Enforcing Email Addresses Privacy using Tokens

Roman Schlegel and Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. We propose a system which allows users to monitor how their email addresses are used and how they spread over the Internet. This protects the privacy of the user and can reduce the SPAM phenomenon. Our solution does not require changes to the email infrastructure, can be set up by the end user on an individual basis and is compatible with any email client as long as emails are centralized on a server (e.g. an IMAP server). Nevertheless, it requires that people use email messaging quite differently.

1 Towards a Fair Competition between Humans and Robots

Anyone can send an email to a given address (which is just a simple string) at basically no cost. Those strings used to be systematically exchanged until robots collected them automatically in every public electronic discussion forum and used them for spamming. That is why today "poor" human users have to face armies of well trained robots which are launched by associations of hackers and spammers. This situation is obviously unfair.

Current counter-spam solutions are based on one of the following techniques.

- Deterministic filters using well configured rules.
- Filters based on artificial intelligence or evolutionary processes (which learn to recognize regular email senders).
- Collaborative filtering, e.g. by polling or Bayesian methods.
- Charging policies to render sending emails less cheap.
- More interactive protocols to prove that an email sender is a real sender.

The last two approaches require changes to the current email infrastructure. The first one is obviously limited and the second one leads to false positive and false negative alerts. The third one only works in the case where spam is sent to a large number of email users within the same community. All approaches are eventually defeated by spammers.

On a separate subject, web page addresses — URLs — are widely exchanged, anyone can access a URL for free, but the HTTP server (a kind of a robot itself) can monitor which (human) client is requesting which URL by using cookies. Indeed, cookies are a privacy threat to legitimate (human) web surfers. So it

seems that machines have powerful audit tools at their disposal which could be used by humans as well. Actually, cookie-like strings could be a solution to strengthen the privacy of email addresses. Indeed, by appending a string (that we actually call a *token*), we can monitor how email addresses spread.

Here is a typical scenario. A user Alice would like to receive emails from another user Bob, so she gives him her email address together with a private token. Incoming emails using this token and not coming from Bob's address can simply be ignored. If a spammer manages to intercept the token and spoof emails as coming from Bob, Alice can easily deactivate the token. Another case is when Bob abuses his privilege to send emails to Alice by sending unsolicited emails. Alice can deactivate the token as well.

In this paper we present an application called XToken[1] which is a first step towards implementing the solution described above. In Section 3 we explain the various possible tokens. Section 4 tells how an email sender should first get a token by using a token distributor. Then Section 5 presents a signature-based alternative which works provided that all users use it. Finally Section 6 proposes an agenda on how to deploy this solution and discusses how spammers may recycle their activities.

2 Previous Works

In recent years a lot of different proposals have been made to defeat spam but none of them proved to be a panacea so far. In addition to that not all proposals have yet been implemented on a larger scale. These include for example (non-exhaustive list):

Filtering: Well-defined deterministic rules or AI techniques can be used to identify unsolicited emails.

Domainkeys: This system uses public key cryptography to sign all messages sent from a specific domain. The recipient can verify the signature by looking up the public key using a DNS query. This system does not prevent abuse of the email system but it makes tracking easier. Domainkeys was submitted as a draft to the IETF [7].

Greylisting: Another technique which has been introduced recently with some success is greylisting. The destination mail server will reject a message for the first 5 minutes. The originating mail server will retry after about 15 minutes and will succeed the second time. The idea behind this technique is that a spam software will not normally try to deliver a message more than once but will move to the next address after a failed attempt. Greylisting was first proposed in 2003 [8].

Micro-payments: This method tries to address the fact which makes spam possible, namely the problem that the cost of sending millions of messages

[1] the application is distributed under the GNU General Public License and can be downloaded from http://xtoken.sourceforge.net. A complete manual is available as part of the distribution.

is negligible with the current, widespread broadband technology. Micropayments would charge a small fee for every message sent, making it economically infeasible to send spam. Alternatively the price can be paid in terms of computation resources, meaning the sender has to demonstrate that some energy was spent in order to send the email [4].

Challenge-response: The so-called challenge-response mechanisms are a technique which has been known for quite some time though not necessarily in relation to spam. Applied to email this means that in order to be able to send a message to a recipient, a challenge has to be solved first. Because this normally requires the exchange of three messages (request, challenge, response + email) the concept of pre-challenges has been devised which allows to send messages without requesting a challenge first. The idea is that an already publicly known challenge (which changes periodically) has to be solved to be able to send a message. This reduces the number of required messages between recipient and sender to one like in normal email [6].

SPA: A concept proposed by John Ioannidis from AT&T Labs called SPA (Single-Purpose Address) [9] is in fact quite similar to the concept proposed in this paper. The idea behind SPA is to encode a security policy into an email address. This policy describes the acceptable use of the address and is enforced by the receiver of the message (because the sender cannot be trusted). For a comparison of SPA and XToken see section 7.

Another often mentioned technology in relation to spam is *captcha* (Completely Automated Public Turing Test to Tell Computer and Humans Apart) [3]. It is not a technology to counter spam *per se* but nowadays it is often used to prevent bots from accessing certain information or pages. The idea of captchas is that they present a problem which is easy to solve for a human but very difficult for a computer program. Captchas exist in different forms: they can contain a transformed text on a special background, show different images together with a semantic challenge or they can be based on audio.

Typical examples are web email providers which require the user to solve a captcha when opening a new account to prevent spam bots from using the service.

Captchas in themselves do not solve the problem of spam, but they can be used as one part of a wider system as it will be explained below.

3 Fighting Spam using XToken

3.1 The Token Concept

XToken uses little pieces of information (so-called tokens) to distinguish legitimate from unsolicited emails. Each user has a collection of tokens which he can distribute to his friends and associates. They can include these tokens when sending emails and thus mark them as legitimate (referred to as "valid" hereafter).

The user can choose whether to create one token per person or send the same token to several people. This is obviously a trade-off: One token per person permits fine-grained control over who can send messages while increasing the time needed for managing the tokens. To give one token to a group of people reduces the management overhead but limits the resolution of control to the whole group.

Technically a token is a combination of identifiers which references a token stored on the local computer of the recipient. The length of a token is usually between 6 and 9 bytes although to be able to transmit the token using the email system it is encoded in base64 which increases the length to at least 8 bytes. The base64 representation is then enclosed in '$' signs and added to a field in the email message. Possible fields include the `To:`, `CC:` and the subject header. A typical token looks as follows:

```
$DAdb12wD$
```

The easiest way to use a token received from a friend or associate is by including it in the email address stored in the address book. An entry would then look like this:

```
John Doe $DAdb12wD$ <john.doe@somedomain.net>
```

When writing an email message and selecting the address from the address book the token will automatically be included in the message. Temporary tokens are best included in the subject line like this:

```
Subject: Re: Your letter $DAdb12wD$
```

3.2 Token Types

XToken uses various types of tokens, each of which offers additional functionality. At the moment there exist four types but new types can be added easily.

SimpleToken: This token can be bound to a specific email address. If the token is received in an email message it is only valid if the sender address matches the address specified when creating the token. It is also possible to use the token without a bound address, in that case the token is always valid, independent of who uses it.

DateToken: As the name implies, this token has a date associated and is considered valid only until the specified date. If the token is used after that date it is considered invalid.

CounterToken: This type of token has an associated counter which is initialized to an arbitrary number when creating the token. Each time the token is received in an email message the counter is decremented until it reaches zero. Once the counter has reached zero the token is no longer considered to be valid.

DateCounterToken: The last token is a combination of a DateToken and a CounterToken and contains a date and a counter. The token can be used in two ways: Either it is valid until the specific date *or* until the counter reaches zero or it can be valid until the specific date *and* until the counter reaches zero.

It is possible to add almost arbitrary functionality by adding new token types. One idea could be to create a token which, if contained in an email, causes this email to be sent to a mobile phone by text message. XToken already supports for example sorting based on tokens, meaning that an email containing a specific token gets automatically moved to a pre-defined IMAP folder.

Tokens can be revoked at any time, rendering them invalid immediately. This is useful if a token has been compromised or should no longer be valid for some other reason.

The XToken interface provides a dialog to manage the token database. The dialog can be used to create, revoke and delete tokens and to display additional information about existing tokens.

3.3 Using Tokens to Process Incoming Emails

XToken can be configured to treat incoming email messages in a multitude of ways depending on whether a message contains a valid token (or a valid signature, see Section 5) or not. The possible actions include:

Move Valid Messages: In this mode XToken will move all messages containing a valid token to a defined folder, separating them from unsolicited messages.

Move Invalid Messages: When using this mode all invalid messages (i.e. without a valid token) will be moved to a defined folder leaving only legitimate messages in the inbox.

Flag Valid Messages: Instead of moving valid or invalid messages this mode leaves all messages in the inbox but flags valid messages with the standard IMAP flag `\Flagged`. Normally mail applications emphasize flagged messages somehow.

Concerning the technical implementation XToken behaves like a normal IMAP client which can be run in parallel with any other IMAP client. It monitors one or several IMAP mailboxes for incoming messages and handles them according to the configuration (see Figure 1). As a consequence XToken need not interact with the user's favorite email client: it cleans the mailbox directly on the IMAP server.

4 Token Distributor

When using tokens one of the difficulties is the distribution. Clearly it is infeasible to send a token to a person who might potentially send an email at some

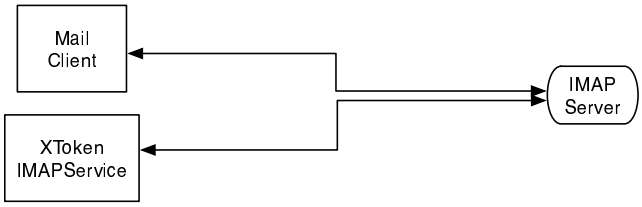

Fig. 1. XToken monitors one or several IMAP mailboxes in parallel with existing mail clients.

time in the future. It is therefore necessary to implement mechanisms which allow a person with a legitimate interest to obtain at least a temporary token. It is important to note that all mechanisms described below introduce an additional hurdle when making first contact with a recipient. Possible mechanisms to distribute tokens include:

- SMTP-based distributor: If someone wants to send a message he first sends a specially crafted email to the recipient. An application (presumably XToken) will intercept the message and automatically generate a response with a one-time token (a token which can be used exactly once). The sender can then include this token in the real message. This method is moderately resistant against spam bots as it requires a valid return email address. When using this method it probably makes sense to limit the number of tokens which are generated for a specific email address in a given period of time (e.g. 1 token per hour or per day etc.).
- Web-based distributor: Instead of sending an email message the sender has to fill in a web form (e.g. his name and email address) and an email containing a one-time token will then be sent to the specified address. This mechanism has the same resistance against spam bots as the distributor based on SMTP. A rate-limiting feature would be useful here as well.
- Web-based distributor with captcha: To eliminate the need of sending email messages one could imagine creating a script which directly generates and displays a one-time token but is hidden behind a captcha. Thus only a human can access the script and consequently the token. Unless some mechanism is implemented to prevent the same person from accessing the script again and again an attacker could generate an arbitrary number of tokens although at considerable cost as it would have to be done manually.
- Combination of SMTP-based distributor and captcha: A captcha is sent by email on request and the solution to the captcha is a one-time token. This is basically a normal SMTP-based distributor but with an added layer of security to prevent spam bots from using the distributor.

The disadvantage of the two web-based distributors is that a second mechanism has to be implemented which transfers a copy of the generated tokens to the computer of the recipient so that XToken will recognize the generated tokens

when someone uses them in an email. This transfer could be made by email but then the messages would have to be authenticated to prevent spammers from injecting invalid tokens.

5 Using Signed Emails

XToken also implements signatures as another mechanism to mark messages as valid (if used consequently signatures completely replace tokens). While signatures need more effort for the initial set-up, once they are configured they require less intervention than tokens. An important prerequisite though is that both sender and recipient use XToken. On the other hand, when using tokens it suffices if the recipient uses XToken.

The initial set-up when using signatures is more complicated for the following reason: In order for XToken to be able to intercept and sign outgoing messages they have to pass through XToken. This is achieved by using a local SMTP proxy (see also below). The additional effort required by the user is that he has to reconfigure his mail client so that outgoing emails pass through the local proxy. He then also has to configure XToken so it knows to which SMTP server it has to forward the signed emails. Although this reconfiguration is not particularly difficult it does require some basic networking knowledge.

The idea of using signatures in XToken has some similarities with domainkeys mentioned earlier. The main difference is that domainkeys work on whole domains while XToken signatures work on individual email addresses. Furthermore, while domainkeys have to be implemented by the system administrator of a specific domain, XToken signatures can be used by any end-user without being dependent on his ISP.

A signature is calculated over several header fields to make it unique for each message. It is made with the private-key of the sender and then added to the message together with the public-key which allows to verify the signature. For performance reasons the body of the message is not signed. While it is true that including the body when signing would increase the robustness of the signature, the current scheme only needs to retrieve the headers from the server (for the signature verification) which is a very efficient operation on an IMAP server. If the body also had to be retrieved this could significantly degrade the performance of the application, especially if a message contains attachments.

When a user receives a signed message XToken automatically retrieves the public-key from the message and compares it to a list of locally stored public-keys. If the key is not known the user is asked whether he wants to trust it and the key is then added to the local list. If the user decides to trust the key any subsequent message signed with this key will be considered valid.

If used consequently, signatures completely automate the process of sending and receiving legitimate messages. The user only has to intervene when a message with a new public-key is received.

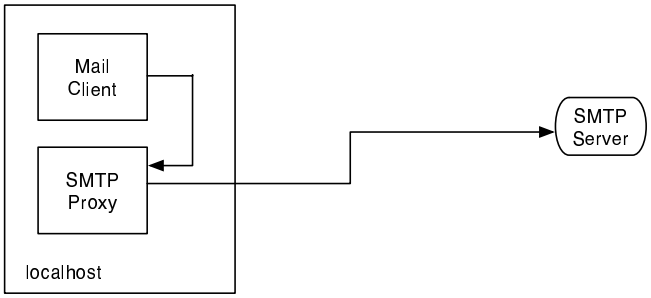

Fig. 2. Outgoing messages are signed using a SMTP proxy.

The signature algorithm used in XToken is ECDSA (elliptic curve digital signature algorithm) [1, 2] because of the compactness of the keys and the generated signatures, but any other standard signature scheme can be added.

Figure 2 illustrates how outgoing messages are signed using an SMTP proxy running on the local computer.

6 A Proposed Deployment Agenda

In this section we propose an agenda to change the way a user uses email messaging. Eventually, only solicited emails will arrive in his/her mailbox, but unexpected emails from human beings can still be received at the additional cost of human involvement.

1. The user distributes tokens and uses the flag action (legitimate emails get flagged). He/she observes how XToken behaves.
2. As the user begins to trust XToken, invalid messages are moved to quarantine in a specific folder. The user still regularly checks the folder but he/she is no longer distracted by incoming spam.
3. The quarantine folder is less and less regularly checked. Emails are automatically answered and removed after a while. The reply includes a free one-time token.
4. More and more users adopt XToken or equivalent applications. They are all aware of the new healthy way to use email. They are familiar with token distributors and understand they have to get a token first to send an email.
5. Users now automatically sign email headers and tokens are less and less used, except for online services.
6. Invalid messages are automatically answered and removed as soon as they arrive. The answer includes an URL explaining how to reach the user and a link to an online one-time token distributor.

At this stage, spammers can continue their activities in the two following ways:

- sniff (valid) email headers and use them with different bodies. This attack is quite limited by 1- the number of valid emails circulating, 2- the ability to sniff Internet traffic. These attacks can be completely thwarted by signing the full email and not just the headers.
- break one-time token distributors. This can easily be done by humans, but the human requirement is a bottleneck in spamming activities. Let us assume that the distributor is hidden behind a secure captcha. Clearly, the human time of spammers is not enough to get a sufficient number of tokens to make sure that sending spam is still profitable. An alternate way is to use distributed human computation [5]. Assuming, for instance, that spammers create a free pornography web site, they can easily make robots which collect captchas and have a front screen asking the visitor to solve one of the collected captchas for the robot before entering the web site. A way to thwart this attack consists of using captchas *and* tokens sent by email. If it became harder and harder for spammers to get sufficient valid email addresses, the front screen would require the email address of the visitor. Clearly, this would be dissuasive for the visitor. Should a visitor nevertheless give his email address then human victims of spam will at least have humans to blame for it so the issue will be brought back on a fairer ground.

7 Comparison XToken and SPA

SPA and XToken share the following features:

- Email addresses are augmented by adding additional information
- Modified addresses are given to friends and associates to enforce an acceptable use policy
- Addresses can be made to expire

Despite these similarities there are some significant differences:

- SPAs store all additional information in the email address itself. XToken only stores an identifier and keeps a local database with the complete information (in the case of tokens). Both methods have advantages and disadvantages. The method how the additional information is stored in an email address also differs.
- XToken is completely independent of the email infrastructure while SPAs as described in [9] rely on some specific infrastructure.

While the aim of SPAs and XToken is the same they differ in the approach. SPAs, at least partially, require a special infrastructure whereas an explicit goal of XToken was to make it infrastructure-independent so that it is entirely controlled by the end-user.

When used with signatures XToken also requires less intervention by the user than SPA. There is no need to create new modified addresses and send them to possible associates. The only effort required by the recipient is to occasionally accept or reject a public key.

8 Conclusion

We proposed a way to limit the spam phenomenon which needs neither any corporate involvement nor any changes to the email infrastructure: users can freely install this solution and use it. Our solution requires human email users to expend some additional effort for sending an email, at least the first time they use the email address. A key challenge is to make the token management interface as user-friendly as possible. Depending on whether users will accept this new way of using emails, our solution provides an easy way to limit spam at the user level. Hopefully, when most users adopt healthy ways to use emails, the activities of spammers will become less and less lucrative and can be eradicated throughout the Internet. We invite people to test the first implementation of this approach and welcome further development.

References

1. ANSI X9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standard Institute. American Bankers Association. 1998.
2. Digital Signature Standard (DSS). *Federal Information Processing Standards* publication #186-2. U.S. Department of Commerce, National Institute of Standards and Technology, 2000.
3. L. von Ahn, M. Blum, N.J. Hopper, J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology EUROCRYPT'03*, Warsaw, Poland, Lecture Notes in Computer Science 2656, pp. 294–311, Springer-Verlag, 2003.
4. C. Dwork, A. Goldberg, M. Naor. On Memory-Bound Functions for Fighting Spam. In *Advances in Cryptology CRYPTO'03*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 2729, pp. 426–444, Springer-Verlag, 2003.
5. C. Gentry, Z. Ramzan, S. Stubblebine. Secure Distributed Human Computation. In *Proceedings 6th ACM Conference on Electronic Commerce (EC-2005)*, Vancouver, Canada, pp. 155–164, ACM Press, 2005.
6. R. Roman, J. Zhou, J. Lopez. Protection against Spam using Pre-Challenges. In *Security and Privacy in the Age of Ubiquitous Computing IFIP TC11 20th International Information Security Conference (SEC'05)*, Chiba, Japan, pp. 281–293, Springer-Verlag, 2005.
7. M. Delany (Editor). Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys). IETF Draft, 2005
8. E. Harris. The Next Step in the Spam Control War: Greylisting. http://www.greylisting.org/articles/whitepaper.shtml, 2003
9. J. Ioannidis. Fighting Spam by Encapsulating Policy in Email Addresses, Symposium of Network and Distributed Systems Security (NDSS) 2003, San Diego, California, February 2003