STEED — Usable End-to-End Encryption

Werner Koch, Marcus Brinkmann

***Abstract*—End-to-end e-mail encryption is still ignored by almost all users. The mails are left in the clear in the mailboxes of the web mail providers, where they are frequently collected by attackers and lead to an escalation of the attack due to the sensitivity of the mail content. We suggest a new and simplified infrastructure to protect mail that is compatible with OpenPGP and S/MIME and relies on an easy-to-use trust model without a central administration.**

***Index Terms*—end-to-end encryption, e-mail security, cryptography, trust model, trust upon first contact, trust on first use, OpenPGP, S/MIME**

FOR several decades, the security community tried to get end-to-end mail encryption into ubiquitous use. From the early history of secure message exchange in the late 1980's, we have steadily built up a technology base that provides strong security guarantees and high interoperability. PEM [1] introduced Base64 encoding in 1987, PGP made strong cryptography available to the general public in 1991 [2], MOSS [3] showed how to use cryptography in MIME body parts in 1995 [4], and with OpenPGP [5] and S/MIME [6] we now have two mature mail encryption and authentication standards widely available.

Yet, ubiquitous use of secure mail has not emerged. While adoption rates for SMTP/POP3/IMAP over TLS are encouraging, the mailbox itself is only weakly protected by passwords, which are easily revealed by phishing attacks [7]. We hardly need to emphasize the personal and business risks that come from a compromised mail account. Also, mail is used as a simple authentication method for many web sites, increasing the exposure of the attacked user even further. The need for end-to-end mail encryption that protects the user's communication from attacks on the communication channel, the service provider, and remote attackers alike seems more urgent than ever before.

As shown in a usability study of PGP 5.0 in [8], the OpenPGP trust model is not intuitive for users and every task related to mail encryption that a user must perform provides significant chance of failure, weakening the security of the system and hindering acceptance among users. Social considerations, such as the lack of perceived value of and risk to one's personal data and the burden that use of security measures impose on others, have also been cited as obstacles to adoption of mail encryption [9]. With no perceived advantage of mail over other communication channels, use of web mail among young people decreases dramatically [10], possibly in favor of broadcasting the information on social networking sites instead.

We propose a new and simplified combination of existing techniques that avoids the known usability pitfalls and aims at providing protection for mail against common attacks:

- Automatic key generation simplifies the configuration setup.
- Automatic key distribution via DNS enables opportunistic encryption, gets rid of key selection and more generally allows the recipient to set his encryption preferences autonomously.
- Opportunistic encryption realizes security-by-default for mail encryption.
- Trust upon first contact and persistence of pseudonym provide a closer match to user expectations and over-all greater utility for the user.

Our approach builds on existing infrastructure and is fully compatible with OpenPGP and S/MIME, whose trust models (WoT and PKIX) may still be used by individual users or organizations with different requirements. Thus it greatly increases the opportunities to send encrypted mail from such entities to less motivated or organized users.

A glossary can be found in Sect. XI for the convenience of the reader.

I. Automatic Key Generation

It has now been recognized for some time that user interaction can lead to a variety of security failures [11]. Consequently, removing unnecessary user interaction reduces the security risk and enables better user interfaces as well. We suggest that the MUA generates key pairs and certificates automatically and associates them with the mail account of the user. The user's name and mail address are already known to the MUA, while the default cryptographic parameters such as algorithm and bit size should be pre-configured by the domain experts and updated together with the software stack.

A major issue that remains is passphrase protection for the key as well as data recovery and mobility. To that, we stipulate that the mail encryption and signing keys are not the only personal information of the user that needs to be protected, stored and distributed. We anticipate that eventually a solution in form of a personal information management (PIM) service will emerge that encompasses all data of the user's personal profile, and that this solution will provide adequate protection for the mail encryption and signing key as well. The MUA needs to tightly integrate to this not just for the cryptographic keys, but also for the mail accounts, address books and other data.

Special consideration needs to be given to S/MIME certificates. Generating such certificates is a particular user-unfriendly multi-step process involving a trusted third party (certificate authority). In our scheme, the MUA generates self-signed certificates locally without consulting the service of any third party, thereby circumventing the whole PKI hierarchy while remaining compatible with the protocol.

After introduction of a certificate, it may at times need to be renewed, for example to update the key size or the

cryptographic algorithms due to progress in technology or mathematics. A simple replacement of the user's certificate would be flagged by the trust system as a possible attack (see Sect. IV). Thus, a key rollover procedure is required in which the new certificate is signed by the old one before it is distributed. Such certificate renewals should be initiated and managed automatically without user interaction, according to policies that are set by the domain experts and updated together with the software stack.

II. Automatic Key Distribution

One challenge in usability of public key cryptosystems is key distribution and retrieval through a public key infrastructure (PKI). Historically, both OpenPGP and S/MIME have come up with unpractical answers to the question in which database and under which name a certificate should be stored, how changes are propagated in the network, and how trust is assigned to the information stored in those databases [12].

We repeat the simple, pragmatic solutions from [13] to address these intractable theoretical problems. Following the PKI design recommendations in [12], certificates are identified by a mandatory mail address, and may also carry a locally meaningful text such as a personal name. This solves the identity problem. Revocation is avoided: The validity of a certificate is given by its presence in the database (online validation). Lastly, as the database for storing and retrieving certificates we propose DNS, which has many useful properties:

- DNS provides decentralization and high availability world-wide.
- Mail addresses split naturally into a user name and a domain name, which fits the existing structure of DNS records.
- The proposal automatically benefits from security improvements to DNS. In particular, DNSSEC disables man-in-the-middle attacks.
- DNS records can be dynamically managed at a fine enough time granularity to match user expectations for all pseudonyms but those lasting for a very short time. But exactly for those non-persistent pseudonyms, security provided by this proposal is already considerably weakened by choice of the trust model (see Sect. IV).

Because certificates can be very long, it is possible to store a fingerprint of the certificate in DNS, along with a URL to the full certificate.

Due to caching, DNS updates may be delayed and thus database entries may appear out of date in the network for some time. In particular, new identities and invalidations may not be immediately visible to all peers. We believe that the advantages by far outweigh this imperfection for typical use patterns in mail communication. The above lookup protocol allows to secure even the initial contact without any user interaction (see Sect. III).

III. Opportunistic Encryption

In [8], Whitten reports that 3 out of 12 test users sent mail accidentially in the clear while exploring the system. This can happen if encryption must be manually enabled by the user. The simple solution is to always encrypt if it is possible, which is easy to do if key generation and distribution is automatic and transparent to the user, as proposed above. Care must be taken to make decryption on the receiving side transparent as well, to overcome social barriers to use of encryption [14].

Garfinkel [15] describes opportunistic mail encryption which provides security by default and transparently for the user. We accept his proposal, with some differences:

- To increase compatibility and acceptance, we do not specify mechanisms dedicated to securing the message header.
- As explained above, we prefer to store the certificate in DNS rather than including it in the message. This has many advantages, such as secure initial contact, more up-to-date certificates, and being able to piggy-back on DNS security measures to exclude man-in-the-middle attacks, all of which are not addressed in [15].
- Instead of using a filter that acts as a transparent SMTP/POP3 proxy, we require each MUA to implement encryption itself. This enables deeper integration for a better user experience: In Garfinkel's proposal, mail is encrypted if it is possible, otherwise the mail is sent in the clear. There is no mechanism to ask the user for feedback in this case. In our proposal, the MUA may implement the same simple policy, or ask the user for feedback interactively for more sophisticated use cases. Also, Garfinkel inserts a + character at the beginning of the decoded subject line to indicate an encrypted mail to the MUA. Any reply to such a mail must then also be encrypted, or it will not be sent. In our proposal, the MUA can implement this policy or more sophisticated ones without resorting to such special header tricks.

IV. TUFC/POP

A major usability barrier in public key cryptosystems is the trust model [8], [14]. The goal is to disable spoofing and man-in-the-middle attacks by verifying that a certificate belongs to the entity (person or organization) described by its user ID. While X.509 defers all trust decisions to third party certificate authorities (CA), OpenPGP implementations commonly rely on a decentralized reputation system, (web of trust, WoT). Both systems require a significant investment by the user: X.509 asks the user to sink money into the artificial certificate market that provides a dubious return [12], while OpenPGP asks the user harder and harder questions about the trustworthiness of peers away from the center of his personal web of trust [14]. The design space for trust models is constrained not only by technical difficulties, such as scalability to billions of IDs, but also must respect the mental model of the user: only a system that provides a natural mapping from user expectation to the trust model has a chance to find user acceptance. The mental model of the CA system is that of a guiding parent: all trust decisions are deferred to a higher authority. The mental model of the WoT is peer recommendation (friend of a friend). Both systems are context free in the sense that they put a single, final, black or

white trust decision before any communication. Neither system utilizes the users own experience with the peer in the context of the communication happening over time.

In contrast, the trust upon first contact (TUFC) model in its simplest form as used by SSH does not presuppose any existing certificate trust infrastructure. The system will simply accept the certificate of the peer as offered at the time of first contact. This is what virtually all users do anyway, when faced with the task to make a trust decision that interrupts their line of work [11]. The certificate is then remembered and verified on all future contacts, which provides a persistency of pseudonym (POP) of the peer for the user. As long as the certificate remains the same, the user can build up trust towards the peer, using all context available to him, including out-of-band communication such as telephone calls, etc. Thus, instead of replacing human judgment, the model enables and supports it.

By evaluating the remembered context of a certificate, the MUA can then give positive (in case of a long track record) or negative (in case of a change of certificate) feedback to the user. If the certificate change of a peer is due to a certificate renewal with key rollover, the trust context of the user for that peer can be transferred to the new certificate automatically.

While the system does not presuppose any external trust infrastructure (it is completely local and thus also decentralized), such external mechanisms can be built to further support and strengthen the model. For example, DNSSEC can provide a strong channel for certificate transfer, and sophisticated monitoring networks can provide a high degree of spatial and temporal persistence, see [16] and Sect. IX.

V. Implementation Notes

The following information notes give advice and suggestions on how the mechanisms above can be realized. Most of this information is relevant to the MUA, but some of it affects the cryptographic back end, too. The use of a personal information management (PIM, [17]) service (such as [18]) that can store, protect, backup and distribute data records such as mail account configurations, key material and other profile data is strongly encouraged and must in fact be stipulated if the user is expected to be able to move smoothly from one device to another while accessing the same remote services. As mail is merely one example of such a remote service, we do not address the issues involving such a PIM service here, nor do we address interoperability issues in exchanging the profile data between different kinds of MUAs.

A. Infrastructure

Mail service providers need to setup their system to support TUFC/POP: At the very least a user must be able to store his public key in their system (i.e. the DNS) and have an opportunity to manage the key.[1]

An automatic protocol between MUAs and the back end needs to be defined to allow for key generation and management. This is required to make the interaction between the user and the mail service provider mostly invisible. A possible way to implement such a protocol in a MUA is by extending IMAP.

A mechanism for key rollover and revocation needs to be defined as well. The latter requires the use of a second channel (e.g. using a phone hot line or SMS based confirmation) in order to avoid denial of service attacks by malware.

[1] Using a separate provider for public key storage has the problem that it again separates mail address and public key.

B. Key Generation

The MUA must check if a certificate is already associated with the mail account, preferably through the PIM service. In the absence of such a service, the MUA may simply consult the DNS to see whether a certificate has already been setup. If it is, and the secret key is available, the setup is complete.

If a certificate is configured, but the secret key is not available on the local machine, the user has failed to copy his complete profile. In this case maybe the complete profile can be retrieved and installed. Otherwise, the user is out of luck and will have to create a new pseudonym (maybe temporarily until he is able to restore his complete profile). He will not be able to read his archived mail and may have to do some extra work to convince his peers about his identity, because a new pseudonym is technically indistinguishable from a compromised account.

The initial and any further pseudonyms are created by the MUA in the background, so the user is not distracted. A standard key will be generated without a way for the user to enter any features of the key. The user specific information such as the name and mail address are already known to the MUA or can be retrieved through the PIM service.

The MUA will be in one of theses states for each configured mail address:

- New mail address configured but no key for it available and key generation has not been started.
- The user has opted not to use or generate a key for the current mail address.
- The key is being generated; the user has already entered a passphrase for it. Alternatively, the passphrase has been generated automatically and stored into a passphrase vault that is part of the PIM service.
- The key has been generated but not been sent to the public key store.
- The key has been sent to the public key store but is not yet publicly available (DNS update delays).
- The key is ready for use.
- The key has been revoked or is in other ways not usable.

C. GnuPG Changes

GnuPG features almost everything required for the new system [19]. However to allow easy integration with the MUA it may be better to move the contact database into GnuPG proper and provide an API in GnuPG and GPGME to allow MUAs to interact with this database.
What we need to implement in detail is:

- A database in GnuPG to record address-key associations observed in the past communication.

- A new API for GPG to maintain this database; optionally add the same API to GPGSM[2].
- A new API for GPGME so that applications can make easy use of the database.
- A new value for GPG's *–trust-model* option to enable the TUFC scheme and small amount of code to implement this trust model.
- A new GPGME API to create a key in the background if it does not exists.

D. MUA Changes

MUAs need to interact with the mail providers when setting up a new mail account. This is required to automatically create a new key or assign an existing key to the account. Thus changes to the mail account setup dialogs are required. Due to the mostly unattended operation of our system, a sufficient mechanism is a check box to disable key generation and a progress bar running during key generation and until the key has been stored by the mail provider's system. The MUA can notify (by mail) the user as soon as the key is available to the public in the DNS.

The function to display a mail needs to tell GPGME the sender's address and render the message in a way to show the verification status as told by GPGME. For best user experience this needs to be done asynchronously so that fast scrolling through mails will not need to wait for the verification result.

MUAs further need to implement a configuration option to disable the TUFC system and to use the crypto functions as found today. Another option to allow the use of both system thereby assigning different trust values to TUFC and PKI verified messages is also desirable to get acceptance by the more traditional members of the crypto community.

In case a full PIM service with integrated backup is not in use, the backup feature of the MUA needs to include the keys. If no backup feature exists at least a regular backup reminder should be displayed by means of an internally generated mail. A simple backup mechanism may be to allow the user to print out the keys on paper, in form of a hex dump with checksums for manual or OCR input (as produced by paperkey [20]) and in form of 2-dimensional barcodes such as DataMatrix or QR codes [21].

VI. Expert Options

Despite that the goal of this system is simplicity, we will only gain acceptance if a few expert options are available:

A. One Key for all Accounts

Compared to the broad user base of mail only a few users need several mail accounts. Our proposal supports this already by creating one key per mail account. Some users might prefer to use the same key for several accounts. One possible reason for this use case is the use of a smartcard which shall by policy only be used for one certificate.

[2]Or maybe a separate back end can handle this.

The system should allow for this use case, which needs to be supported by all clients by allowing previously created keys to be configured and deployed with an account.

The key element to implement this feature is the indirection expressed by PKA. That is that only the fingerprint of the key is associated with the mail address and not a particular key. In addition a hint is given where to find the key (here this hint is required to be able to retrieve the key without external information). Either an URL or a DNS cert record reference may be used in the PKA record for a primary user ID (aliasing). A primary user ID has the advantage that other mail addresses are more loosely tied together; this has advantages for user ID management.

B. Using a PKI

There are two small communities which are used to their PKI models: OpenPGP users sometimes make heavy use of the WoT whereas hierarchical organized groups demand the use of the PKIX (X.509) trust model. They should be allowed to keep on using their particular trust model. A way to implement this is a configuration switch to change the rendering of TUFC protected messages from green to yellow and use only green[3] trust achieved by the WoT or PKIX.

VII. User Interface

Designing an UI for this is a bit of a challenge. However over the last years a lot of experience was been collected in the domain of web browsers. We can build upon this.

VIII. Challenges

For a successfully deployment of such a system it is of paramount importance that major web mail providers support their users by providing the infrastructure to store key information in the DNS using an automated or semi-automated system. Finding incentives for the providers to implement and support this infrastructure may be difficult.

Although the trust model can provide positive and negative feedback to the user, such feedback is likely to be ignored in the current computing environment due to adverse user conditioning in the past decades [11]. To improve user perception in the long term, the quality of the feedback must improve significantly. In particular, we have to eliminate false negatives. We already explained how false negatives due to certificate renewal can be eliminated by automatic key rollover procedures. We expect that a major source of false negatives comes from user mobility, i. e. the use of multiple accounts and devices. Mobility of personal information across user devices and service providers will increasingly become an urgent problem for a wide range of applications. Any solution to this problem in general can also be applied to mail certificates and the trust contexts of the user.

[3]Of course, these colors need to be supported by other indicators like different frame styles or background textures as well.

IX. Related Work

Garfinkel [15] describes opportunistic mail encryption and also uses automatic key generation.

Storing public key information (either the full certificate or a fingerprint) in DNS records is specified in [22]. A simple trust model for OpenPGP based on fingerprint records in DNS is presented in [13]; in contrast, our new proposal uses a more elaborate trust model based on TUFC/POP.

The TUFC/POP trust model was first used by SSH [23], and formally described in [24] under the name "the resurrecting duckling" (but without reference to SSH). In [25], Gutmann applies this security policy to a variety of other use cases under the name "Key Continuity Management (KCM)". The usability of the security model was evaluated in [14] for S/MIME and Outlook Express, with encouraging results. However, Garfinkel suggests to send the key material in-band with the mail, while we propose to use DNS, which allows further improvements such as DNSSEC or temporal and spatial redundancy using a network of monitors [16].

The TUFC/POP principle is first used in [16], where Wendlandt describes how a network of monitoring servers ("notaries") can provide spatial and temporal redundancy on published fingerprints. This information can increase the context in which the user has to make trust decisions. SSH and SSL/TLS follow a star topology where the communication happens between many clients and a central server, while mail communication is much more connected. Nevertheless, the notary system may scale well enough to make it applicable to fingerprints associated with mail addresses, or the same redundancy could be provided by a P2P monitoring network instead.

X. Future Work

To bring the STEED project forward we will write specifications for the management protocols between the MUAs and mail providers, describe the underlying principles in technical terms, prepare a user manual showing how STEED help them to keep their data confidential and come up with implementations of the required software components.

MUAs we plan to support early are Thunderbird/Enigmail, Outlook, Kmail and a webmailer. Our focus will be on widely used MUAs to get early feedback from non-power mail users. However, to ease experiments during development we may also integrate this system in Mutt (a small and well portable non-graphical MUA).

For the management protocols (DNS, backup, key-rollover) we need to research the best way on how to integrate STEED into existing infrastructures.

Readers are welcome to discuss STEED on the GnuPG development mailing list.[4]

XI. Conventions

We define a few commonly used terms to make clear how they are used in this paper:

Certificate A data structure combining a key, a user-id, other meta information and a self-signature.

DNS *Domain Name System* Hierarchical, distributed database to map domain names to IP address and other data. In this paper we exploit the fact that DNS is an essential part of the internet and that it is extensible by means of custom record types.

IMAP *Internet Message Access Protocol* A common protocol to access mail stored on a provider over the internet.

GnuPG *GNU Privacy Guard* A well known implementation of the OpenPGP and S/MIME protocols. It is freely available for almost all operating systems.

GPGME *GnuPG Made Easy* An application library used to access the feature of GnuPG.

Key The actual public key. For example, the public modulus and exponent for the RSA algorithm. In particular no user-id or any kind of self-signature or meta information is part of this data structure. As an exception, the creation date is part of the key in OpenPGP, because it is required to compute the fingerprint.

MUA *Mail User Agent* The mail client software of the user.

PIM *Personal Information Management* The task to collect and organize all personal information of a user.

PKA *Public Key Association* A simplified trust model for GnuPG that relies on DNS records to provide public key information.

PKI *Public Key Infrastructure* A system to identify, store, retrieve and determine the validity of certificates.

PKIX *Public Key Infrastructure for X.509* An X.509 profile developed by the Internet Engineering Task Force (IETF).

POP *Persistence of Pseudonym* Remembering the identity of a peer temporally and/or spatially.

POP3 *Post Office Protocol (version 3)* A common protocol to transfer mail from a provider to the user over the internet.

Public Key The same as *key*.

STEED *Secure Transmission of Encrypted Electronic Data* The name for the described peer based encryption system.

Secret Key Preferred term in OpenPGP over the often used term *Private Key*. It describes the private parameters of a public key algorithm.

SMTP *Simple Mail Transfer Protocol* The internet standard for mail transmission across the internet.

SSH *Secure Shell* A protocol to access remote computers that displaced telnet. It employs trust upon first contact as a simple trust model.

SSL/TLS *Secure Sockets Layer/Transport Layer Security* A cryptographic protocol to secure end-to-end communication across the internet.

TUFC *Trust Upon First Contact* The trust model usually used by Secure Shell (SSH) applications, which is applied to mail certificates in this paper. This model is also known as Trust On First Use (TOFU).

WoT *Web of Trust* The de-facto standard PKI used with OpenPGP.

[4] See http://lists.gnupg.org/mailman/listinfo/gnupg-devel

REFERENCES

[1] J. Linn, "Privacy enhancement for Internet electronic mail: Part I: Message encipherment and authentication procedures," RFC 989, Internet Engineering Task Force, Feb. 1987, obsoleted by RFCs 1040, 1113. [Online]. Available: http://www.ietf.org/rfc/rfc989.txt

[2] P. Zimmermann, "PGP Marks 10th Anniversary," http://www.philzimmermann.com/EN/news/PGP_10thAnniversary.html (retrieved on 28. July 2011).

[3] S. Crocker, N. Freed, J. Galvin, and S. Murphy, "MIME Object Security Services," RFC 1848 (Historic), Internet Engineering Task Force, Oct. 1995. [Online]. Available: http://www.ietf.org/rfc/rfc1848.txt

[4] J. Galvin, S. Murphy, S. Crocker, and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted," RFC 1847 (Proposed Standard), Internet Engineering Task Force, Oct. 1995. [Online]. Available: http://www.ietf.org/rfc/rfc1847.txt

[5] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "OpenPGP Message Format," RFC 4880 (Proposed Standard), Internet Engineering Task Force, Nov. 2007, updated by RFC 5581. [Online]. Available: http://www.ietf.org/rfc/rfc4880.txt

[6] B. Ramsdell, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification," RFC 3851 (Proposed Standard), Internet Engineering Task Force, Jul. 2004, obsoleted by RFC 5751. [Online]. Available: http://www.ietf.org/rfc/rfc3851.txt

[7] D. Florencio and C. Herley, "A large-scale study of web password habits," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 657–666. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242661

[8] A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: a usability evaluation of PGP 5.0," in *Proceedings of the 8th conference on USENIX Security Symposium - Volume 8*. Berkeley, CA, USA: USENIX Association, 1999, pp. 14–14. [Online]. Available: http://portal.acm.org/citation.cfm?id=1251421.1251435

[9] S. Gaw, E. W. Felten, and P. Fernandez-Kelly, "Secrecy, flagging, and paranoia: adoption criteria in encrypted email," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, ser. CHI '06. New York, NY, USA: ACM, 2006, pp. 591–600. [Online]. Available: http://doi.acm.org/10.1145/1124772.1124862

[10] "The 2010 U.S. Digital Year in Review," White Paper, comScore, Feb. 2011.

[11] P. Gutmann, "Security usability fundamentals," http://www.cs.auckland.ac.nz/~pgut001/pubs/usability.pdf (retrieved on 23. August 2011).

[12] ——, "Pki: It's not dead, just resting," *Computer*, vol. 35, pp. 41–49, August 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=619078.622041

[13] W. Koch, "Public key association," in *GUUG Frühjahrsfachgespräche 2006: Proceedings*. Köln, Germany: GUUG, 2006, pp. 159–167. [Online]. Available: http://g10code.com/docs/pka-intro.de.pdf

[14] S. L. Garfinkel and R. C. Miller, "Johnny 2: a user test of key continuity management with s/mime and outlook express," in *Proceedings of the 2005 symposium on Usable privacy and security*, ser. SOUPS '05. New York, NY, USA: ACM, 2005, pp. 13–24. [Online]. Available: http://doi.acm.org/10.1145/1073001.1073003

[15] S. L. Garfinkel, "Enabling email confidentiality through the use of opportunistic encryption," in *Proceedings of the 2003 annual national conference on Digital government research*, ser. dg.o '03. Digital Government Society of North America, 2003, pp. 1–4. [Online]. Available: http://portal.acm.org/citation.cfm?id=1123196.1123245

[16] D. Wendlandt, D. Andersen, and A. Perrig, "Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing," in *Proc. USENIX Annual Technical Conference*, Boston, MA, Jun. 2008.

[17] O. B. Tel, O. Bergman, and R. Boardman, "Personal information management," in *Extended Abstracts of the 2004 ACM Conference on Human Factors and Computing Systems*. ACM Press, 2004, pp. 1598–1599.

[18] T. Adam and M. Boehm, "When the bazaar sets out to build cathedrals," in *Beautiful Architecture*, M. Treseler, Ed. O'Reilly Media, 2009, ch. 12, pp. 279–311.

[19] D. Mahoney, "The complete guide to publishing PGP keys in DNS," http://www.gushi.org/make-dns-cert/HOWTO.html (retrieved on 5. July 2011).

[20] D. Shaw, "Paperkey - an OpenPGP key archiver," http://www.jabberwocky.com/software/paperkey/ (retrieved on 30. August 2011).

[21] T. Jost, "HOWTO Backup your GnuPG secret key on paper," http://schnouki.net/2010/03/22/howto-backup-your-gnupg-secret-key-on-paper/ (retrieved on 30. August 2011).

[22] S. Josefsson, "Storing Certificates in the Domain Name System (DNS)," RFC 4398 (Proposed Standard), Internet Engineering Task Force, Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4398.txt

[23] T. Ylönen, "Ssh: secure login connections over the internet," in *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography - Volume 6*. Berkeley, CA, USA: USENIX Association, 1996, pp. 4–4. [Online]. Available: http://portal.acm.org/citation.cfm?id=1267569.1267573

[24] F. Stajano and R. J. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Proceedings of the 7th International Workshop on Security Protocols*. London, UK: Springer-Verlag, 2000, pp. 172–194. [Online]. Available: http://portal.acm.org/citation.cfm?id=647217.760118

[25] P. Gutmann, "Why Isn't the Internet Secure Yet, Dammit?" in *AusCERT Asia Pacific Information Technology Security Conference 2004; Computer Security: Are we there yet?* AusCERT, May 2004. [Online]. Available: http://www.cs.auckland.ac.nz/~pgut001/pubs/dammit.pdf

Werner Koch is the principal author of GnuPG, Free Software activist, and Business Manager at g10code GmbH. Contact him at wk@g10code.com.

Marcus Brinkmann holds a diploma degree in mathematics from the Ruhr University Bochum and is Software Architect at g10code GmbH. Contact him at mb@g10code.com.