Lightweight Email Signatures (Extended Abstract)

Ben Adida[1], David Chau[1], Susan Hohenberger[2,⋆], and Ronald L. Rivest[1]

[1] CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2] IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon

Abstract. We present *Lightweight Email Signatures* (LES), a simple cryptographic architecture for authenticating email. LES is an extension of DKIM, the recent IETF effort to standardize domain-based email signatures. LES shares DKIM's ease of deployment: they both use the DNS to distribute a single public key for each domain. Importantly, LES supports common uses of email that DKIM jeopardizes: multiple email personalities, firewalled ISPs, incoming-only email forwarding services, and other common uses that often require sending email via a third-party SMTP server. In addition, LES does not require DKIM's implied intra-domain mechanism for authenticating users when they send email.

LES provides these features using identity-based signatures. Each domain authority generates a master keypair, publishes the public component in the DNS, and stores the private component securely. Using this private component, the authority delivers to each of its users, via email, an individual secret key whose identity string corresponds to the user's email address. A sender then signs messages using this individual secret key. A recipient verifies such a signature by querying the appropriate master public key from the DNS, computing the sender's public key, and verifying the signature accordingly. As an added bonus, the widespread availability of user-level public keys enables deniable authentication, such as ring signatures. Thus, LES provides email authentication with optional repudiability.

We built a LES prototype to determine its practicality. Basic user tests show that the system is relatively easy to use, and that cryptographic performance, even when using deniable authentication, is well within acceptable range.

1 Introduction

1.1 The State of Email and DKIM

Email has become a highly polluted medium. More than 75% of email volume is spam [27], and phishing attacks – spoofed emails that trick users into revealing private information – are on the rise, both in volume [3] and sophistication [20]. Email users are repeatedly warned that an email's `From:` field cannot be trusted [35], and that links distributed by email should not be followed [2,29]. Still, studies show that users remain highly vulnerable, even to low-tech phishing attempts [11].

⋆ Research performed while at the Massachusetts Institute of Technology.

Domain Keys & Identified Mail (DKIM) is a promising proposal for providing a foundation to solve the phishing problem: domains are made cryptographically responsible for the email they send. Roughly, `bob@foo.com` sends emails via `outgoing.foo.com`, which properly identifies Bob and signs the email content. The public key is distributed via a DNS TXT record for `_domainkeys.foo.com`. The details of how DKIM should handle mailing lists, message canonicalization, message forwarding, and other thorny issues, are being resolved in the context of a recently-formed IETF Working Group [18].

1.2 Lightweight Email Signatures

We propose *Lightweight Email Signatures*, abbreviated LES, as an extension to DKIM. We show how LES preserves all of the major architectural advantages of DKIM, while offering three significant improvements:

1. **Automatic Intra-Domain Authentication**: DKIM assumes that server `outgoing.foo.com` can tell its users `bob@foo.com` and `carol@foo.com` apart, which is not a safe assumption in a number of settings – e.g. university campuses or ISPs that authenticate only the sending IP address. By contrast, LES authenticates users without requiring additional authentication infrastructure within `foo.com`.
2. **Flexible Use of Email (Better End-to-End)**: LES allows Bob to send email via any outgoing mail server, not just the official `outgoing.foo.com` mandated by DKIM. This is particularly important when supporting existing use cases. Bob may want to alternate between using `bob@foo.com` and `bob@bar.com`, while his ISP might only allow SMTP connections to its outgoing mail server `outgoing.isp.com`. Bob may also use his university's alumni forwarding services to send email from `bob@alum.univ.edu`, though his university might not provide outgoing mail service.
3. **A Privacy Option**: LES enables the use of repudiable signatures to help protect users' privacy. Bellovin [6] and other security experts [32,7] warn that digitally signed emails entail serious privacy consequences. We believe the option for repudiable signatures can alleviate these concerns.

In a nutshell, LES provides more implementation flexibility for each participating domain – in particular flexibility that addresses *existing legitimate uses of email* –, without complicating the domain's public interface. A LES domain exposes a single public key in the DNS, just like DKIM. A LES domain can implement DKIM-style, server-based signatures and verifications, or user-based signatures and verifications where each user has her own signing key.

1.3 The LES Architecture

We now describe the LES architecture as diagrammed in figure 1.

The DKIM Baseline. A LES-signed email contains an extra SMTP header, `X-LES-Signature`, which encodes a signature of a canonicalized version of the

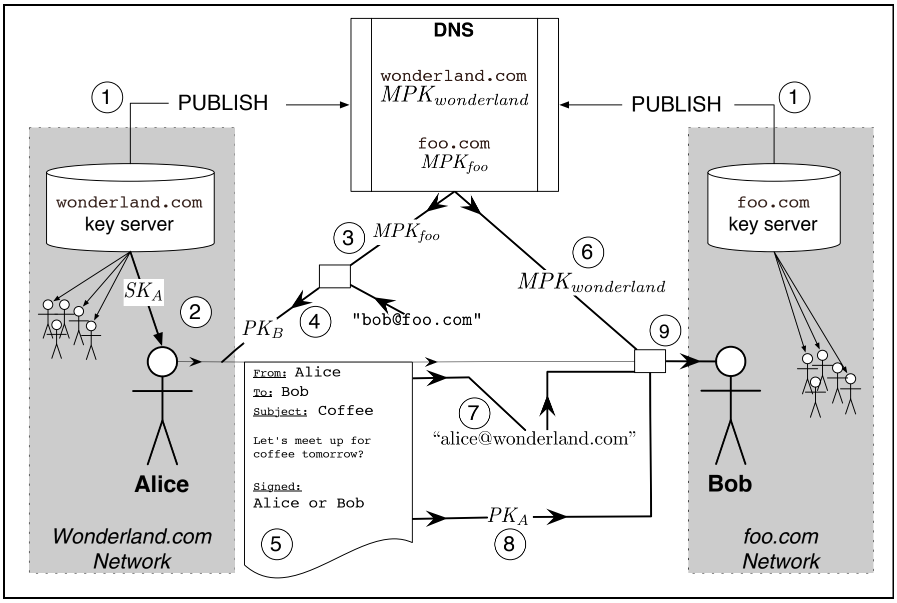

Fig. 1. LES: (1) The domain keyservers for Alice and Bob publish their *MPKs* in the DNS (2) Alice's domain sends Alice her secret key SK_A, via email (3) Alice obtains the MPK for Bob's domain, `foo.com` (4) Alice computes Bob's public key PK_B (5) Alice signs her email with a ring signature and sends it to Bob (6) Bob obtains the MPK for Alice's domain, from the DNS (7) Bob extracts the `From:` field value, `alice@wonderland.com`, from the email (8) Bob computes Alice's public key PK_A, using the claimed identity string "`alice@wonderland.com`" (9) Bob verifies the signature against the message and PK_A

message. We leave to the DKIM Working Group the details of this canonicalization – which includes the `From:` field, the subject and body of the message, and a timestamp –, as they do not impact the specifics of LES. Verification of a LES-signed email is also quite similar to the DKIM solution: the recipient retrieves the sender domain's public key from a specially crafted DNS record, and uses it to verify the claimed signature on the canonicalized message.

Limitations of DKIM. A DKIM domain uses a single key to sign all of its emails. This simple architecture is what makes DKIM so appealing and easy to deploy. Not surprisingly, it is also the source of DKIM's limitations: users must send email via their approved outgoing mail server, and this outgoing mail server must have some internal method of robustly distinguishing one user from another to prevent `bob@foo.com` from spoofing `carol@foo.com`. LES aims to overcome these limitations while retaining DKIM's deployment simplicity.

User Secret Keys with Identity-Based Signatures. LES assigns an individual secret key to each user, so that `bob@foo.com` can sign his own emails. This means Bob can use any outgoing server he chooses, and `outgoing.foo.com` does not need to authenticate individual users (though it may, of course, continue to use any mechanism it chooses to curb abusive mail relaying.)

To maintain a single domain-level key in the DNS, LES uses *identity-based signatures*, a type of scheme first conceptualized and implemented in 1984 by Shamir [36]. A LES domain publishes (in the DNS) a master public key MPK and retains the counterpart master secret key MSK. Bob's public key, $PK_{\texttt{Bob}}$, can be computed using MPK and an identification string for Bob, usually his email address "`bob@foo.com`". The corresponding secret key, $SK_{\texttt{Bob}}$, is computed by Bob's domain using MSK and the same identification string. Note that, contrary to certain widespread misconceptions, identity-based signatures are well tested and efficient. Shamir and Guillou-Quisquater signatures, for example, rely on the widely-used RSA assumption and are roughly as efficient as normal RSA signatures.

One might argue that a typical hierarchical certificate mechanism, where the domain certifies user-generated keypairs, would be just as appropriate here. There are some problems with this approach. First, a user's public-key certificate would need to be sent along with every signed message and would require verifying a chain of two signatures, where the identity-based solution requires only one signature and one verification operation. Second, with user-generated keypairs, it is much more difficult to use ring signatures (or any of the known deniable authentication methods) between a sender and a receiver who has not yet generated his public key. The identity-based solution ensures the availability of any user's public key.

Distributing User Secret Keys via Email. LES delivers the secret key $SK_{\texttt{Bob}}$ by sending it via email to `bob@foo.com` [14], using SMTP/TLS [17] where available. Thus, quite naturally, only someone with the credentials to read Bob's email can send signed emails with `bob@foo.com` as `From` address. Most importantly, as every domain already has *some* mechanism for authenticating access to incoming email inboxes, this secret-key delivery mechanism requires no additional infrastructure or protocol.

Privacy with Deniable Signatures. Privacy advocates have long noted that digital signatures present a double-edged sword [6,32,7]: signatures may make a private conversation publicly-verifiable. The LES framework supports many forms of deniable authentication [8] through its use of identity-based keys: Alice can create a deniable signature using her secret key $SK_{\texttt{Alice}}$ and Bob's public key $PK_{\texttt{Bob}}$. Only Bob can meaningfully verify such a signature. We note that this approach does not provide anonymity beyond that of a normal, unsigned email. However, unlike DKIM and other signature proposals, LES does not make the signature publicly-verifiable: only the email recipient will be convinced.

1.4 A Prototype Implementation

To determine the practical feasibility of deploying LES, we built a basic prototype, including a key server and a plugin to the Apple Mail client. We deployed a real MPK in the DNS for `csail.mit.edu`, using the Guillou-Quisquater identity-based scheme [15] for its simplicity, and ring signatures for deniability. We then conducted a small test with nine users. Though our test was too small to provide complete, statistically significant usability results, we note that most participants were able to quickly install and use the plugin with no user-noticeable effect on performance.

Detailed performance numbers, in section 6, show that basic ring signature and verification operations perform well within acceptable limits – under 40ms on an average desktop computer –, even before serious cryptographic optimizations. A small keyserver can easily compute and distribute keys for more than 50,000 users, even when configured to renew keys on a daily basis.

The complete prototype's source code is available for download at `http://crypto.csail.mit.edu/projects/antiphishing/`.

1.5 Previous and Related Work

The email authentication problem has motivated a large number of proposed solutions. End-to-end digital signatures for email have repeatedly been proposed [4,39] as a mechanism for making email more trustworthy and thereby preventing spoofing attacks such as phishing. One proposal suggests labeling email content and digitally signing the label [16]. Apart from DKIM, all of these past proposals require some form of Public-Key Infrastructure, e.g. X.509 [13], although the idea of using the DNS for identity-based master key distribution has appeared once before in the context of email encryption and IPSEC [37]. Alternatively, path-based verification has been proposed in a plethora of initiatives. Those which rely on DNS-based verification of host IP addresses were reviewed by the IETF MARID working group [19,24,23]. The latest proposal in this line of work is SIDF [30].

A number of spam-related solutions have been suggested to fight phishing. Blacklists of phishing mail servers are sometimes used [38,25], as is content filtering, where statistical machine learning methods are used to detect likely attacks [34,26,28]. Collaborative methods [12] that enable users to help one another have also been proposed. LES can help complement these approaches.

1.6 This Paper

In section 2, we review the necessary cryptographic and systems building blocks. In section 3, we detail the LES system based on these building blocks, specifically the identity-based key distribution infrastructure and the repudiability option. We then briefly explore the issue of technology adoption in section 4, discuss the threats model for LES in section 5, and describe our prototype and performance results in section 6. More detailed comments on these three issues appear in the appendix. Finally, we conclude in section 7.

2 Cryptographic and System Preliminaries

We now review and present new extensions to cryptographic and system building blocks involved in LES.

2.1 Identity-Based Signatures

In 1984, Shamir proposed the concept of identity-based signatures (IBS) [36]. Since then over a dozen schemes have been realized based on factoring, RSA, discrete logarithm, and pairings. (See [5] for an overview, plus a few more in [1].) Most IBS signatures can be computed roughly as fast as RSA signatures, and those based on pairings can be 200 bits long for the equivalent security of a 1024 bit RSA signature.

IBS schemes were introduced to help simplify the key management problem. Here, a single master authority publishes a master public key MPK and stores the corresponding master secret key MSK. Users are identified by a character string *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 the same *id_string*, then delivered to the user.

2.2 Ring Signatures from Any Keypairs

Ring signatures [9,33] allow an individual to sign on behalf of a group of individuals without requiring any prior group setup or coordination. Although rings can be of any size, consider the two party case. Suppose Alice and Bob have keypairs $(PK_{\texttt{Alice}}, SK_{\texttt{Alice}})$ and $(PK_{\texttt{Bob}}, SK_{\texttt{Bob}})$ respectively. Alice can sign on behalf of the group "Alice or Bob" using her secret key $SK_{\texttt{Alice}}$ and Bob's public key $PK_{\texttt{Bob}}$. Anyone can verify this signature using both of their public keys. We require the property of *signer-ambiguity* [1]; that is, *even if Alice and Bob reveal their secret keys*, no one can distinguish the actual signer.

In the full version of this paper, we describe a compiler for creating signer-ambiguous ring signatures using keypairs of almost any type. That is, Alice may have a PGP RSA-based keypair and Bob may have a pairing-based identity-based keypair, yet Alice can still create a ring signature from these keys! For our purposes here, it does not matter *how* this compiler works. It is enough to know that: (1) the security of the resulting ring signature is equivalent to the security of the weakest scheme involved, and (2) the time to sign (or verify) a ring signature produced by our compiler is roughly the sum of the time to sign (or verify) individually for each key involved, plus an additional hash. See [1] for the technical details.

Using ring signatures for deniable authentication is not a new concept [33,7]. The idea is that, if Bob receives an email signed by "Alice or Bob," he knows Alice must have created it. However, Bob cannot prove this fact to anyone, since he *could* have created the signature himself. In section 3.4, we describe how ring signatures are used to protect a user's privacy in LES.

2.3 Email Secret-Key Distribution

Web password reminders, mailing list subscription confirmations, and e-commerce notifications all use email as a semi-trusted messaging mechanism. This approach, called Email-Based Identity and Authentication [14], delivers semi-sensitive data to a user by simply sending the user an email. The user gains access to this data by authenticating to his incoming mail server in the usual way, via account login to an access-controlled filesystem, webmail, POP3 [31], or IMAP4 [10]. For added security, one can use SMTP/TLS [17] for the transmission.

3 Lightweight Email Signatures

We now present the complete design of LES, as previously illustrated in Figure 1.

3.1 Email Domain Setup

Each email domain is responsible for establishing the cryptographic keys to authenticate the email of its users. The setup procedure for that master authority run by `wonderland.com` is defined as follows:

1. select one of the identity-based signatures (IBS) discussed in section 2.1. (For our section 6 experiment, we chose the RSA-based Guillou-Quisquater IBS [15] because of its speed and simplicity.)
2. generate a master keypair $(MPK_{\texttt{wonderland}}, MSK_{\texttt{wonderland}})$ for this scheme.
3. define key issuance policy *Policy*, which defines if and how emails from this domain should be signed. (Details of this policy are defined in the full version of this paper.)
4. publish $MPK_{\texttt{wonderland}}$ and *Policy* in the DNS as defined by the DKIM specifications.

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*.

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 *issuing_domain* parameter into the user identity character string. Note that `foo.com` should always refuse to issue secret keys for identity strings whose *issuing_domain* is not `foo.com`. However, `foo.com` may choose to issue a key for `alice@wonderland.com`, as long as the *issuing_domain* within the identity string is `foo.com`. We provide a clarifying example shortly.

Key Types. The LES infrastructure may be expanded to other applications in the future, such as encryption. To ensure that a key is used only for its intended purpose, we include type information in *id_string*. 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 key revocation capabilities, the user identity string includes 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, which include an indication for the timezone. For now, we default to UTC for timezone disambiguation.

Constructing Identity Character Strings. An *id_string* is thus constructed as: $\langle$*issuing_domain*$\rangle$, $\langle$*email*$\rangle$, $\langle$*expiration_date*$\rangle$, $\langle$*type*$\rangle$. For example, a 2006 LES identity string for email address `bob@foo.com` would be: `foo.com,bob@foo.com, 2006-12-31,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,2006-12-31,lightsig`. Here `lightsig.org` happily issues a secret key for Bob, even though his email address is not within the `lightsig.org` domain. This is legitimate, as long as the *issuing_domain* in 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 most recent 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 an algorithm specific to the chosen IBS scheme (most schemes have such an algorithm), or by attempting to sign a few messages with the new key and then verifying those results. (For more details, see section 2.3.)

3.4 The Repudiability Option

The downside of signing email is that it makes a large portion of digital communication undeniable [6,32,7]. An off-the-record opinion confided over email to a once-trusted friend may turn into a publicly verifiable message on a blog! We believe that repudiable signatures should be the *default* to protect a user's privacy as much as possible, and that non-repudiable signatures should be an option for the user to choose.

Numerous approaches exist for realizing repudiable authentication: designated-verifier signatures [21], chameleon signatures [22], ring signatures [33], and more (see [8] for an overview of deniable authentication with RSA). In theory, any of these approaches can be used. We chose the ring signature approach for two reasons: (1) it fits seamlessly into our identity-based framework without creating new key management problems, and (2) our ring signature compiler can create ring signatures using keys from different schemes, as discussed in section 2.2. Thus, no domain is obligated to use a single (perhaps patented) IBS scheme.

Let us explore why ring signatures are an ideal choice for adding repudiability to LES. Most repudiation options require the sender to know something about the recipient; in ring signatures, the sender need only know the receiver's public key. In an identity-based setting, the sender Alice can easily derive Bob's public key using the $MPK_{\texttt{foo.com}}$ for `foo.com` in the DNS and Bob's *id_string*. Setting the *issuing_domain* to `foo.com`, the *type* to `lightsig`, and the email field to `bob@foo.com` for Bob's *id_string* is straight-forward. For *expiration_date*, Alice simply selects 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 repudiability. Such requests for back-dated keys can simply be handled by signed email to the keyserver.

This "Alice or Bob" authentication is valid: if Bob is confident that *he* did not create it, then Alice must have. However, this signature is also repudiable, because Bob cannot convince a third party that he did not, in fact, create it. In the full version of this paper, we discuss what Alice should do if `foo.com` does not yet support LES, and in section 4, we discuss methods for achieving more repudiability.

3.5 Signing and Verifying Messages

Consider Alice, `alice@wonderland.com`, and Bob, `bob@foo.com`. On `2006-09-06`, Alice wants to send an email to Bob with subject $\langle \textit{subject} \rangle$ and body $\langle \textit{body} \rangle$. When Alice clicks "send," her email client performs the following actions:

1. prepare a message $\mathcal{M}$ to sign, using the DKIM canonicalization (which includes the `From:`, `To:`, and `Subject:` fields, as well as a timestamp and the message 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 $\textit{id\_string}_{\texttt{Bob}}$, an identity string for Bob using `2006-09-06` as the *expiration_date*: `foo.com,bob@foo.com,2006-09-06,lightsig`
   (c) compute $PK_{\texttt{Bob}}$ from $MPK_{\texttt{foo.com}}$ and $\textit{id\_string}_{\texttt{Bob}}$. (We assume that $PK_{\texttt{Bob}}$ contains a cryptosystem identifier, which determines which IBS algorithm is used here.)
3. sign the message $\mathcal{M}$ using $SK_{\texttt{Alice}}$, $MPK_{\texttt{wonderland.com}}$. Optionally, for repudiability, also use $PK_{\texttt{Bob}}$ and $MPK_{\texttt{foo.com}}$ with the section 2.2 compiler. The computed signature is σ.
4. using the DKIM format for SMTP header signatures, add `X-LES-Signature` containing σ, $\textit{id\_string}_{\texttt{Alice}}$, and $\textit{id\_string}_{\texttt{Bob}}$.

Upon receipt, Bob needs to verify the signature:

1. obtain the sender's email address, `alice@wonderland.com`, and the corresponding domain name, `wonderland.com`, from the email's `From` field.
2. obtain $MPK_{\texttt{wonderland.com}}$, using DNS lookup (as specified by DKIM).

3. ensure that $PK_{\texttt{Alice}}$ is correctly computed from the claimed $\textit{id\_string}_{\texttt{Alice}}$ and corresponding issuing domain $MPK_{\texttt{wonderland.com}}$, 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 canonical message $\mathcal{M}$ that was signed, using the declared `From`, `To`, and `Subject` fields, the email body, and the timestamp.
5. If Alice applied an ordinary, non-repudiable signature, verify $\mathcal{M}$, σ, $PK_{\texttt{Alice}}$, $MPK_{\texttt{wonderland.com}}$ to check that Alice's signature is valid.
6. If Alice applied a repudiable signature, Bob **must** check that this signature verifies against both Alice's and his own public key following the proper ring verification algorithm [1]:
   (a) ensure that $PK_{\texttt{Bob}}$ is correctly computed from the claimed $\textit{id\_string}_{\texttt{Bob}}$ 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 and type).
   (b) verify $\mathcal{M}$, σ, $PK_{\texttt{Alice}}$, $MPK_{\texttt{wonderland.com}}$, $PK_{\texttt{Bob}}$, $MPK_{\texttt{foo.com}}$ to check that this is a valid ring signature for "Alice or Bob."

If all verifications succeed, Bob can be certain that this message came from someone who is authorized to use the address `alice@wonderland.com`. If the `wonderland.com` keyserver is behaving correctly, that person is Alice.

3.6 LES vs. Other Approaches

The LES architecture provides a number of benefits over alternative approaches to email authentication. We consider three main competitors: SIDF [30] and similar path-based verification mechanisms, S/MIME [40] and similar certificate-based signature schemes, and DKIM, the system upon which LES improves. A comparison chart is provided in table 1, with detailed explanations as follows:

Table 1. LES compared to other approaches for authenticating email. ‡: PGP and S/MIME can be adjusted to issue keys from the server, somewhat improving scalability.

Property	**SIDF**	**S/MIME**	**DKIM**	**LES**
Logistical Scalability	No	No‡	No	**Yes**
Deployable with Client Update Only	No	**Yes**	No	**Yes**
Deployable with Server Update Only	**Yes**	No‡	**Yes**	**Yes**
Support for Third-Party SMTP Servers	No	**Yes**	No	**Yes**
Easy Support for Privacy	**Yes**	No	No	**Yes**
Email Alias Forwarding	No	**Yes**	**Yes**	**Yes**
Support for Mailing Lists that Modify Content	**Good**	Poor	Fair	Fair

1. **Logistical Scalability:** When a large organization deploys and maintains an architecture for signing emails, it must consider the logistics of such a deployment, in particular how well the plan scales. With SIDF or DKIM, domain administrators must maintain an inventory of outgoing mail servers and ensure that each is properly configured. This includes having outgoing

mail servers properly authenticate individual users to prevent intra-domain spoofing. Meanwhile, with certificate-based signature schemes, domain administrators must provide a mechanism to issue user certificates. By contrast, LES does not require any management of outgoing mail servers or any additional authentication mechanism. LES only requires domains to keep track of which internal email addresses are legitimate, a task that each domain already performs when a user's inbox is created. Thus, LES imposes only a small logistical burden, while DKIM, SIDF, and S/MIME all require some new logistical tasks and potentially new authentication mechanisms. Note that it is technically possible to use PGP in a way similar to LES, with email-delivered certificates, though the PGP keyserver then needs to keep track of individual user keys where LES does not.

2. **Deployment Flexibility:** SIDF and DKIM can only be deployed via server-side upgrades, which means individual users must wait for their domain to adopt the technology before their emails become authentic. PGP can only be deployed via client-side upgrades, though one should note that many clients already have PGP or S/MIME support built in. LES can be implemented either at the server, like DKIM, or at the client, like PGP.
3. **Support for Third-Party SMTP Servers:** SIDF and DKIM mandate the use of pre-defined outgoing mail servers. A user connected via a strict ISP may not be able to use all of his email personalities. Incoming-mail forwarding services – e.g. alumni address forwarding – may not be usable if they do not also provide outgoing mail service. PGP and LES, on the other hand, provide true end-to-end functionality for the sender: each user has a signing key and can send email via any outgoing mail server it chooses, regardless of the `From` email address.
4. **Privacy:** LES takes special care to enable deniable authentication for privacy purposes. SIDF, since it does not provide a cryptographic signature, is also privacy-preserving. DKIM and S/MIME provide non-repudiable signatures which may adversely affect the nature of privacy in email conversations. (Note that is is *not* valid to claim that DKIM signatures are repudiable because the server signs messages instead of the user; either the server is trustworthy or it isn't.) Even a hypothetical LES-S/MIME hybrid, which might use certificates in the place of identity-based signatures, would not provide adequate privacy, as the recipient's current public key would often not be available to the sender without a PKI.
5. **Various Features of Email:** SIDF does not support simple email alias forwarding, while S/MIME, DKIM, and LES all support it easily. SIDF supports mailing lists and other mechanisms that modify the email body, as long as mailing list servers support SIDF, too. On the other hand, S/MIME, DKIM, and LES must specify precise behavior for mailing lists: if the content or `From` address changes, then the mailing list must re-sign the email, and the recipient must trust the mailing list authority to properly identify the original author of the message. This is particularly difficult for S/MIME, which must assume that the mailing list has an S/MIME identity, too, that recipients trust (this is related to the PKI requirement of S/MIME-like solutions).

LES provides a combination of advantages that is difficult to obtain from other approaches. Of course, these features come at a certain price: new security threats. We explore these LES-specific threats in section 5.

4 Technology Adoption

The most challenging aspect of cryptographic solutions is their path to adoption and deployment. The deployment features of LES resembles those of DKIM: each domain can adopt it independently, and those who have not yet implemented it will simply not notice the additional header information. Like DKIM, LES allows each domain to express a DNS-based policy about its use of signatures, letting certain high-risk organizations – e.g. financial institutions – simply declare that all emails should be LES-signed, while other organizations – e.g. small ISPs – may allow both signed and unsigned emails.

LES offers two distinct advantages over DKIM in technology adoption. LES can be deployed *either at the mail server or client* without altering the DNS LES record. LES can also be deployed using *alternate domain authorities* to let users adopt LES individually before their email domain has adopted it. Once again, this can be done without changes to the DNS records.

Details about these deployment extensions are in the full version of this paper, including mechanisms for deployment of the repudiability option when the recipient hasn't yet deployed LES or when the recipient is a mailing list.

5 Threats

LES shares enough in architectural design with DKIM that both systems face a number of common threats. For example, both solutions can be compromised by DNS spoofing, domain key compromise, zombie user machines, and user confusion. Fortunately, the unique properties of LES help to mitigate some DKIM-specific threats, such as the ability to keep the domain secret key offline and allowing for recovery from user key compromise without a DNS update.

Of course, the unique properties of LES also cause certain unique threats to emerge, such as potentially increasing user confusion and allowing for new denial of service attacks. We examine all these threats in detail in the full version.

6 Experimental Results

We implemented a complete LES environment using Guillou-Quisquater identity-based signatures [15] based on the RSA assumption. Ring signatures were formed using a CDS proof of partial knowledge construction [1]. Our implementation includes a web-based key distribution server and a plugin to the Apple Mail client that implements key storage, message signing with repudiability, and signature verification. We used Python for the server-side components, and Objective C with the GNU Multi-Precision Library for the client-side Apple Mail plugin.

Table 2. Performance estimates for an average of 1000 runs. Time is in milliseconds. The sizes are in bytes and do not include encoding overhead. The symbol * indicates the number includes an estimated 50 bytes for the identity string of the user.

		1024-bit modulus		**2048-bit modulus**	
Operation	**Machine**	**Time**	**Size**	**Time**	**Size**
Master Keypair Generation	server	143	200	1440	300
User Secret Key Computation	server	167	178*	1209	316*
User Public Key Computation	client	0.03	178*	0.03	316*
Ring Signature of 100K msg	client	37	575*	210	1134*
Ring Verification of 100K msg	client	37	N/A	211	N/A

For space reasons, the details of this implementation are provided in the full versionof this paper with a summary here. Briefly, our implementation shows that performance of the LES architecture is quite reasonable for transparent deployment. A small server can manage keys for tens of thousands of users, and the average desktop computer takes only 37ms to sign or verify a message. (Even with 2048-bit keys, signing/verification take only 210ms, before optimizations).

Experimental Setup. We ran server benchmarks on a single-processor, 3.2Ghz Intel Pentium 4 with 2 Gigs of RAM and 512MB of L2 cache, running Fedora Core Linux with kernel v2.6.9. We used Python v2.3.3. We instrumented the Python code using the standard, built-in `timeit` module, running each operation 1000 times to obtain an average performance rating. We did not make any overzealous attempts to cut down the number of standard background processes.

We ran client benchmarks on a 1.5Ghz Apple Powerbook G4 with 1.5Gigs of RAM, running Mac OS X 10.4.4. We instrumented the Objective C code using the built-in Cocoa call to `Microseconds()`, which returns the number of microseconds since CPU boot. We ran each operation 1000 times to obtain an average running time. Though we were not actively using other applications on the Powerbook during the test, we also made no attempt to reduce the typically running background processes and other applications running in a normal Mac OS X session.

7 Conclusion

We proposed Lightweight Email Signatures (LES), an extension to DKIM which conserves its deployment properties while addressing a number of its limitations. LES allows users to sign their own emails and, thus, to use any outgoing mail server they choose. This helps to preserve a number of current uses of email that DKIM would jeopardize: choosing from multiple email personalities with a single outgoing mail server because of ISP restrictions, or using special mail forwarding services, e.g. university alumni email forwarding, that do not provide an outgoing mail server.

LES also offers better privacy protection for users. Each individual email address is associated with a public key, which anyone can compute using only the domain's master public key available via DNS. With the recipient's public key available, any number of deniable authentication mechanisms can be used, in particular the ring signature scheme we propose.

Our prototype implementation shows that LES is practical. It can be quickly implemented using well-understood cryptographic algorithms that rely on the same hardness assumptions as typical RSA signatures.

We are hopeful that proposals like DKIM and LES can provide the basic authentication foundation for email that is so sorely lacking today. These cryptographic proposals are not complete solutions, however, much like viewing an SSL-enabled web site is not a reason to fully trust the site. Reputation systems and "smart" user interfaces will likely be built on the foundation that DKIM and LES provide. Without DKIM or LES, however, such reputation systems would be nearly impossible.

Acknowledgments

We wish to thank Rob Miller and Min Wu for their helpful pointers to and explanations of user-interface-related solutions to the phishing problem. We are also grateful to Seth Gilbert and Steve Weis for their comments. We thank our ten volunteer anonymous testers, as well as Greg Shomo and Matt McKinnon who provided the equipment and system administration for our pilot test. Susan Hohenberger was supported by an NDSEG Fellowship.

References

1. B. Adida, S. Hohenberger, and R. L. Rivest. Ad-hoc-group signatures from hijacked keypairs, 2005. `http://theory.lcs.mit.edu/~rivest/publications`.
2. American Banking Association. Beware of Internet Scrooges this Holiday. `http://biz.yahoo.com/prnews/041209/dcth013_1.html`.
3. Anti-Phishing Working Group. `http://www.antiphishing.org/`.
4. Anti-Phishing Working Group. Digital Signatures to Fight Phishing Attacks. `http://www.antiphishing.org/smim-dig-sig.htm`.
5. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In EUROCRYPT, pp. 268–286, 1999.
6. S. M. Bellovin. Spamming, phishing, authentication, and privacy. *Inside Risks, Communications of the ACM*, 47:12, December 2004.
7. N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *WPES '04*, pp. 77–84. ACM Press, 2004.
8. D. R. Brown. Deniable authentication with rsa and multicasting. In *Cryptology ePrint Archive, Report 2005/056*, 2005.
9. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In CRYPTO, pp. 174–187, 1994.
10. M. Crispin. RFC 1730: Internet Mail Access Protocol - Version 4, Dec. 1994.
11. R. Dhamija and J. D. Tygar. Phish and hips: Human interactive proofs to detect phishing attacks. In *HIP*, vol. 3517 of *LNCS*, pp. 127–141, 2005.
12. E. D. et. al. Spam Attacks: P2P to the Rescue. In *WWW '04*, pp. 358–359, 2004.
13. M. C. et. al. Internet X.509 Public Key Infrastructure (latest draft). *IETF Internet Drafts*, Jan. 2005.

14. S. L. Garfinkel. Email-Based Identification and Authentication: An Alternative to PKI? *IEEE Security & Privacy*, 1(6):20–26, Nov. 2003.
15. L. C. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *CRYPTO*, vol. 403, pp. 216–231, 1988.
16. A. Herzberg. Controlling spam by secure internet content selection. In *4th Security in Communication Networks (SCN)*, vol. 3352 of LNCS, pp. 337–350, 2004.
17. P. Hoffman. SMTP Service Exten. for Secure SMTP over Transport Layer Security. Internet Mail Consortium RFC. `http://www.faqs.org/rfcs/rfc3207.html`.
18. IETF. The DKIM Working Group. `http://mipassoc.org/dkim/`.
19. IETF. MTA Authorization Records in DNS (MARID), June 2004. `http://www.ietf.org/html.charters/OLD/marid-charter.html`.
20. M. Jakobsson. Modeling and Preventing Phishing Attacks. In A. Patrick and M. Yung, editors, *Financial Cryptography '05*, LNCS, 2005.
21. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT '96*, vol. 1233 of *LNCS*, 1996.
22. H. Krawczyk and T. Rabin. Chameleon signatures. In *Network and Distributed System Security (NDSS)*, 2000.
23. J. Levine, A. DeKok, and et al. Lightweight MTA Authentication Protocol (LMAP) Discussion and Comparison, Feb. 2004. `http://www.taugh.com/draft-irtf-asrg-lmap-discussion-01.txt`.
24. J. R. Levine. A Flexible Method to Validate SMTP Senders in DNS, 2004. `http://www1.ietf.org/proceedings_new/04nov/IDs/draft-levine-fsv-01.txt`.
25. MAPS. RBL - Realtime Blackhole List, 1996. `http://www.mail-abuse.com/services/mds_rbl.html`.
26. J. Mason. Filtering Spam with SpamAssassin. In *HEANet Conference*, 2002.
27. MessageLabs. Annual Email Security Report, Dec. 2004. `http://www.messagelabs.com/intelligence/2004report`.
28. T. Meyer and B. Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system. In *Conference on Email and Anti-Spam*, July 2004.
29. Microsoft. Phishing Scams: 5 Ways to Help Protect Your Identity. `http://www.microsoft.com/athome/security/email/phishing.mspx`.
30. Microsoft. The Sender ID Framework. `http://www.microsoft.com/mscorp/safety/technologies/senderid/default.mspx`.
31. J. Myers. RFC 1939: Post Office Protocol - Version 3, May 1996.
32. Z. News. `http://news.zdnet.com/2100-9595_22-519795.html?legacy=zdnn`.
33. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT '01*, vol. 2248 of *LNCS*, pp. 552–565, 2001.
34. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization*, May 1998.
35. B. Schneier. Safe Personal Computing. Schneier On Security Weblog, Dec. 2004. `http://www.schneier.com/blog/archives/2004/12/safe_personal_c.html`.
36. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO '84*, vol. 196 of *LNCS*, pp. 47–53, 1985.
37. D. Smetters and G. Durfee. Domain-based administration of identity-based cryptosystems for secure email and IPSEC. In *USENIX Security Symposium*, 2003.
38. The Spamhaus Project. The Spamhaus Block List. `http://www.spamhaus.org/sbl/`.
39. Tumbleweed Communications. Digitally-Signed Emails to Protect Against Phishing Attacks. `http://www.tumbleweed.com/solutions/finance/antiphishing.html`.
40. P. Zimmerman. Pretty Good Privacy. `http://www.pgp.com`.