Software requirements:

- One or more devices which can be desktop computers, mobile devices or a mixture of the two; but
- There is no mandatory requirement for a mobile device.

You register on the Keybase platform with a username and password of your choice. The username serves as your persona. You are advised to set up Keybase on at least two devices and preferably more. If you lose a device or key, you can continue working and authenticate a replacement[*], so long as you still have a device with a non-revoked key; only if you have lost all your keys do you lose access to your account. If this happens, you must reset the account and start again from scratch, but it is a rare event. Keybase does not use safety numbers.

The Keybase software is open sourced under a BSD-style licence and is available free-of-charge from:

Download:	https://keybase.io/

Use of the web-site and services are governed by a separate licence:

Terms:	https://keybase.io/docs/terms/

† See Bernstein, Lange & Schwabe: Securing Communication (https://crabgrass.riseup.net/assets/263828/NaCl+lib+securing-communication.pdf)

* See the Keybase document: Keybase is not softer than TOFU (https://keybase.io/blog/chat-apps-softer-than-tofu)

Telegram *(limited testing)* is aimed mainly at the recreational market. It supports text, video and voice calls. It is published by Telegram Messenger Inc., owned by the Durov brothers and based in Dubai.

It has two modes of operation. The default 'cloud chats' mode uses server-client encryption which gives only limited security. The 'secret chats' mode has full end-to-end encryption. For more information on Secret Chats, see companion paper *How to Do Secret Chats in Telegram*.

Hardware and Software requirements:

- One or more devices which can be desktop computers, mobile devices or a mixture of the two; and
- An SMS capability for authenticating yourself to Telegram by *SMS ping*. If your mobile device does not support SMS, you can use an old-fashioned mobile phone to authenticate instead.

The mobile telephone number serves as your persona. You must provide it to register on the Telegram platform for the first time and again to identify yourself each time you login. For stronger security, you can turn on *Two-Step Verification* and set a password to provide a second form of authentication. Telegram has a similar feature to Signal's safety numbers, but using images rather than numbers.

The Telegram software is partly open and partly closed source, not independently audited. The encryption protocol is Telegram's proprietary MTProto.

Telegram is licensed under Terms of Service and Privacy Policy:

Terms: https://telegram.org/tos/ and
https://telegram.org/privacy/

It is free-of-charge and can be downloaded:

for Android:	https://telegram.org/dl/android/
for iPhone:	https://telegram.org/dl/ios/
for desktop:	https://desktop.telegram.org/ (PC/Mac/Linux)
for macOS:	https://macos.telegram.org/

WhatsApp *(not tested)* is the most popular cryptographic messaging platform, aimed mainly at the recreational market. It supports text, video and voice calls, with full end-to-end encryption by default and optional forward secrecy. It is owned by Facebook.

Hardware and Software requirements:

- Mandatory: a mobile device;
- Mandatory: An SMS capability for authenticating yourself to WhatsApp by *SMS ping*. If your mobile device does not support SMS, you can use an old-fashioned mobile phone to authenticate instead; plus
- Optional: a desktop or laptop computer, so you can have a standard keyboard and screen for working on big documents. You must be logged in via the mobile device throughout your desktop session and will need to run WhatsApp Web or WhatsApp Desktop (see FAQ below) on your desktop computer.

(This section to be completed)

Documentation on WhatsApp is available in:

FAQ:	`https://faq.whatsapp.com/`.

Appendix IV – Compatibility Summary

Cryptographic E-mail Clients

Processor:[1]	**32-bit without SSE2**		**32-bit + SSE2**		**64-bit**
Operating System:[2]	**Windows XP + Service Pack 2**	**Windows XP + Service Pack 3**	**Windows XP + Service Pack 2**	**Windows XP + Service Pack 3**	**Windows 7.0 + Service Pack 1**
Outlook + Gpg4win[3]	Unknown	Unknown	Unknown	Unknown	YES (untested)
Outlook Express	NO	S/MIME only (untested)	NO	S/MIME only (test failed)[4]	Not available
Thunderbird[5] **+ Gpg4win**[3]	NO	NO	OpenPGP only	YES	YES
Claws Mail + Gpg4win[3]	YES	YES	YES	YES	YES
eM Client	NO	NO	NO	NO	YES
iPGMail[6]	Runs on Apple iPhone/iPad under iOS only (untested)				

Notes on the Table

1. SSE2 is a set of additional machine instructions introduced in Intel processors in the early 2000s and in AMD processors slightly later, so may be missing from older 32-bit machines. Some modern software (e.g. Thunderbird) needs them.

2. **Windows XP is no longer maintained and may have security issues, so is not recommended.** If used, it should be updated to Service Pack 3 plus all available post-SP3 updates. Microsoft no longer provide Service Packs for XP.

3. On current Windows versions, use the latest Gpg4win. On Windows XP, use the older Gpg4win version 2.3.3 instead: https://files.gpg4win.org/gpg4win-2.3.3.exe

4. Signed e-mails and encrypted e-mails sent by Outlook Express were displayed successfully by OE and other mail clients. Signed messages and encrypted messages to OE from other mail clients could not be verified or decrypted.

5. Thunderbird requires Enigmail add-on for OpenPGP support.

6. Supports OpenPGP only.

To find out if your machine has SSE2 and the service pack level of its operating system, see companion paper *How to Find Processor and Windows Capabilities.*

Cryptographic Messaging Platforms

Device:	**Mobile**		**Desktop**	
Operating System:	**Android**	**iOS**	**Windows**	**Linux**
Signal	YES (untested)	YES (untested)	YES (untested)[7]	YES (untested)[7]
Keybase	YES	YES (untested)	YES (untested)	YES
Telegram	YES	YES (untested)	YES (untested)[8]	YES[8]
WhatsApp	Unknown	Unknown	Unknown	Unknown

Notes on the Table

7. Requires mobile to be set up first.

8. Does not support Secret Chats.