Lightweight Signatures for Email

Ben Adida* David Chau* Susan Hohenberger*,† Ronald L. Rivest*

June 24, 2005

Abstract

We present the design and prototype implementation of a new public key infrastucture for email authentication. Our approach applies recent developments in identity-based cryptography and observations concerning the role of DNS and email servers in the current email architecture to produce end-to-end email signatures with no infrastructure change or new security assumption. Like current email signature proposals, this solution prevents email spoofing attacks such as phishing and, to some degree, spam. Unlike prior proposals, it conserves all current legitimate uses of email, transparently protects average Internet users, and optionally provides privacy-preserving repudiability. In other words, assuming today's email infrastructure, we provide the best possible email signature scheme with the smallest possible side-effect. We call this approach Lightweight Signatures.

1 Introduction

1.1 The State of Email

Email is not what it used to be. Spam – unsolicited bulk email that generally advertises commercial products – makes up more than 75% of all email volume [31], most of it spoofed. Email-based phishing attacks – spoofed emails that trick users into revealing private information, usually by linking to spoofed web sites – are estimated to have cost more than $10B in 2004. The number of these attacks is on the rise at a staggering 28% per month [6], and, most importantly, they are becoming quite sophisticated: spoofed emails now appear to come from close friends, relatives or colleagues, in reference to a topic relevant to the victim [25].

The consequences are devastating to the average email user. Banking institutions and major software companies have been reduced to recommending that their users not click on links in emails [4, 34]. Security specialists are instructing users not to trust the `From:` field [43]. The very openness that originally made email so easy to use is now threatening to make the medium completely unusable: email has become completely untrustworthy. **Our aim is to restore email trust using existing infrastructure.**

1.2 Making Spoofing as Difficult as Intercepting

The root of the problem is with SMTP, the Simple Mail Transfer Protocol [40], and its unbalanced threat model. While it is trivial to spoof Bob's outgoing email, the consistent interception of Bob's

*Computer Science and Artifical Intelligence Laboratory; Massachusetts Institute of Technology; 32 Vassar Street; Cambridge, MA 02139, USA. Email: {ben,ddcc,srhohen,rivest}@mit.edu.

†Supported by an NDSEG Fellowship.

incoming email is far more difficult. We aim to make spoofing as difficult as intercepting[1].

Our solution uses identity-based email signatures built on a new key distribution mechanism. This mechanism stems from two practical observations. SMTP already relies on (1) *the security of DNS* to correctly distribute MX records which dictate email routes, and (2) *the security of incoming mail servers* to store emails and make them available only to their intended recipient.

While DNS and incoming mail servers may (and probably should) be upgraded in the future to withstand more advanced attacks, their respective logical roles of email routing and storage will remain unchanged. **Our solution, lightweight signatures, thus relies on DNS to distribute domain-level cryptographic keys, and email delivery to distribute user-level cryptographic keys.**

1.3 A Foundation of Trust & Accountability using Existing Infrastructure

We *do not claim* that our proposal will provide bulletproof, general-purpose authentication, nor that it will suffice to single-handedly fend off all phishing attacks and spam. However, our system does provide the best possible foundation of trust given current email architecture.

For example, our system cannot detect, on its own, that an email from `cit1bank.com` (note the "1" instead of an "i") might fool a user into thinking the sender is from Citibank. However, our system *does* provide an accountability basis for verifying the true originating domain. Once this real sender is identified, a reputation management system or other mechanism can warn the user about such a suspiscious-looking domain name. Note that, without a system for trusting the `From:` address to begin with, such reputation management would be impossible. Our system provides a guarantee of sender address correctness, thus enabling a host of other methods and tools for determining which senders to trust.

Other proposed email authentication solutions, such as S/MIME [22] or PGP [47], could theoretically provide the very same foundation of trust. However, these solutions require a parallel public-key infrastructure that has proven too difficult to deploy in an Internet-wide setting, in large part because individual users are generally unable to manage cryptographic keys successfully. Even if one were to retrofit a classic signature scheme onto a domain-centric key generation and distribution model, it would not match the simplicity and efficiency of identity-based, single-keypair domain management. More importantly, the repudiability feature is available only under the assumption that a public key can be computed before its associated secret key.

1.4 Protocol Overview

We define our identity-based public key infrastructure and the associated signature and verification protocols. The components described here are diagrammed in Figure 1.

1.4.1 Identity-Based Cryptography

First conceptualized in 1984 by Shamir [44] and first implemented in 1988 by Guillou and Quisquater [18], *identity-based signatures* are typical signature schemes with the added property that a public key can be easily derived from any character string. In practice, the user's public key is derived from his email address.

A master authority publishes a master public key MPK, which is used in the derivation of a user's public key PK from his identity string id_string. The master secret key MSK is used to generate the user's corresponding secret key SK, which is then delivered to the user by the master

[1] One should note that SMTP was not originally designed to withstand these types of spoofing attacks.

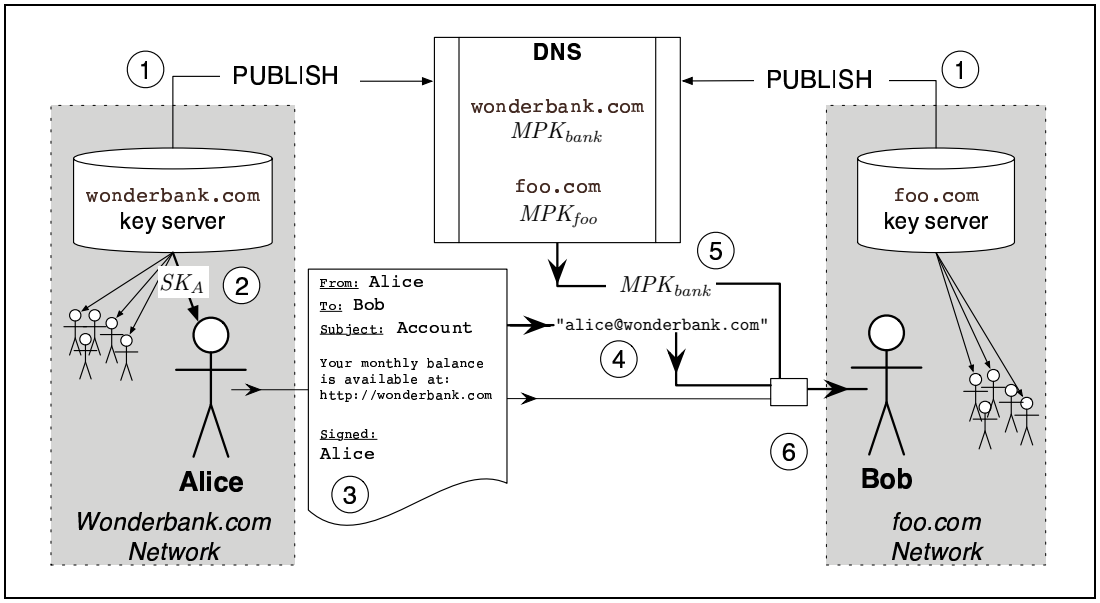

Figure 1: Lightweight PKI: (1) The domain keyserver for Alice publishes its MPK in the DNS (2) Alice's domain sends her her secret key SK_{Alice} (3) Alice signs her email and sends it to Bob (4) Bob extracts the `From:` field value, `alice@wonderbank.com`, from the email (5) Bob obtains $MPK_{wonderbank}$, the master public key for Alice's domain, from the DNS and computes PK_{Alice} against $MPK_{wonderbank}$ and `alice@wonderbank.com` (6) Bob verifies the signature against the message and PK_{Alice}.

authority. These same concepts are used in Identity-Based Encryption, first implemented by Boneh and Franklin in 2001 [8].

Identity-based cryptography offers two significant advantages: (1) key management becomes an institutional task, rather than an individual one, and (2) user public keys can be computed and used before the corresponding private key has been created. The practicality of these identity-based schemes has led to the launch of commercial enterprises, most notably Voltage Security [3].

1.4.2 An Identity-Based Public Key Infrastructure

The identity-based literature usually considers a single MPK from which all users derive their public keys. In practice, this is obviously inadequate: each institution will want to manage its users' keys. Thus, we consider that each email domain will become its own, independent, identity-based master authority. `wonderland.com` will issue keys for `alice@wonderland.com`, and `foo.com` will issue keys for `bob@foo.com`.

DNS Distribution of Master Public Keys. With each email domain functioning as an independent identity-based master authority, the master public key for each domain must be distributed in a manner that inspires confidence in the domain-MPK correspondence. For this purpose, we use the DNS system, much like DomainKeys [12] does for typical RSA public keys. We expect the lifetime of domain master keys to match that of generic DNS records, making the DNS architecture a good candidate for this type of distribution.

Email Distribution of User Secret Keys. Unlike typical keypairs, identity-based secret keys must be computed by a master authority, then transmitted to the appropriate users. Obviously, this delivery must be somewhat secure: a passive adversary that intercepts the secret key can

then impersonate the intended recipient. Lightweight Signatures exploit the existing trust between a user and his incoming mail server, a concept called Email-Based Identity and Authentication [17]. Secret keys are delivered to users by email, and users' incoming mail servers are expected to store them securely. For maximal security, the master authority should be the incoming mail server itself. However, a master authority on a separate machine, closely linked on the network or communicating via SMTP/TLS [21] with the incoming mail server, would work well, too.

1.4.3 The Repudiability Option

Privacy advocates have long noted that digital signatures present a double-edged sword [38]: signatures may turn a private conversation into a publicly-verifiable publication (e.g. an ex-girlfriend or boyfriend revealing past secrets). It is important to note that such **non-repudiation** is unnecessary for our purposes. Only the recipient needs to be convinced of the email's authenticity.

Thus, we design our system to support repudiable signatures that convince just the recipient, but no one else, of the message's authenticity. Specifically, we select an ad-hoc group signature scheme [41] that supports identity-based keys across different master authorities, and even different identity-based signature schemes altogether [5]. Using the well-known repudiation technique of designated-verifier signatures [29], the sender forms a signatory group including herself and the recipient. Upon receipt, the recipient is confident that the claimed sender authored the message, but he cannot convince a third party, as he himself could have forged this group signature.

This approach gives the sender complete control over the type of signature she chooses: a typical, non-repudiable signature, or a more privacy-friendly repudiable signature. A typical signature requires only the sender's private key, while a repudiable one also requires the recipient's public key. Our lightweight identity-based PKI thus proves particularly useful for creating repudiable email signatures.

1.5 Bootstrap and Adoption

Our proposed solution offers deployment options favorable both to commercial entities with large financial stakes in eradicating phishing attacks and individual early adopters.

Thanks to the identity-based construction, all cryptographic operations can be performed by the appropriate servers: a bank's outgoing mail server can sign all outgoing emails, while an ISP's incoming mail server, particularly those ISPs which provide web-based email, can discard all unsigned or improperly-signed emails, all without any end-user intervention. Note that false positives are not an issue in our system: ISPs can safely discard unsigned emails from domains that advertise a master public key in the DNS (with a policy flag set to discard unsigned emails).

That said, these signatures remain end-to-end: even when a user sends mail via a third-party outgoing mail server, on a foreign network, he can sign his emails using an upgraded mail client. In addition, we propose a mechanism for individuals to use alternate master keyservers when their own domain does not yet support lightweight signatures.

1.6 Prototype & Performance

We built a prototype of this solution using Guillou-Quisquater identity-based signatures [18] and the latest, separable ad-hoc group signature construction [5], as inspired by Cramer, Damgard, and Schoenmakers [10]. The software is currently available for review upon request, and will be released as free software with the final version of this paper.

The Guillou-Quisquater signature scheme requires one hash and two integer group modular exponentiations for a single signature, and one hash and a single integer group modular expo-

nentiation for verification. The two-person ad-hoc group signature needed for repudiation simply doubles those measurements for signing and verification. In either case, performance on a modern workstation is very good, requiring less than 10ms for each operation.

1.7 Previous and Related Work

The email authentication problem has motivated a large number of proposed solutions. We review the relevant ones, particularly the latest solutions motivated specifically by phishing and spam, and note the advantages of our proposal.

General Purpose Authentication. End-to-end digital signatures for email have been proposed [7, 46] as a mechanism for making email more trustworthy and thereby preventing spoofing attacks such as phishing. One other proposal suggests labeling email content and digitally signing the label [19]. All of these proposals require some classic form of Public-Key Infrastructure, e.g. X.509 [15]. As detailed in section 1.3, the classic PKI requirement is generally a non-starter for achieving widespread deployment.

Fighting Phishing Emails. A number of spam-related solutions have been applied to phishing. Blacklists of phishing mail servers are sometimes used [45, 28], though the recent attacks that hijack virus-infected PCs from around the world make this technique fairly useless in the phishing wars. Content filtering, where statistical machine learning methods are used to detect likely attacks [42, 30, 32] has been unable to withstand the latest, more advanced phishing attacks that look *exactly* like legitimate emails. Meanwhile, collaborative methods [14] that enable users to help one another have been countered by the rise of highly customized and personalized phishing [25].

Alternatively, path-based verification has been proposed in a plethora of initiatives. Those which rely on DNS-based verification of host IP addresses were recently reviewed by the IETF MARID working group [23, 27, 26]. Another major proposal, Yahoo's DomainKeys [12], suggests a cryptographic proof of path, where the outgoing mail server, certified via DNS, signs all outgoing emails. Conceptually, these proposal are all similar, though their handling of special cases – mailing lists, mail forwarding, bouncing – varies. Though these approaches may prove useful in preventing a number of phishing attacks, they do not provide the end-to-end signatures needed to preserve current legitimate uses of email, including multiple email personalities, mobile users, and complete email forwarding and mailing list support.

Fighting Phishing Web Sites. On the web site front, a number of solutions attempt to inform users of the reputation and origin of the web sites they visit. These generally come in the form of web browser toolbars [37, 20], or sometimes email-to-web gateway warnings [16]. All of these schemes attempt to counter the social engineering aspect of phishing attacks. Because web sites have a much stronger accountability architecture than emails, these techniques can successfully flag worrisome public web sites by reputation management.

These approaches address only the web component of the attack when the spoofed web sites are well-known commercial entities. Recent "colleague-to-colleague" phishing attacks that target smaller sites or internal corporate credentials cannot be prevented by these approaches. The root cause remains the lack of email trust.

1.8 This Paper

In section 2, we review the necessary cryptographic and systems building blocks. In section 3, we detail the protocol based on these building blocks, specifically the identity-based key distribution

infrastructure and the repudiability option. We then explore the issue of technology adoption with bootstrap mechanisms and extensions to the core scheme in section 4. In section 5, we note how our scheme establishes a platform of trust and accountability for email addresses and domains. We wrap up with a description of our prototype and its performance in section 6, and conclude in section 7.

2 Building Blocks

We now review and extend certain basic building blocks which will form key components of our Lighweight Signature solution.

2.1 Identity-Based Domains

An identity-based signature scheme defines a master authority in possession of a master secret key MSK with an associated, publicly-distributed master public key MPK. Users are identified by *id_string*, which is typically the user's email address. A user's public key PK can be publicly computed from MPK and *id_string*, while a user's secret key SK is computed by the master authority using MSK and *id_string*, then delivered to the user.

In our scheme, we propose that every email domain become a master authority for an identity-based signature (IBS) scheme *of their choosing*. One of the strengths of our solution is that no coordination between these domains is needed. If Alice has an email account `alice@wonderland.com`, then `wonderland.com` should generate a master keypair ($MPK_{\texttt{wonderland.com}}, MSK_{\texttt{wonderland.com}}$) according to the IBS scheme it has chosen, and Alice's public key $PK_{\texttt{alice@wonderland.com}}$ will be derived from $MPK_{\texttt{wonderland.com}}$ and the string `'alice@wonderland.com'` (plus some additional parameters defined in section 3.2).

2.2 DNS Master Public-Key Distribution

DNS provides a semi-trusted naming and routing infrastructure for domains. Hostname to IP address mappings obviously rely on it. Mail routing also relies on it, using MX records. Ingeniously, DNS has been proposed as a pseudo public-key infrastructure with Yahoo's DomainKeys [12].

In the DomainKeys scheme, RSA public keys are stored in specially named DNS records. Specifically, DomainKeys reserves the `_domainkeys` subdomain for every domain with an MX record. Any entries within this subdomain are public keys in base64-encoding with some associated parameters. By keeping each public-key record short (i.e. less than the size of a single UDP packet), this DNS-based key distribution mechanism functions over a large portion of existing DNS servers.

The cryptographic algorithms we use are quite different from those used in DomainKeys, but our approach uses the same technique for distributing domain-based keys, cryptographic parameters, and domain email policy. Specifically, Alice's domain `wonderland.com` will publish a base-64 encoded $MPK_{\texttt{wonderland.com}}$ in a DNS TXT record associated with its MX record. The lifetime of such a key should be equivalent to that of a hostname, which makes DNS's time-to-live update mechanism quite adequate for this task. No user-level keys are distributed via DNS.

One should note that MPK might eventually be certified by a stronger mechanism than plain DNS records. DNSSEC [13], for example, could be used to distribute the very same MPK with no change to our scheme.

2.3 Email Secret-Key Distribution

The ability to receive email at a certain address is an oft-used authentication mechanism in existing applications. Web password reminders, mailing list subscription confirmations, and web e-commerce notifications all use email as a semi-trusted messaging mechanism. This approach, called Email-Based Identity and Authentication [17], is particularly well-adapted to establishing email address trust, because safeguarding a user's online identity already requires mechanisms for keeping adversaries out of the user's inbox.

Delivering personal, semi-sensitive data to a user can thus be performed by simply sending a user email. The user will then gain access to this data by authenticating to their incoming mail server in the usual way, via account login to an access-controlled filesystem, webmail, POP3 [36], or IMAP4 [11].

In particular, we propose to distribute the identity-based user-level secret keys, e.g. Alice's $SK_{\texttt{alice@wonderland.com}}$, via email. Certainly, one concern with this approach is that an adversary might tap the network and intercept the key. To thwart this threat, we suggest eventually packaging master domain functionality within the incoming mail server for secure, internal delivery. In the interim, a secure configuration can be achieved by using SMTP/TLS [21] between the master keyserver and the mail server.

2.4 Repudiability from Ad-Hoc Group Signatures

The downside of digital signatures is that they often deliver more than the strictly necessary functionality. In our case, specifically, universally-verifiable signatures can pose a privacy problem: a secret conversation with a once-trusted friend may turn into a public posting where anyone can verify the messages' authenticity, once this friend is no longer trusted. This notion that repudiability is important for privacy was most recently explored by Borisov et al. [9].

As has been suggested in the literature [41], we can use group signatures to enable repudiability without forgeability. Specifically, we select an ad-hoc group signature scheme which enables the formation of groups across different master public keys and even different identity-based signature schemes [5]. We then craft the group signature to include exactly the sender and recipient in its signatory group will ensure the two important properties we seek:

1. the recipient is confident the email was authored by the sender, since the recipient knows he himself did not sign it.
2. the sender can later repudiate his message, claiming the recipient is trying to frame him.

With this construction, we obtain a fairly strong form of repudiability: anyone can fake a message sent to themselves. However, we simultaneously do away with forgeability, since only the sender or recipient can properly sign an email. It should also be noted that the group can include more than just the sender and recipient. This additional group member should be construed as an additional avenue for repudiation. Such an approach will be particularly useful in extensions to our core protocol (Section 4).

3 The Protocol

With our building blocks in place, we now present the complete design of Lightweight Signatures.

3.1 Email Domain Setup

In the basic setup, Alice with email address `alice@wonderland.com` will obtain her secret key from a keyserver dedicated to her domain `wonderland.com`. The setup procedure for that master authority is defined as follows:

1. choose any identity-based signature scheme from the current literature.
2. generate a master keypair $(MPK_{\texttt{wonderland.com}}, MSK_{\texttt{wonderland.com}})$ for this scheme.
3. define key issuance policy *Policy*, including whether emails originating from this domain can, should, or must be signed. (Details of this policy are defined in section 4.2.)
4. publish $MPK_{\texttt{wonderland.com}}$ and *Policy* as a DNS TXT record associated with the MX record for `wonderland.com`. The DNS record content is formatted as `mpk=`$\langle MPK \rangle$`,policy=`$\langle Policy \rangle$ with *Policy* a short string and MPK in base64 encoding.

3.2 User Identities

Per the identity-based construction, a user's public key PK can be derived from any character string *id_string* that represents the user's identity. We propose a standard format for *id_string* in order to specifically support email address authentication with various master domains, domain policies, key types and key expiration mechanisms.

Master Domain. In most cases, `bob@foo.com` obtains a secret key derived from a master keypair whose public component is found in the DNS record for the expected domain, `foo.com`. However, in cases related to bootstrapping (see Section 4), Bob might obtain a secret key from a domain *other than* `foo.com`.

For this purpose, we build a *key_domain* parameter into the user identity character string. Note that `foo.com` should always refuse to issue secret keys for identity strings whose *key_domain* is not `foo.com`. However, `foo.com` may choose to issue a key for `alice@wonderland.com`, as long as the *key_domain* within the identity string is `foo.com`. The complete example described at the end of this section clarifies this particular issue.

Key Types. An identity-based domain may be used for any number of applications. In order to ensure that a single key is used only for its intended purpose, we propose including type information in *id_string*. Thus, our identity string format may be reused by other cryptographic schemes without generating a separate master keypair. Consider *type*, a character string composed only of lowercase ASCII characters. This type becomes part of the overall identity string. For the purposes of our application, we define a single type: `lightsig`.

Key Expiration. In order to provide simple key revocation capabilities, the user identity string includes key expiration information. Specifically, *id_string* includes the last date on which the key is valid: *expiration_date*, a character string formatted according to ISO-8601 [24].

Constructing Identity Character Strings. An *id_string* is thus constructed as:

$$\langle key\_domain \rangle, \langle email \rangle, \langle expiration\_date \rangle, \langle type \rangle$$

For example, a 2005 Lightweight Architecture identity string for Bob would be:

`foo.com,bob@foo.com,2005-12-31,lightsig`

If Bob obtains his secret key from a master authority different than his domain, e.g. `lightsig.org`, his public key would necessarily be derived from a different *id_string*:

`lightsig.org,bob@foo.com,2005-12-31,lightsig`

Notice that `lightsig.org` will happily issue a secret key for that identity string, even though the email address is not within the `lightsig.org` domain. This is legitimate, as long as the *key_domain* portion of the *id_string* matches the issuing keyserver.

3.3 Delivering User Secret Keys

Each domain keyserver will choose its own interval for regular user secret key issuance, possibly daily, weekly or monthly. These secret keys are delivered by email, with a well-defined format – e.g. XML with base64-encoded key, including a special mail header – that the mail client will recognize. The key-delivery email is kept in the user's inbox for all mail clients to access, in case the user checks his email from different computers. The mail client may check the correctness of the secret key it receives against its domain's master public key, either using a specific algorithm inherent to most IBS schemes, or by attempting to sign a few messages with the new key and then verifying those results.

3.4 The Repudiability Option

As mentioned earlier, privacy concerns may lead users to seek repudiability via the ad-hoc group signature construction [5]. In fact, we anticipate many person-to-person emails will use this approach, as it preserves the status-quo of casual conversation over email. We note that a group signature between Alice, the sender, and Bob, the recipient, functions as follows:

- Alice signs the email using her secret key SK_{Alice} and Bob's public key PK_{Bob}.
- Anyone can use Bob or Alice's public key to verify that one of the two signed the email.

Thus, it is clear that the repudiable-signature protocol is a strict superset of the simple-signature protocol. We will detail the repudiable one and note the steps that can be skipped in the case of simple signatures.

Back-Dated Secret Keys. When Alice wishes to compute Bob's public key for repudiation purposes, she may not know which expiration date to use for Bob. Alice can simply select the current date. We then require that domains be willing to distribute back-dated secret keys (to match the incoming public key) on request to any of their members. Few users will take this opportunity, but the fact that they *could* yields more complete repudiability. Such requests for back-dated keys can simply be handled by signed email to the keyserver.

3.5 Signing & Verifying Messages

Consider Alice, `alice@wonderland.com`, and Bob, `bob@foo.com`. On date `2005-05-09`, Alice wants to send an email to Bob with subject $\langle subject \rangle$ and body $\langle body \rangle$. When Alice clicks "send," her email client performs the following actions:

1. prepare a message $\mathcal{M}$ to sign:

```
From:  alice@wonderland.com
To:  bob@foo.com
Subject:  ⟨subject⟩
Timestamp:  ⟨timestamp⟩
Body:  ⟨body⟩
```

2. if Alice desires repudiability, she needs to obtain Bob's public key:
   (a) obtain $MPK_{\texttt{foo.com}}$, the master public key for Bob's domain `foo.com`, using DNS lookup.
   (b) assemble id_string_{Bob}, an identity string for Bob using `2005-05-09` as the *expiration_date*: `foo.com,bob@foo.com,2005-05-09,lightsig`
   (c) compute PK_{Bob} from $MPK_{\texttt{foo.com}}$ and id_string_{Bob}.
3. sign the message $\mathcal{M}$ using SK_{Alice}, $MPK_{\texttt{wonderland.com}}$. Optionally, for repudiability, use PK_{Bob} and $MPK_{\texttt{foo.com}}$. The computed signature is σ.
4. add headers to the mail message:
   - σ in base64-encoding in SMTP header `X-LSSignature`,
   - the exact id_string_{Alice} in SMTP header `X-LSSenderIDString`,
   - and the timestamp used in the signature in SMTP header `X-LSTimestamp`
   - (Optional) the exact id_string_{Bob} in SMTP header `X-LSRecipientIDString`,

Upon receipt, Bob needs to verify the signature:

1. determine the sender's email address, `alice@wonderland.com`, and domain name, `wonderland.com`, according to the email's `From` field.
2. obtain $MPK_{\texttt{wonderland.com}}$, using DNS lookup on `_lta.wonderland.com`.
3. ensure that PK_{Alice} is correctly computed from the claimed `X-LSSenderIDString` and public $MPK_{\texttt{wonderland.com}}$, and that this *id_string* is properly formed (includes Alice's email address exactly as indicated in the `From` field, a valid expiration date, a valid type).
4. recreate the message $\mathcal{M}$ that was signed, using the declared `From`, `To`, and `Subject` fields, the email body, and the timestamp declared in `X-LSTimestamp`.
5. Verify $\mathcal{M}$, σ, PK_{Alice}, $MPK_{\texttt{wonderland.com}}$ to check that Alice is in the signatory group.
6. If Alice applied a repudiable signature, Bob **must** check that this second signature verifies against his own public key. If this second signature is invalid, the first signature (Alice's) may be invalid, too:
   (a) ensure that PK_{Bob} is correctly computed from the claimed `X-LSRecipientIDString` and the DNS-advertised $MPK_{\texttt{foo.com}}$, and that this *id_string* is properly formed (includes Bob's email address, a valid expiration date, a valid type).
   (b) verify $\mathcal{M}$, σ, PK_{Bob}, $MPK_{\texttt{foo.com}}$ to check that Bob (himself) is in the signatory group.

If all verifications succeed, Bob can be certain that someone in possession of Alice's secret key SK_{Alice} produced σ. If Alice's master domain keyserver is behaving correctly, that signer can only be Alice.

4 Technology Adoption

The most challenging aspect of cryptographic solutions is their path to adoption and deployment. In this section, we describe a number of techniques for bootstrapping the adoption of Lightweight Signatures. We also show how our solution lends itself quite well to commercial adoption and early-adopter usage.

4.1 How to Get Started: Bootstrapping Lightweight Trust

Alice wishes to send an email to Bob. If both have upgraded email clients, and both domains are Lightweight-Signature-enabled, they can proceed as described in section 3. What happens, however, if one of these elements is not yet in place?

Getting an Alternate Secret Key. The preeminent bootstrap situation occurs when Alice wishes to adopt Lightweight Signatures before her email domain `wonderland.com` supports it. In this situation, Alice may choose an alternate master authority domain created specifically for this purpose, e.g. `lightsig.org`. `lightsig.org` must explicitly support the issuance of external keys, i.e. keys corresponding to email addresses at a different domain than that of the issuing keyserver. To obtain such a key, Alice will have to explicitly sign up with `lightsig.org`, most likely via a web interface. Her *id_string* will read:

```
lightsig.org,alice@wonderland.com,2005-12-31,lightsig
```

Note that we do not expect `lightsig.org` to perform any identity verification beyond email-based authentication: the requested secret key is simply emailed to the appropriate address.

One expects that a non-commercial organization would run `lightsig.org`, much like MIT runs the MIT PGP Keyserver [35], where anyone can freely use the service. Another possibility is that existing identity services – e.g. Microsoft Passport [33] or Ping Identity [39] – will issue Lighweight keys for a small fee. Where PGP requires large, mostly centralized services like the MIT PGP Keyserver, Lightweight Signatures users may choose from any number of keyservers.

Of course, when Bob receives a lightweight-signed email, he must consider his level of trust in the issuing keyserver, not in the `From:` email address domain. Most importantly, domain policies should be established to enable a domain to prevent the issuance of keys for its users by alternate domains. We return to this point shortly.

Bootstrapping Repudiability. When Alice wishes to send an email to Bob, she may notice that Bob's domain `foo.com` does not support Lightweight Signatures. In order to obtain repudiability, however, she needs to compute a public key for which Bob has at least the capability of obtaining the secret key counterpart. For this purpose, Alice can use the same `lightsig.org` service with an advertised DNS policy of issuing keys for external users. Thus, Alice may use:

```
lightsig.org,bob@foo.com,2005-02-05,lightsig
```

as *id_string* from which to derive a PK for Bob.

Note that Bob need not ever actually retrieve a secret key from the `lightsig.org` service. The mere fact that *he could potentially retrieve a secret key at any time* is enough to guarantee repudiability for Alice. Note also that, somewhat unintuitively, it makes sense to have Alice select the Lightweight Signature service to generate Bob's public key: in our setting, Bob's public key serves Alice, not Bob, as it is an avenue for sender repudiability.

4.2 How Domains Announce Their Participation: Domain Policies

Once an email domain decides to deploy Lightweight Signatures, it simply needs to create a keyserver and specify the appropriate DNS record. We propose that this record include a *Policy* parameter to specify how the domain chooses to participate in the Lightweight Signature architecture. *Policy* contains two major components: *ExternalPolicy* and *InternalPolicy*.

ExternalPolicy determines whether the domain in question is willing to issue keys for users of other domains, as the previous section suggested. A domain like `bigbank.com` will likely set that parameter to False, while a domain like `lightsig.org` will likely set that parameter to True.

InternalPolicy determines the domain's requirements on its email users as well as its guarantees to any recipients. Three possible values should be considered:

- None: users of this domain may sign their emails, either with keys issued by this domain's keyserver or by another keyserver, or not sign them at all.
- Basic: users of this domain can only sign emails with keys issued by this keyserver, not by any other keyserver, or not sign them at all.
- Strong: users of this domain are required to sign all their emails with a key issued by this domain.

A company like `bigbank.com` would likely set their *InternalPolicy* to Strong, while an ISP like `wonderland.com` might set their's to Basic.

Using Domain Policies to Automatically Detect Spoofing. When Bob receives an email from Alice, he may be faced with two possibly confounding situations: the message bears no signature, or the message bears a signature signed with a PK for Alice derived from a different master authority than that of Alice's email address.

Bob must check the *Policy* for `wonderland.com`.

- If the message bears no signature:
  - If there is no policy or if *InternalPolicy* is None or Basic, then Bob finds himself in a grey zone where he has no positive or negative information on the email authenticity (same situation as we face without lightweight signatures).
  - If the policy is Strong, Bob can confidently discard the email.
- If the message bears a signature with a master authority that doesn't match Alice's email address:
  - If there is no policy or if *InternalPolicy* is None, then Bob can trust the email as much as he is willing to trust this alternate master authority.
  - If *InternalPolicy* is either Basic or Strong, then Bob can confidently discard the email.

4.3 Lightweight Signatures at the Server Level

Though Lightweight Signatures respect the end-to-end nature of email and do not require a mail server upgrade, numerous optimizations can be performed with an upgraded incoming or outgoing mail server. Specifically, we consider a server-optimized deployment of Lightweight Signatures where Bob's incoming mail server performs verifications, and Bob's outgoing mail server has access to SK_{Bob} (via $MSK_{\texttt{foo.com}}$) and signs all emails accordingly.

DNS MPK Lookups. An incoming mail server can easily look up the MPK records of senders when emails are received, and include this data in additional SMTP headers before delivering the emails to the user's mailbox. This is particularly useful for mail clients that may be offline when they finally process the downloaded emails.

Signature Verification. The incoming mail server can go even further and perform the signature verification, indicating the result by yet another additional SMTP header. The client would be saved the effort of performing the cryptographic operations. This is particularly useful for inexperienced users and low-computation-power clients, like cell phones or PDAs. In cases where the incoming email clashes with the sending domain's *Policy*, as illustrated in section 4.2, the incoming mail server can confidently discard the fraudulent email without any user intervention.

Signature Creation. If a user sends email through his regular outgoing mail server, the signature can be applied by that server. This optimization is also particularly useful for inexperienced users and low-computation-power clients. Note that, unlike the signature applied by DomainKeys, this server-applied signature certifies the `From:` address, not the email's path. Thus, this solution maintains end-to-end support for mailing lists and arbitrary email forwarding.

Transparent Lightweight Signatures. Internet Service Providers can combine the previous optimizations to provide all functionality at the server. In other words, without ever upgrading their mail client, home users can get all the benefits of Lightweight Signatures without upgrading software or taking any action. ISPs can deploy Lightweight Signatures transparently.

This approach is even more appealing – and necessary – for web-based email, particularly web-only email providers like Hotmail, Yahoo, and Gmail. A web client does not provide the necessary framework to perform public-key cryptography. In these cases, the web server is the only system component which can provide signing and verification functionality.

4.4 Mailing Lists, Multiple Recipients, and Forwarding

Note again that the signatures we propose here are truly end-to-end: they certify the `From:` field of an email, independently of the email transmission path, client software, and recipient. Thus, mailing lists, multiple recipients, and, more generally, arbitrary email forwarding, are completely supported by our "out-of-the-box" scheme as described in section 3.

With repudiable signatures, the recipient's email address becomes important. Arbitrary single-user email forwarding presents no additional challenge. Sending email to multiple recipients requires that repudiable signature be performed pairwise. Mailing list repudiation is the most complicated case, which we address in the next section.

4.5 Advanced Repudiation via Evaporating Keys

Lightweight Signatures offer repudiability because someone in possession of a secret key *other than the message author's* might have signed the message. When the recipient's email address doesn't present a proper avenue for repudiation, an alternate approach to achieving the same end-goal is to set up an **evaporating key**. A similar idea was recently discussed by Borisov et al. [9].

In our scheme, an evaporating key has an expiration date, like other keys defined for Lightweight Trust. Once that expiration date has passed, the key's master domain publishes the corresponding secret key, most likely on a web site. This publication is the "evaporation" phase, after which the key is effectively useless. Most importantly, all signatures whose group includes the evaporating key become completely repudiable at evaporation time, since the secret key is available to anyone.

In order to distinguish these keys from user-specific keys (which we do *not* want to evaporate), evaporating keys bear a different *type* in their user identity character string: `ltaevap`. Thus, a master domain can choose whether to issue evaporating keys at all. If it chooses to issue evaporating keys, it can choose which email address character strings are acceptable for evaporation purposes. Since evaporating keys provide repudiability, and since repudiability benefits the sender, Alice should obtain her evaporating keys from a domain she trusts for this purpose, possibly her own domain `wonderland.com`.

Repudiability for Mailing Lists. Evaporating keys can provide repudiation for messages sent to mailing lists. When Alice sends an email to a mailing list, she may not know the eventual recipients of the email. If she signs with her secret key and an evaporating public key, recipients can trust Alice's authorship as much as they trust the evaporating domain, and Alice gains repudiability as soon as the evaporation interval comes to an end. Because email clients are not always aware of the fact that the recipient is a mailing list, one practical option is to create a 3-way repudiable signature using Alice's secret key, the recipient's public key, and an evaporating public key.

Weaknesses of Evaporating Keys. Given the utility of evaporating keys, one might be tempted to forgo repudiability against a real recipient altogether in favor of repudiability against an evaporating key. However, evaporating keys do not provide perfect repudiability, as they are vulnerable to timestamping attacks during the pre-evaporation period. Thus, evaporating keys should be considered a *backup avenue for repudiation.*

4.6 Practical Deployment Scenarios

Using the extensions described above, Lightweight Signatures can handle two very different, but equally important use cases.

Scenario One: Big Companies and Average Internet Users. A primary goal of any email authentication scheme is to stop phishing emails from getting into customer inboxes. In our solution, eBay could set its *InternalPolicy* to Strong. Then, Hotmail, Yahoo, and Gmail could immediately discard all improperly signed "`customerservice@ebay.com`" emails at the server level. This protection would be transparent for the average user.

Scenario Two: Advanced User on a Business Trip. Any solution, however, that addresses the above scenario should not infringe on the flexiblity of Alice, an advanced user, to send emails from various outgoing mail servers while on her business trip; nor should it infringe on her *current right* not to have to publicly sign all her emails. Using our end-to-end signatures with the repudiability option, Alice maintains all the flexibility and privacy that she enjoys today.

5 A Platform of Trust & Accountability

Like straight digital signatures, Lightweight Trust prevents email sender spoofing. This spoofing protection is the key enabling property of our system: recipients of email can confidently identify the senders of the emails they receive.

This is, without a doubt, insufficient to single-handedly block spoofing attacks based on social engineering, where the sender's email domain might be similar, but not identical, to a well-known domain – e.g. `cit1bank.com` instead of `citibank.com`. However, much like SSL provides a basic building block of trust for the web, lightweight signatures provide an accountability trace for all

emails. The user within the sending domain, and the sending domain itself, can be held accountable for the content of the emails that are properly signed according to the domain's MPK.

With accountability comes the potential deployment of numerous existing techniques for trust. Reputation management of email domains, user interface hints, and numerous other tricks can be built on the platform of accountability that Lightweight Signatures provide.

6 Prototype

We built a prototype implementation of lightweight signatures with the Guillou-Quisquater [18] identity-based signature scheme using the GNU Multi-Precision math library in C++. This prototype provides command-line executables for generating a master keypair, extracting a secret key, computing a public key, packaging and unpackaging a secret key in an email, signing a message given a secret key, and verifying an incoming message. MPK's were successfully stored and retrieved via a standard DNS server [2]. A simple keyserver was built using Python with a web interface for key requests. Finally, the command-line executables were integrated into the Emacs RMAIL client [1].

This prototype showed that an average user can begin using Lightweight Signatures using a simple software upgrade and a one-step web signup process. A domain that wishes to support Lightweight Signatures can do so with minimal effort: a key generation, a DNS record update, and a simple web application for distributing keys. Signature and verification computations take less than 10ms each on a modern 1.5GHz AMD Athlon computer with 1GB of RAM.

The source code for this prototype is available upon request, and we plan to release it as free software to accompany the final version of this paper.

7 Conclusion

We have presented the design and prototype implementation of an identity-based public key infrastructure specifically made for email authentication. Our approach makes spoofing Bob's outgoing email as difficult as reliably intercepting Bob's incoming email. Though not bulletproof, we believe this scheme can significantly reduce phishing and potentially mitigate spam.

Our approach is practical. All existing email functionality is preserved, and no significant new infrastructure is required. Bootstrapping is natural, and the solution's potential for adoption is promising, given the flexible alternatives for early-adopter and institutional deployment. Our solution is also the only anti-spoofing proposal that addresses the privacy issues raised by the widespread deployment of digital signatures.

Future Problems. We note that our solution, like others before it, includes the DNS system as a secure component of a cryptographic architecture. With such an approach, we suspect the DNS system will come under more scrutiny and attack. There is, without a doubt, a need to further secure the DNS system and generally establish a highly secure mechanism for distributing certified domain-specific information. Our solution will inherit any success in this area.

We also note that our solution specifically does not cover encryption. One will surely want to explore how this system might be adapted for this purpose, as encryption is surely the next question on every reader's mind.

References

[1] The default emacs mail reader. `http://www.gnu.org/software/emacs/manual/html_node/Rmail.html`.

[2] MaraDNS, a DNS Server. `http://maradns.org`.

[3] Voltage Security. `http://voltage.com`.

[4] American Banking Association. Beware of Internet Scrooges this Holiday. `http://biz.yahoo.com/prnews/041209/dcth013_1.html`.

[5] Anonymous. How to Sign for Any Group. In submission, contact authors for a copy of the paper.

[6] Anti-Phishing Working Group. `http://www.antiphishing.org/`.

[7] Anti-Phishing Working Group. Digital Signatures to Fight Phishing Attacks. `http://www.antiphishing.org/smim-dig-sig.htm`.

[8] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer Verlag, 2001.

[9] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES '04: the 2004 ACM workshop on Privacy in the electronic society*, pages 77–84. ACM Press, 2004.

[10] Ronald Cramer, Ivan Damgard, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology – CRYPTO '94*, volume 839 of LNCS, pages 174–187, 1994.

[11] M. Crispin. RFC 1730: Internet Mail Access Protocol - Version 4, December 1994.

[12] Mark Delany. Domain-based Email Authentication Using Public-Keys Advertised in the DNS (DomainKeys), August 2004. `http://www.ietf.org/internet-drafts/draft-delany-domainkeys-base-01.txt%`.

[13] D. Eastlake. RFC 2535: Domain Name System Security Extensions, March 1999.

[14] E. Damiani et. al. Spam Attacks: P2P to the Rescue. In *WWW '04: Thirteenth International World Wide Web Conference*, pages 358–359, 2004.

[15] M. Cooper et. al. Internet X.509 Public Key Infrastructure (latest draft). *IETF Internet Drafts*, January 2005.

[16] Eudora. ScamWatch. `http://www.eudora.com/email/features/scamwatch.html`.

[17] Simson L. Garfinkel. Email-Based Identification and Authentication: An Alternative to PKI? *IEEE Security & Privacy*, 1(6):20–26, November 2003.

[18] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *LNCS*, pages 216–231. Springer Verlag, 1988.

[19] Amir Herzberg. Controlling spam by secure internet content selection. Cryptology ePrint Archive, Report 2004/154, 2004. `http://eprint.iacr.org/2004/154`.

[20] Amir Herzberg and Ahmad Gbara. Trustbar: Protecting (even naive) web users from spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004. `http://eprint.iacr.org/2004/155`.

[21] P. Hoffman. SMTP Service Extension for Secure SMTP over Transport Layer Security. Internet Mail Consortium RFC. `http://www.faqs.org/rfcs/rfc3207.html`.

[22] IETF. S/MIME Working Group. `http://www.imc.org/ietf-smime/index.html`.

[23] IETF. MTA Authorization Records in DNS (MARID), June 2004. `http://www.ietf.org/html.charters/OLD/marid-charter.html`.

[24] International Standards Organization. ISO 8601: Numeric representation of Dates and Time, January 2003.

[25] Markus Jakobsson. Modeling and Preventing Phishing Attacks. In Andrew Patrick Moti Yung, editor, *Financial Cryptography 2005*, LNCS. Springer Verlag, 2005.

[26] John Levine, A. DeKok, and et al. Lightweight MTA Authentication Protocol (LMAP) Discussion and Comparison, February 2004. `http://www.taugh.com/draft-irtf-asrg-lmap-discussion-01.txt`.

[27] John R. Levine. A Flexible Method to Validate SMTP Senders in DNS, April 2004. `http://www1.ietf.org/proceedings_new/04nov/IDs/draft-levine-fsv-01.txt`.

[28] MAPS. RBL - Realtime Blackhole List, 1996. `http://www.mail-abuse.com/services/mds_rbl.html`.

[29] Markus Jakobsson and Kazue Sako and Russell Impagliazzo. Designated Verifier Proofs and their Applications. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *LNCS*. Springer Verlag, 1996.

[30] Justin Mason. Filtering Spam with SpamAssassin. In *HEANet Annual Conference*, 2002.

[31] MessageLabs. Annual Email Security Report, December 2004. `http://www.messagelabs.com/intelligence/2004report`.

[32] T.A. Meyer and B. Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Conference on Email and Anti-Spam 2004*, July 2004.

[33] Microsoft. Passport Identity Service. `http://www.passport.net`.

[34] Microsoft. Phishing Scams: 5 Ways to Help Protect Your Identity. `http://www.microsoft.com/athome/security/email/phishing.mspx`.

[35] MIT. PGP Public Key Server. http://pgp.mit.edu.

[36] J. Myers. RFC 1939: Post Office Protocol - Version 3, May 1996.

[37] Netcraft. Anti-Phishing Toolbar. http://news.netcraft.com/archives/2004/12/28/netcraft_antiphishing_tool%bar_available_for_download.html.

[38] ZDNet News. http://news.zdnet.com/2100-9595_22-519795.html?legacy=zdnn.

[39] Ping Identity Corporation. SourceID Toolkit. http://www.pingidentity.com.

[40] Jonathan B. Postel. Simple Mail Transfer Protocol. RFC 821, August 1982.

[41] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT '01*, volume 2248 of *LNCS*, pages 552–565. Springer Verlag, 2001.

[42] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, May 1998.

[43] Bruce Schneier. Safe Personal Computing. Schneier On Security Weblog, December 2004. http://www.schneier.com/blog/archives/2004/12/safe_personal_c.html.

[44] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer Verlag, 1985.

[45] The Spamhaus Project. The Spamhaus Block List. http://www.spamhaus.org/sbl/.

[46] Tumbleweed Communications. Digitally-Signed Emails to Protect Against Phishing Attacks. http://www.tumbleweed.com/solutions/finance/antiphishing.html.

[47] Phil Zimmerman. Pretty Good Privacy. http://www.pgp.com.