Dark Internet Mail Environment

Linus Lotz
Tutor: Oliver Gasser
Seminar Future Internet SS2015
Chair for Network Architectures and Services
Department of Computer Science, Technische Universität München
E-mail: lotz@in.tum.de

ABSTRACT
The current mail system is still very important for communcation today, but does not integrate End-to-End encryption and a lot of metadata is leaked during transit. This paper will have a look at Dark Internet Mail Environment (DIME) developed by Ladar Levison.

DIME tries to address these issues by using a new encrypted mail format called D/MIME to provide End-to-End encryption and removing the identifiying information from the protocol conversations in order to reduce the visible metadata. We close with a comparision of DIME with other systems.

Keywords
Dark Mail, DIME, DNSSEC, E-mail, Privacy, Metadata

1. INTRODUCTION
The Dark Mail initiative was founded by Ladar Levison. He was the founder of Lavabit, an e-mail service, that stored e-mails encrypted on the server. After Edward Snowden revealed himself, Lavabit was forced to reveal it's TLS private keys.

In order to protect his users Ladar Levison chose to shut down his service. Shortly after he announced that he had teamed up with Silent Circle, a company founded by Phil Zimmerman (author of PGP and ZRTP). He started a very successful kickstarter campaign for Dark Mail and in December 2014 the first public documentation was released. This documentation describes the Dark Internet Mail Environment (DIME), which we present in this paper.

We start with an detailed overview of DIME in Section 2. In Section 3 we will have a look at the attacker model and the security promised by DIME. Related work will be presented in Section 4 and we will conclude in Section 5.

2. ARCHITECTURE OF DARK MAIL
The Dark Internet Mail Environment[13] consists of several parts. It uses a new Public Key Infrastructure that will be described in section 2.1. The new DIME message format called D/MIME is described in Section 2.2. For transporting and retrieving messages DMTP and DMAP are used, which are described in Sections 2.3 and 2.4.

2.1 Public Key Infrastructure
The DIME specification defines a new certificate called signet, which is used for users and domains. These signets are verified by a Primary Organization Key (POK) or Secondary Organization Key. The last two keys are stored in a DNS record.

2.1.1 Primary Organization Key
The Primary Organization Key (POK) can sign the organizational signets, outgoing mails and user signets for this organization.

2.1.2 Secondary Organization Key
The Secondary Organization Key (SOK) is used to sign the outgoing mails and user signets. When using a Secondary Organization Key, it is possible to keep the POK offline, so it cannot be compromised so easily.

The offline POK would then be used to sign new organizational signets. The SOK would be on the live system to sign outgoing mails. If the SOK is compromised, it cannot be used to create another organizational signet.

2.1.3 Signets
The DIME specification also specifies a simple certificate format for user and domain keys. They are designed so they fit their needs and are very simple. There are two kinds of signets: user signets and organization signets. The user signet is signed with the key from the current organization signet. Every user signet needs to be signed by the POK or an authorized SOK and an organization signet needs to be signed by the POK.

2.1.4 Organization Signets
The organization signets are required to contain the POK and a secp256k1 key which is used to encrypt informations that should be visible to the mailserver. It may also include the hosts for web and mail access and their respective TLS public key fingerprints.

It is also possible to add an onion access host, for access over TOR. The signet can also contain contact addresses for 'abuse', 'admin' or 'support'.

2.1.5 User Signets
Every time a user generates a new key, he signs it with his previous key. This creates a chain of keys, where each key

doi: 10.2313/NET-2015-09-1_03

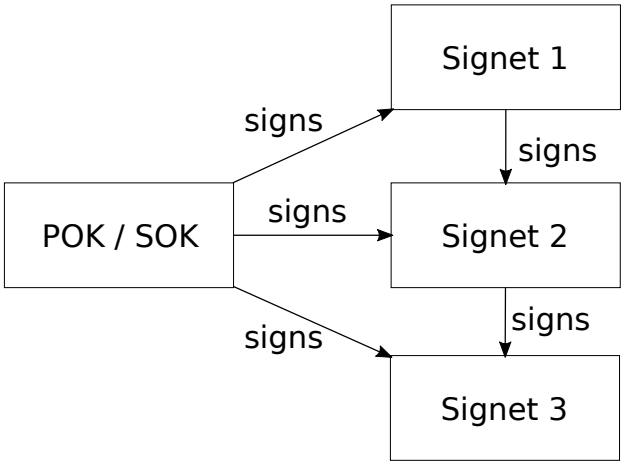

Figure 1: The signing of new user signets

is signed by it's predecessor. The key is then sent to the mailserver, which then signs it with the POK or SOK. The resulting signatures can be seen in Figure 1.

After verifying a key in this chain manually it is possible to verify future keys automatically. All public keys are stored on the mailserver, which can be asked for the keys. This makes it easy to replace the current key regularily. This is a very simple form of Forward Secrecy.

The specification does not describe what happens, if the user loses his key at one point. We assume, that the user has to create a new chain of keys, which can be verified by the server. How other users would switch to this new chain is unclear.

2.1.6 DNS record

DIME uses DNS to get a special DIME record or a `_dime` TXT record [13, chap. 4]. This record contains a Primary Organization Key and optionally a Secondary Organization Key. It is also used to specify the delivery host, which is responsible for accepting messages and providing the signets for this domain.

The record also allows to specify a "policy" setting. This policy setting determines if the domain either accepts Dark Mail or legacy mail or both.

Another special host that can be specified in the record is a so called "syndicate". This host can be used to retrieve signet information for validation.

A "sub" field is used to set a policy for subdomains, which can either allow different DIME records or disallow them. This setting also determines whether subdomains should use the same information as a parent or should be considered as not DIME capable, if no record exists.

An expiery field is used to tell how long a domain is considered DIME enabled even after the record was removed. The TTL is used to set when the record should be refreshed.

An example record could look like this[13, p. 32]:

```
ver=1 pok=MmprdmRjaWtjOTYyZDllMGhrOGNhMTRsZmoyam
t2ZGM pol=mixed
syn=mirror.example.tld
tls=QUYxRjA0MkZDMjQ0OUEzOUJENEE5QkU2MTdENDM3OUV
EQTI1QjQ1REYwODEwODE2ODlGMUE2Q0U1MjQ3M0Y2Mw
dx=dmtp.domain.tld ttl=1776 exp=30 sub=strict
```

Using DNSSEC is encouraged to secure these records to prevent an attacker from performing a downgrade attack. An attacker, who is able to change a DNS response for a DIME record to an NXDOMAIN response, could force a fallback to normal SMTP when DNSSEC is not used. [13, p. 27]

2.1.7 TLS Public Key

The TLS public key fingerprint can be added to the DIME record, which is used in conjunction with DNSSEC to authenticate the certificate. If the record is DNSSEC secured the TLS certificate can be self-signed or signed by an unknown or untrusted CA[13, p. 29].

As a fallback, a valid certificate that matches the hostname and is from a trusted CA is accepted too. This makes it possible to deploy DIME, even if the domain is not secured by DNSSEC.

2.1.8 User Key Management

Dark Mail differentiates between different account modes, which differ in how much the user keys are exposed to the service provider.

In the lowest level "Trustful", the keys are stored on the server and might be encrypted with the user password. The advantage of this procedure is, that any legacy client can be used to access the mails and send mails and the service provider handles all of the Dark Mail encryption. In this case the e-mail content is visible to the service provider.

In the stronger "Cautious" level, the service provider has copies of the encrypted user keys. In this mode a legacy client can not be used, but the message content is not known to the service provider unless he can decrypt the key or sends wrong signets as a Man-in-the-Middle attack.

The strongest level is called "Paranoid" in this case, the provider never has access to the private keys, not even in their encrypted form. If the user regularily rotates his keys and deletes the old ones, he can attain a very simple form of perfect forward secrecy, since he was the only one who had access to the private keys.

2.2 Message Data Format (D/MIME)

D/MIME[13, chap. 6] is a new binary format for storing mails. It is the DIME alternative to MIME[6], the mail format used by e-mails today. D/MIME divides the message in chunks (Figure 3), which get encrypted seperately (Figure 2).

Each encrypted payload (Figure 4) is encrypted with a random 256 bit key and AES-CBC. Messages also contain an ephemeral ECDH key in an unencrypted chunk ("Ephemeral" in Figure 2).

doi: 10.2313/NET-2015-09-1_03

Envelope	Tracing	Unencrypted
	Ephemeral Origin	Unencrypted AOR
	Destination	ADR
Metadata	Common Fields	AR
	Other Fields	AR
Content	Text	AR
	Attachment	AR
Signature	User	AOR
	Organization	AODR

Figure 2: The example of an D/MIME message with access: A=Author, R=Recipient, D=Destination organization, O=Origin organization

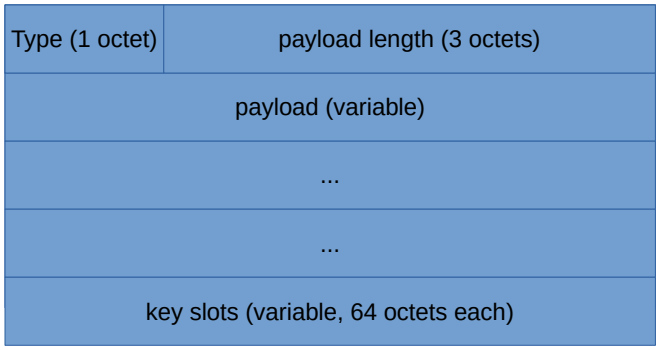

Figure 3: Layout of a chunk in an D/MIME message

This key is used together with the key from an actor (recipient user signet or recipient domain organization signet) to generate a Key Encryption Key (KEK) for this actor:
Sending:

$$\text{KEK} = ECDH(Ephemeral_{private}, Actor_{public})$$

Recieving:

$$\text{KEK} = ECDH(Actor_{private}, Ephemeral_{public})$$

The Initialisation Vector (IV) used for the AES CBC is XORed with 16 bytes of random data, which are encrypted with the KEK along with the result of the XOR and the 256bit key. This makes the first 32 byte different for each keyslot, before encrypting them. The idea is "to prevent a variety of known, and future differential cryptanalysis attacks"[13, p.61]. The result (Figure 5) is then stored in the keyslot of that actor. Using this mechanism the sender can limit, which parts of the message are visible to each actor.

At the end of a D/MIME message, it is signed by both the user and the organization. This is done by signing each hash of the chunks and then the complete content. An example message can be seen in Figure 2.

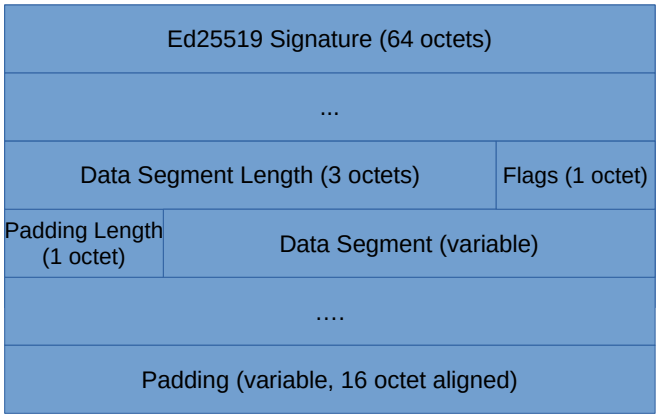

Figure 4: Layout of an encrypted payload

A D/MIME Message only has one recipient, which means that for each recipient the sender has to create a seperate D/MIME message. Since the recipient is stored in a chunk, which can only be read by the destination domain, the actual recipient of the message is hidden to third parties. If an attacker breaks the TLS encryption, he would still need to break the encryption of the D/MIME message to recover this information.

Another benefit of the message format with included encryption and signature is, that it does not suffer from the problems described in [9]. In PGP it is prossible to forward signed messages to a third party and pretend the original signer was the origin and the new recipient the original recipient.

Consider the following scenario: Alice sends Bob a mail which is signed by her and encrypted for Bob:

$$\{\{\text{"Company secret"}\}_{sigAlice}\}_B$$

Bob can now decrypt this message and has now the signed message from Alice. He can create a message which has the "From:" header set to Alice's e-mail, reencrypt the message for Charlie and send it to him:

$$\{\{\text{"Company secret"}\}_{sigAlice}\}_C$$

To Charlie it looks like Alice sent him the message, as she signed the message. This is possible, because the headers in MIME are not signed in PGP.

In D/MIME the complete message is signed, including the sender and recipient used, when sending the e-mail. Forwarding such a mail would change these fields and thus require to resign the message. This is a big improvement over PGP.

Since all D/MIME messages are signed by the organization, faking the origin domain is not as easy as in regular MIME.

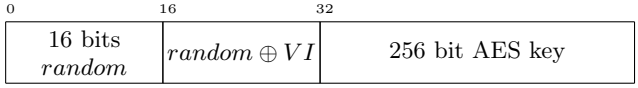

Figure 5: Structure of a unencrypted Keyslot

doi: 10.2313/NET-2015-09-1_03

2.3 DMTP

In the classical e-mail system, mails are sent by the user using an Mail User Agent(MUA) collects the mails from the user and hands them over to a Mail Transfer Agent(MTA)[8]. This MTA then uses SMTP to transfer the message to the destination domain.

DMTP is the replacement for SMTP[8] in the Dark Internet Mail Environment. It is very similar to SMTP but avoids using account names in the conversation. The idea is to remove the information which users are communicating from the protocol conversation. The recipient server can decrypt the chunk which contains the recipient and thus knows to which user the message should be delivered.

The specification describes how an SMTP connection advertises a DMTP capable mail server and how the connection can be upgraded to DMTP with a special STARTTLS command. All DMTP traffic is protected with TLS 1.2 with ephemeral (Elliptic Curve) Diffie-Hellman keys to provide Perfect Forward Secrecy.

DMTP is also used to request the user and organization signets for this domain. For the user signets it is possible to request all signets, that were ever signed by this organization.

Mails are only exchanged between MTAs over DMTP, users who want to send e-mails use DMAP instead. This is different from the current mail architecture, where the user sends the mails using SMTP.

2.4 DMAP

When an e-mail reches the mailbox of the user, he can retrieve it from a Mail Delivery Agent (MDA). In the current mail environment, there are different protocols which are used in this step. One of the more popular protocols is IMAP[4], which will be the model for DMAP [13, chap. 8].

This protocol will be similar to IMAP[4] but without server side search capabilities. The server side search would require the server to have access to the plaintext, which would defeat the purpose of using DMAP for having true End-to-End encryption. If a user creates a new key, he can ask the server over DMAP to sign it for him. When sending a mail over DMAP the server signs the message before handing it over to the MTA. The authentication will be done using a "cryptographic key process"[13, chap. 8]

When sending e-mails, mails are transfered via SMTP to the MTA. To preserve a copy of the sent mail, the client can store a copy on the IMAP server, but this requires retransmitting the same e-mail over IMAP. Using IMAP for sending mails was tried before [5][1], but is not widespread yet. Using DMAP to send mails is an improvement over the current protocols and seperates server-to-server communication and server-to-client communication.

The DMAP protocol is not specified currently and will probably be released with a later specification.

2.5 Protocol Stack Example

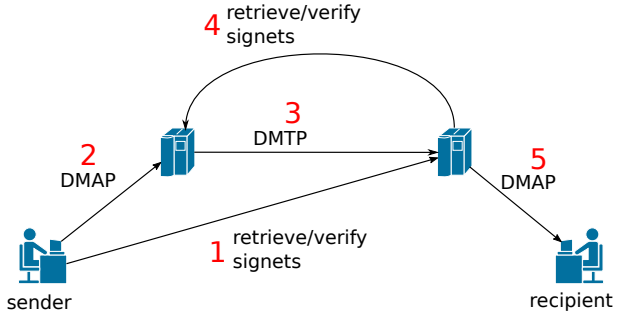

Figure 6: An example message transfer

To illustrate how DIME works, we will show an example where Alice on domain alice.com wants to send a message to bob at bob.com (Figure 6). First Alice, needs to check if Alice has the latest user signet of Bob. She contacts the server for bob.com and asks for the fingerprint of bobs latest key. If she has a different key, she can now ask the server to send her every key between the last key Alice knew and the curent key. She also requests the latest organization signet of bob.com.

After receiving these keys, she needs to verify, that each key signs the next one in the chain and that the signet is signed correctly by the POK or an SOK (where applicable) of bob.com. Now that she has the current key, she can encrypt the chunks and send it via DMAP to alice.com. The MTA on alice.com opens a connection to bob.com with DMTP and transfers the e-mail.

On bob.com, the server needs to verify the signature from alice.com. If he does not already have a matching signet, he must request it from alice.com and verify the signature is from the POK, that is stored in the DNS for alice.com. If the signature is valid and by a valid signet, he can now further process the message.

Now bob.com decrypts the destination chunk. This chunk contains the user that is the recipient on bob.com, in our case Bob. Now that the MTA of bob.com knows the user, it can deliver the mail to his inbox. To receive the message, bob opens a DMAP connection to bob.com and retrieves the message to his local computer. He can now decrypt the chunks and read Alice's e-mail.

The message contains the fingerprint of the signet Alice used for signing. To verify the signature, Bob can check if he already has this key. If not, he contacts the DMTP service for alice.com and asks for the signet with this fingerprint (if he already had a signet from Alice, he would request the signets in between as well).

3. SECURITY ANALYSIS

In the following section we will have a look at the security goals of DIME and the security decisions that were taken.

3.1 Security Goals

Dark Mail Specification states their goals are as follows:"

doi: 10.2313/NET-2015-09-1_03

1. Automate key management, which includes: creation, rotation, discovery and validation
2. Transparently encrypt and sign messages to ensure content confidentiality
3. Ensure the system is resistant to manipulation by Advanced Persistent Threats (APTs)
4. Link security to the complexity of a user's password, and the strength of an endpoint's defenses
5. Minimize the exposure of metadata
6. Give control back to the user"[13, chap. 3]

The current specification of DIME seems to have solved points one and four. Since the attack surface is smaller than with regular e-mail, it is more resilient to APTs. If an attacker had control over the mailserver in the classical mail environment, he would have access to all the metadata and content of the messages. Since the messages are End-to-End encrypted and the metadata is better protected, an attacker would gain mucch less from this kind of access. But this also depends on the account security, which can be controlled by the user (see Section 2.1.8). In practise it will also be improved by a widespread usage of DIME.

The DMAP specification is not released yet and it is the protocol the MUA uses to communicate with the mail server. This is where the password would be used to authenticate the user and to decrypt the encryption keys, if they are stored on the server. Until this part is specified, not much can be said about the attainment of goal three.

Goals two and six also depend on a widespread adoptation of DIME, since otherwise the legacy protocols would be used. These goals are achieved through the D/MIME format, when used with End-to-End encryption.

3.2 DNSSEC

DNSSEC is an extension to the DNS system that adds signing of DNS records to prevent forging of DNS records.

3.2.1 Architecture

DNSSEC adds several new Resource Records (RRs) to each zone. Each zone has two keys, a Key Signing Key (KSK) and a Zone Signing Key (ZSK). The hash of the KSK is stored in a DS record of the parent domain. The Key Signing Key is used to sign the DNSKEY records, where the ZSKs are stored. Every other record is signed by the ZSK. This makes it possible to frequently change the ZSK while keeping the KSK private key offline.

To prevent an attacker from simply sending NXDOMAIN (Domain does not exisist) responses, DNSSEC introduces a record type "NSEC", which points to the next secure record. If the client requests an unknown domain, the server sends this record to prove that it does not exist.

DANE[2] proposes an additional TLSA record, which is used to verify TLS certificates. For it to be trusted by an application, it has to be secured with DNSSEC. It can be used as an additional verification or as an alternate trust-anchor. It's use is not widespread, as DNSSEC secured zones are still very rare.

The TLS certificate validation idea in DIME is very similar to that of DANE. DANE offers more flexibility, since it is also possible to specify the CA, that issued the certificate.

3.2.2 Problems

DNSSEC's trustmodel assumes that you can trust the root zone and the Top Level Domains, which might not be the case. It seems to be very likely, that an intelligence service like the NSA, can get access to the DNSKEYs of the TLDs, that are controlled by the US, like ".com". With this key they are able to forge DNS responses, that are trusted for any domain in ".com".

Even though Eliptic Curve Cryptography in DNSSEC is specified, it is not required to be implemented. RSA is the only algorithm, that is mandatory to implement. RSA has the disadvantage that it's key sizes are relatively large and DNS has a limit for each message, effectively limiting the key size for DNSSEC.

The deployment of DNSSEC requires the use of EDNS, which allows bigger DNS packets than 512 bytes. With larger keysizes the response packets are getting close to 1500 bytes and would usually get fragmented in order to be transmitted.

3.3 Certificate Authentication

In DIME it is possible to verify the certificate with the DIME DNS record or by using CAs.

3.3.1 Certificate Authentication with DNSSEC

Dark Mail uses DNSSEC to verify TLS certificates and the POK via DNS. This involves some disadvantages. Each country code top level domain (ccTLD) is usually controlled by the respective country. This would include the DNSSEC keys. It would be very easy for a country to fake any DNS request for any domain in their ccTLD. It seems very likely that an intelligence agency is able to get these keys in their possession.

3.3.2 Certificate Authentication with CAs

As a fallback it is still possible to use a X.509 certificate issued by a trusted Certificate Authority. This makes it possible to use DIME on domains that are not secured with DNSSEC. For an attacker like the NSA, getting a valid certificate is not infeasible. For example Stuxnet is believed to be created by the American, and Isreali intelligence service had a valid signature for it's Windows kernel driver.

3.4 User Authentication

The specification suggests using a "pre-nounced hash" to store the password. The password should be used to authenticate to the server but should never be sent to the server. This way an attacker can not get the password by obtaining the account information.

If the account password is not sent over the wire it can be used to encrypt the private keys, if they are stored on the

doi: 10.2313/NET-2015-09-1_03

server. When loosing this password however, the encryption keys would be inevitably lost.

3.5 Metadata and Privacy

The D/MIME format allows to hide the information from the mail servers, if it is not necessary for them to see it. To see the complete metadata, an attack would either need the private keys of both servers or the private key of the sender or recipient.

Even though a lot of metadata is hidden through the DIME format, it is only beneficial if a domain has many users. If a user would like to setup his own server for his domain, any mail from or to his server can still be linked to him. The signets contain fields for onion addresses, so the transport could additionally go over TOR. This could make it more difficult to tell from the connections between the servers which were communicating.

4. RELATED WORK

There are several systems which try to achieve similar goals as DIME, some are presented in the following section.

4.1 Pretty Good Privacy (PGP)

PGP was developed in 1991 and is capable of encrypting and signing messages. It's main advantage over DIME is, that no change in the infrastructure is needed. Two people who want to exchange encrypted messages, can just install a PGP client and send encrypted messages after exchanging keys. The current Message Format Specification can be found in [7].

There are so called PGP keyservers which provide a database of PGP keys, which can be queried. They don't require any authentication so anyone can send a PGP key for any e-mail address. It is possible to query the server for specific key fingerprints.

Dark Mail was designed to fix some of the flaws of PGP. While Dark Mail tries to protect metadata, PGP only encrypts the message text, leaving the metadata in the open. The user signets from Dark Mail are used to verify, that the key for an e-mail address belongs to the account, which makes it harder to forge a key and simplifies the key validation.

This has to be done manually in PGP. Furthermore, combining this with signing the new key with the previous key, when rotating keys, allows to frequently change keys without any real drawbacks. Doing so creates a simple form of forward secrecy. Adding forward secrecy to PGP was tried before[11], but was not successful so far.

4.2 S/MIME

The Secure/Multipurpose Internet Mail Extensions[3](S/MIME) are another solution for encrypting e-mails. Instead of a Web of Trust, like in PGP, certificate authorities are used. In order to sign messages or receive encrypted messages, a user must apply for a certificate from a Certificate Authority. There are some Certificate Authorities that issue them for free.

Most modern mail clients include support for S/MIME. It's advantage over PGP is, that it does not require any user interaction when trusting other certificates.

4.3 Bitmessage

Bitmessage is a P2P messaging system, that tries to provide sender and receiver anonymity. Messages are sent by broadcasting them in the network, which tries to hide the origin of a message.

The message is stored in the network for a certain time, after which it is deleted. To prevent spam flooding the network, every message needs a proof of work. The messages don't disclose the recipient, so every bitmessage client has to try to decrypt every message in the network. The messages are stored in a so called blockchain, similar to that of Bitcoin.

4.4 Pond

Pond[12] is a asynchronous messaging system similar to e-mail with very strong anonymity. Message transmissions are done over TOR[10] and always use the same amount of traffic, to complicate traffic analysis.

Before a message can be sent to someone, that person has to add the sender to a groupkey on the server, which is done to prevent spam.

4.5 Other Solutions

A very exhaustive list of projects trying to provide secure email can be found in [14].

5. CONCLUSION

Overall the Dark Mail initiative is a good improvement compared to the existing mail infrastructure. It allows for a transparent transition, to a more privacy preserving infrastructure.

The D/MIME format allows to hide information from actors, which they don't need to know and the signature protects the complete message including the metadata and not only the content.

The DMTP protocol is designed to leak less metadata and forces TLS encryption.

To be successful, it will take time, since all the exisiting e-mail software needs to be modified to support DIME completely. On a security perspective the strong trust in DNSSEC seems to contradict it's security goal and should be reconsidered. It would be better if additionally another trust anchor would be required.

References

[1] The Internet Email to Support Diverse Service Environments (Lemonade) Profile. RFC 5550, Internet Engineering Task Force, August 2009.

[2] The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, Internet Engineering Task Force, August 2012.

[3] Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751, Internet Engineering Task Force, January 2010.

[4] INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1. RFC 3501, Internet Engineering Task Force, March 2003.

[5] Push Extensions to the IMAP Protocol (P-IMAP). Draft, Internet Engineering Task Force, March 2006. URL `https://tools.ietf.org/html/draft-maes-lemonade-p-imap-12`.

[6] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, Internet Engineering Task Force, November 1996.

[7] OpenPGP Message Format. RFC 4880, Internet Engineering Task Force, November 2007.

[8] Simple Mail Transfer Protocol. RFC 5321, Internet Engineering Task Force, October 2008.

[9] D. Davis. Defective Sign & Encrypt in S/MIME, PKCS#7, MOSS, PEM, PGP, and XML. In *Proceedings of the 2001 USENIX Annual Technical Conference, June 25-30, 2001, Boston, Massachusetts, USA.*, pages 65–78.

[10] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router, 2004.

[11] Internet Engineering Task Force. Forward Secrecy Extensions for OpenPGP, 2001. URL `http://tools.ietf.org/html/draft-brown-pgp-pfs-03`.

[12] A. Langley. Pond Technical Information. URL `https://pond.imperialviolet.org/tech.html`. [Accessed March 30th, 2015].

[13] L. Levison. Dark Internet Mail Environment: Architecture and Specifications, 2014. URL `https://darkmail.info/downloads/dark-internet-mail-environment-december-2014.pdf`. [Accessed March 30th, 2015].

[14] Open Tech Fund. Overview of projects working on next-generation secure email, 22.7.2014. URL `https://github.com/OpenTechFund/secure-email`. [Accessed March 30th, 2015].

doi: 10.2313/NET-2015-09-1_03