Fighting Phishing Attacks: A Lightweight Trust Architecture for Detecting Spoofed Emails

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

Abstract

We present a novel key distribution architecture and a novel use of a particular identity-based digital signature scheme for making email trustworthy. Like typical digital signatures, our solution fights email-based phishing attacks and mitigates spam by detecting spoofed emails. Unlike typical digital signatures, our approach requires no complex, preestablished public-key infrastructure nor cooperation between email domains. Furthermore, it provides just enough trust to make email useful again, but not too much: email remains repudiable. All current legitimate uses of email – alternate email personalities, alternate outgoing mail servers, PGP or S/MIME encryption, sending attachments, web-based email etc. . . – remain fully functional. The end-to-end nature of email is preserved: the only requirements are an upgraded email client and at least one keyserver. We call this approach a Lightweight Trust Architecture.

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 [34], most of it spoofed. Even more worrisome are **email-based phishing attacks**: spoofed emails that trick users into revealing private information, usually banking credentials requested on a fraudulent web site linked in the email body. The cost of these attacks is estimated at more than $10B this year alone, and the number of attacks is on the rise at a staggering 28% per month [4].

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

†Supported by an NDSEG Fellowship.

The consequences are devastating to email as a communication medium. Banking institutions have been reduced to recommending that their users not click on links in emails [3]. Security specialists are instructing users not to trust the `From:` field [9]. The very openness that originally made email so easy to use is now threatening to make the medium completely unusable.

1.2 A Two-Pronged Solution

We present a novel solution to the email-based phishing problem in two steps:

1. the design of *a lightweight public-key infrastructure* to distribute identity-based public keys using existing hardware and software architectures.

2. the use of *repudiable identity-based signatures* that convince an email recipient of the sender's authenticity, without allowing that recipient to prove this authenticity to any third party.

Taken together, these two ideas constitute a **Lightweight Trust Architecture** (LTA) for email. This architecture allows users to regain trust in the email medium without altering users' expectations of repudiability and privacy.

1.3 Previous and Related Work

Email-based phishing is a relatively new, but well-documented [11], beast with two major components: email hooks and web collection sites. Defenses exist on both fronts, often stemming from very different fields.

Phishing Emails. On the email front, a number of spam-related solutions have been applied to phishing. Content filtering is a common method, using statistical machine learning methods to detect likely attacks [31, 28, 48], or collaborative methods [14] to enable users to

help one another. Blacklists of spamming/phishing mail servers are also used [49, 32], though the recent attacker techniques that hijack virus-infected PCs from around the world makes this technique far less useful.

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 [24, 1, 26]. Another major proposal, Yahoo's DomainKeys [33], suggests a cryptographic proof of path, where the outgoing mail server, certified via DNS, signs all outgoing emails. Conceptually, these proposal are all very similar, though their handling of special cases – mailing lists, mail forwarding, bouncing – varies.

End-to-end digital signatures for email have been proposed [5, 50] as a mechanism for making email more trustworthy and thereby preventing phishing. One other proposal suggests labeling email content and digitally signing the label [20]. All of these proposals require some form of Public-Key Infrastructure, e.g. X.509 [15].

Web Phishing. 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 [39, 21], or sometimes email-to-web gateway warnings [16]. All of these schemes attempt to counter the all-too-effective social engineering tactics of phishing attacks. Because web sites have a much stronger accountability architecture than emails, these techniques can successfully flag worrisome web sites by reputation management.

Repudiability. Repudiability, the ability to deny a prior act of authentication, is often thought to be a security weakness in numerous protocols, particularly those where authentication needs to be audited – e.g. contracts. In the context of email and casual conversation, however, repudiability is a highly desirable feature among the privacy-conscious. Recent work on repudiable instant-messenger conversations notes this concern [38]. In the context of privacy-preserving protocols, repudiability is also called *plausible deniability*.

1.4 Just Enough Trust

Email has always been a casual communications medium: email content is not authenticated, and forging an email is an easy operation. These properties should not be lightly discounted: they are, in fact, features which preserve casual conversation and user privacy. Our approach strives to make email senders accountable to their recipients, without otherwise changing user expectations.

Email remains casual as long as it is repudiable. Today's email is universally forgeable: anyone can forge anyone else's email address. However, there is a large, exploitable gap between repudiability – *someone else* could have written this email – and forgeability – *anyone else* could have written this email .

We use signatures that limit forgeability but maintain repudiability: someone other than the sender could have written the email, but not just anyone. In particular, if Alice sends Bob an email, then Bob can be certain that Alice sent the email. However, if Bob shows the email and signature to his friend Charlie, Charlie will be unable to determine which of Alice or Bob authored the email. This mechanism allows anyone to forge emails destined to themselves. That is just enough to provide repudiability, but not so much as to open Pandora's Box of spoofed emails.

1.5 Protocol Overview

We define the design of our novel identity-based key distribution infrastructure and our novel use of a recent identity-based repudiable signature scheme. The components described here are diagrammed in Figure 1.

1.5.1 Identity-Based Cryptography

First conceptualized in 1984 by Shamir [46] and first implemented in 1988 by Guillou and Quisquater [19], *identity-based signatures* are typical asymmetric operations 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 must thus be delivered to the user by the master authority. These same concepts are used in Identity-Based Encryption, first implemented by Boneh and Franklin in 2001 [8].

One major advantage of identity-based cryptography is its ability to perform cryptographic operations involving a user's public key before the corresponding secret key has been generated.

1.5.2 An Identity-Based Public Key Infrastructure

We propose a practical mechanism built on today's existing software infrastructure to distribute identity-based keypairs. 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`.

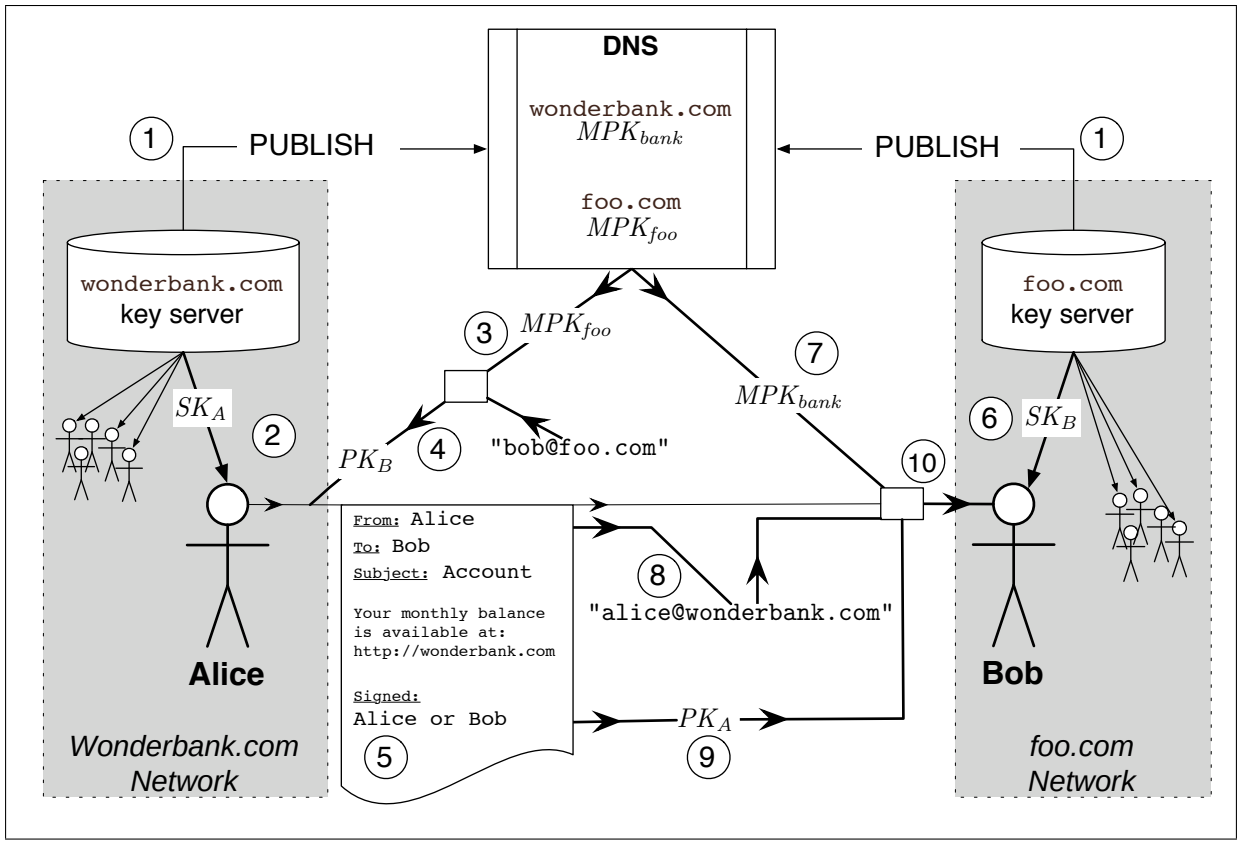

Figure 1: Key Distribution Mechanism in Lightweight Trust Architecture: (1) Domain key servers for Alice and Bob publish their respective MPK in their DNS records (2) Alice's domain sends her her secret key SK_A (3) Alice fetches MPK_{foo}, the master public key for Bob's domain (4) Alice computes PK_B, Bob's public key, given his id_string and MPK_{foo} (5) Alice signs her email with a repudiable signature that verifies against Alice's or Bob's public key, and sends the email to Bob (6) Optionally, Bob obtains his secret key SK_B from his key server (7) Bob extracts the sender address `alice@wonderbank.com` from the email (8) Bob obtains $MPK_{wonderbank}$, the master public key for Alice's domain, from the DNS (9) Bob extracts Alice's claimed public key PK_A and claimed id_string from the email (10) Bob verifies the message against the signature, the signature against PK_A, the `From` address against the claimed id_string, and PK_A and the sender id_string against $MPK_{wonderbank}$.

DNS Public-Key Certification. With each email domain functioning as an independent identity-based master authority, master public keys corresponding to each domain must be distributed and certified in some fashion. For this purpose, we use the DNS system, much like DomainKeys [33]. 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. The threat model also fits: if an adversary were to gain control over a domain's DNS record, they would be able to hijack users' email addresses completely. There is no sense in a defense stronger than the current mechanism for email routing itself. Most importantly, if DNS security improves in the future – e.g. with DNSSEC [13] –, our master key distribution mechanism will automatically gain this increased level of security.

Email Secret-Key Distribution. 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 Trust exploits the existing trust between a user and his incoming mail server, a concept called Email-Based Identity and Authentication [47]. 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 mail server itself. However, a master authority on a separate machine, closely linked on the network or communicating via SMTP/TLS

[40] with the mail server, would work well, too.

1.5.3 Repudiable Signatures

Once identity-based keypairs are distributed, we need a signature scheme that supports repudiation. We use an approach often mentioned in the cryptography literature [43]: ad-hoc group signatures.

Ad-Hoc Group Signatures (aka Ring Signatures). An *ad-hoc group signature scheme* lets Alice create a signature for a message with an arbitrarily chosen signatory group, as long as that group includes Alice herself and all other group members' public keys are available. Anyone can confirm that one member of this group must have computed the signature, but the individual signer's identity cannot be determined. The schemes we're interested in hide the identity of the actual signer with information-theoretical security: even if all secret keys are subpoenaed, the identity of the actual signer remains hidden amongst all group members.

Repudiable signatures in two-person communication are achieved using a specially-formed signatory group. Alice sends Bob an email signed with an ad-hoc group signature whose signatory group includes exactly Alice and Bob. Upon receipt, Bob is confident that Alice authored the email, since he remembers that he did not author the email himself (we assume Bob is not a sleepwalking emailer). However, if Bob shows Charlie the email, he cannot make the case that Alice was the actual signer: Bob might have authored the message. In other words, emails signed in this manner might as well have been forged by their recipient. Only the recipient can tell the difference.

Identity-Based and Cross-Domain Support. A typical ad-hoc group signature scheme requires a public-key infrastructure to distribute the user public keys necessary to form a group. In order to get around this deployment non-starter, Lightweight Trust employs identity-based keypairs and the identity-based public key distribution mechanism previously described. Thus, we need and implement an *identity-based ad-hoc group signature scheme* [52].

In practice, just as our identity-based key distribution mechanism notes, not all users belong to the same identity-based master domain. We need a signature scheme that supports the formation of signatory groups that span multiple identity-based domains. This cross-domain functionality is often called "separability" in the cryptography literature [10], a nomenclature relating to the number of system parameters – e.g. the master public keys – that can be separated among parties. Recent work by the authors of this paper [7], based on components introduced by Hess [22] and Abe et. al. [2], addresses this need with a *separable, identity-based, ad-hoc group signature scheme*.

With this scheme, Alice can create an ad-hoc group signature using a signatory group that includes herself and Bob, the intended recipient of her message. Alice can derive Bob's public key directly from his email address and some basic DNS information. Alice and Bob may well belong to different domains. They may in fact not know each other at all. With identity-based cryptography and cross-domain support, there is no need for any special setup.

1.6 Bootstrap and Adoption

The most promising aspect of our proposed solution to email spoofing is the practicality of its deployment by commercial entities who have a large financial stake in eradicating phishing attacks.

Thanks to identity-based cryptography and our novel public-key distribution mechanism, bootstrapping is practical. With each commercial adoption comes an immediate benefit: that entity's legitimate emails are immediately authenticated using simple software and existing infrastructure. Thus, once an institution adopts Lightweight Trust, it can immediately protect its customers by guiding them to perform the proper software upgrade. This software upgrade is non-specific: all institutions benefit from having their users perform the same upgrade.

Just as important, our solution is highly compatible with the mechanism by which most users get their email: web-based email providers. Customers of these web-based email providers can be transparently protected as soon as their provider adopts LTA verification.

We believe that our solution's potential for real-world success is largely due to these favorable deployment characteristics.

1.7 This Paper

In section 2, we explore the motivation for our work in detail, including the costs of phishing attacks, the reasons why current approaches are insufficient, and the high-level needs. In section 3, we define the necessary cryptographic and systems building blocks. In section 4, we detail the protocol based on these building blocks, specifically the identity-based key distribution infrastructure and the repudiable signature scheme. We then explore the issue of technology adoption with bootstrap mechanisms and commercial incentives in section 5. We expand on the basic protocol with extensions – including mailing list support – and optimizations in section 6. In section 7, we note how our scheme enables the estab-

lishment of a reputation management system for email addresses and domains. We wrap up with performance metrics in section 8 and some concluding thoughts in section 9.

2 Motivation & Requirements

2.1 The Cost of Phishing Attacks

Phishing attacks date back to the mid-1990s, when social engineering techniques were used to surreptitiously obtain AOL account passwords. In the context of email, phishing attacks appeared in mid-2003 with the same ultimate goal of fraudulently obtaining sensitive information, such as web site login credentials. The targets of phishing attacks are often customers of financial institutions. Since 2003, the attacks have become more intricate in their spoofing technique, fooling even savvy Internet users. Gartner estimates that 3% of phishing recipients fall victim to the scam [18], and the Anti-Phishing Working Group [4] estimates that, in 2004, phishing attacks were responsible for more than $10B in fraud-related expenses.

With few solutions in sight, banks have resorted to desperate recommendations, asking their users not to click on any link they receive by email [3]. The FTC has echoed this recommendation [17], as has Microsoft [36]. Their recommended alternative is for users to manually type in URLs.

In other words, the cost of phishing attacks is greatly reduced email functionality: users cannot trust email, and consequently legitimate institutions can use it only for very limited communication with their members. Without a real fix, email may become so untrustworthy or so feature-poor as to end up completely unusable.

2.2 Current Approaches Are Not Enough

Numerous solutions to email-based phishing have been presented over the past two years. Some of these, particularly those based on machine-learning techniques to detect fraudulent patterns, propose a solution that requires continual tweaking and accepts a non-negligible fraction of false negatives. Other solutions provide stronger trust but cause radical changes in the usage of email, either in functionality or user expectation. In the context of phishing attacks, where a false negative is very costly and a practical solution is needed rapidly, none of these is sufficient.

Content-Based Verification. A large portion of anti-spam solutions also used against phishing provide filtering based on machine learning of different types of email content. Because spam content is qualitatively different than that of legitimate emails, these techniques work well in that context. However, phishing attacks are trickier: phishing emails are meant to resemble legitimate emails as closely as possible. Filtering on email content is thus bound to fail to detect advanced phishing attacks.

A somewhat unique content-based approach is Eudora's ScamWatch [16]. This mechanism warns users of suspiscious-looking URLs in the emails they receive, specifically URLs that appear to direct a user to a known institutional site, when in fact the real clickthrough URL is very different. Like other content-based approaches, ScamWatch may well provoke an arms race between the phishers and anti-phishers, with anti-phishers defining ever-newer, more intricate rules, and phishers constantly finding new techniques to outsmart them.

One less-obviously content-based approach is whitelist filtering, where only emails with the right `From` address are accepted. Because these sender addresses are not certified, they are effectively content items against which to filter. In the case of phishing attacks, these filters are particularly counter-productive: actual customers of spoofed institutions will likely receive the emails, while non-customers will block them. In other words, the attacks are less likely to raise a red flag and be reported.

Path-Based Authentication. A recent breed of anti-spam and anti-phishing solution, notably Sender Policy Framework (SPF) [29], prevents spoofed emails by checking an email's network path. Specifically, an email domain declares, in an SPF record, which hosts are allowable starting points for emails authored by users of that domain. A recipient's incoming mail server can then check the sending host's IP address against these allowable origins.

Another solution is DomainKeys [33]. In fact, this technology is conceptually similar to SPF in that it also provides email origin verification. Unlike SPF, DomainKeys uses a cryptographic proof of origin: the outgoing mail server signs all emails. The email recipient's incoming mail server can check the validity of this signature against the domain's public key stored in its DNS record.

Though these approaches are more likely to prevent phishing than content-based solutions, they present significant problems. The most notable is that they break the end-to-end nature of email. Emails from a given domain can now only originate from certain mail servers. Users connecting to the network in a restricted fashion, e.g. users behind strong firewalls, may not be able to reach their usual outgoing mail server. More importantly, forwarding services, including mailing lists, will break

these schemes because the path of the email is altered and the sender domain no longer matches the SMTP network path.

Thus, these approaches fail to support important, legitimate use cases for email.

Typical Digital Signatures. One solution that does not violate the end-to-end nature of email is the use of normal digital signatures using PGP [53] or S/MIME [23]. With signed emails, recipients are inherently protected from spoofing. A reputation management system can be established to determine the trustworthiness of various email domains and addresses.

However, these non-identity-based solutions require a PKI, e.g. X.509 certificates [15]. Practically speaking, this requirement is a non-starter in a non-controlled environment like the public Internet [47]. In addition, these non-repudiable signatures provoke a radical change in user expectations: emails run the risk of becoming legally-binding contracts [38, 12].

In terms of deployment and user expectation, typical digital signatures are impractical and inappropriate.

MAC-Based Authentication. Another cryptographic approach to authenticated communications with repudiation is a MAC-based mechanism where Alice uses her secret key and Bob's public key to generate a shared secret. This shared secret is then used to key a Message Authentication Code on the message content. This approach is the one used in very recent work by Borisov et. al. on secure instant-messenger conversations [38] , precisely for the purpose of obtaining limited authentication and subsequent repudiability.

In fact, in that very paper, Borisov et. al. briefly suggest adapting their techniques to email by exchanging the first set of keys using ring signatures for repudiability. Their solution attempts to provide the same features we do, but is inadequate in terms of deployment: without identity-based keypairs and a complete public-key infrastructure, deployment of this MAC-based solution for email is highly impractical.

2.3 High-Level Requirements

A solution to the phishing problem should be immediately usable, should preserve current email functionality, and should respect current user expectations of the medium.

A Practical Key Distribution Infrastructure. The problem calls for an infrastructure with minimal and practical deployment requirements. Given the focus on email security, each email domain should be able to manage independently its security choices and key distribution activities – generation, revocation, etc.... In addition, all parameters and key data required to interact with a user of a given domain should be easily and securely obtained using existing architecture.

End-to-End Architecture A number of legitimate email use cases do not fit the clean pattern expected by some proposed phishing solutions. Some users manage multiple email addresses from a single mail client. Others travel with their laptop and expect to use whichever outgoing mail server they can reach from random local networks (internet cafes, hotels, etc...). All of these usage cases contribute to the ubiquity and simplicity of email; a solid anti-phishing approach should respect the end-to-end nature of email [44] and support these atypical situations. Only a mail client upgrade should be necessary.

Repudiability without Forgeability. The ubiquity and freedom of email is in large part due to its repudiability. Email messages constitute casual conversation, not contractually binding exchanges. Interestingly, it is this ease-of-forgeability that has created the phishing problem and greatly aggravated the spam problem: because the sender cannot be verified, the sender address can be spoofed, and the recipient can be fooled.

Thus, our goal is to authenticate email senders enough to convince the recipient, but not so much that the recipient can prevent the sender from repudiating the message. The way to tease apart these two issues is to identify the gap between minimal repudiability – *someone* might have forged this message – and total forgeability – *anyone* could have forged this message. Certainly, some applications require non-repudiation: a proper solution should also support non-repudiable signatures via the same infrastructure.

3 Building Blocks

3.1 DNS Public-Key Certification

The DNS system 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 [33]. In the DomainKeys scheme, RSA public keys are stored in specially named DNS records.

Specifically, DomainKeys reserves a subdomain `_domainkeys` 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. Though DNS records can technically spread fairly rapidly, the DNS time-to-live architecture is not optimized for intensive update activity. In our scheme, we propose to use this mechanism only for domain-wide, long-standing keys, of which there will be few and which will generally last as long as generic DNS host records. We specifically do not distribute user-specific keys via this mechanism.

3.2 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 [47], 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 [27], or IMAP4 [30].

Certainly, one concern with this approach is that an adversary might tap the network and intercept the sensitive data in question. 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 [40] between the master domain authority and the mail server.

3.3 Identity-Based, Separable, Ad-Hoc Group Signatures

Group Signatures. Group signatures allow a sender $\mathcal{S}$, member of a pre-determined group $\mathcal{G}$, to produce a signature σ on a message $\mathcal{M}$ such that

1. anyone can verify that σ was computed by a member of group $\mathcal{G}$ (using one or more public keys identifying the group).
2. no one, not even the other members of $\mathcal{G}$, can recognize which member of $\mathcal{G}$ computed σ.

Classic group signatures require a group manager to set up each member's secret key for each group. The structure of the group is thus centrally controlled and fairly static. While this is a feature in a number of cases, other applications call for more decentralized, ad-hoc group formation.

Ad-Hoc Group Signatures. Ring signatures (see [43] and the references cited therein) provide this ad-hoc property, where any user $\mathcal{S}$ can generate a group signature using any group $\mathcal{G}$ of members with available public keys (assuming $\mathcal{S}$ is in $\mathcal{G}$, of course). The keypairs need not even be specifically tailored to the ring-signature application: the implementation of ring signatures given in [43], for example, uses normal RSA [42] keypairs. The most important constraint of these ring signatures is the Public-Key Infrastructure (PKI). Users can only create ad-hoc groups composed of members with available, certified public keys.

Identity-Based Ad-Hoc Group Signatures. Identity-based ring signatures [52], conceptualized by Shamir [46] and implemented by Guillou and Quisquater [19], use the typical identity-based keypairs and thus do not require a PKI. In such a scheme, Bob's public key PK_B is directly computable from his email address `bob@foo.com` and a master public key MPK. In this setting, Bob must obtain his secret key SK_B from the master authority, which holds MSK, the secret counterpart to MPK. SK_B is derived from the identity string `bob@foo.com` and MSK, and the master authority delivers SK_B to Bob once Bob has properly identified himself.

Identity-Based, Separable, Ad-Hoc Group Signatures. Recently, identity-based, separable, ad-hoc group signatures were developed [7] to address the practical, multiple identity-domain setting. In the typical identity-based manner, users' public keys are derived from their identity character string and a master public key. The improvement lies in that groups can be composed of members who derive their public keys from different master authorities, and thus different MPK_i.

An additional improvement [7] lies in that keypairs generated for other purposes – e.g. identity-based encryption – can be reused for our purposes instead of generating new, application-specific keypairs. This type of signature allows for maximal interoperability with minimal setup requirements.

More specifically, a separable, identity-based, ad-hoc group signature scheme provides six algorithms:

- Generate(key_length)
  outputs (MPK, MSK), a master public and corresponding master secret key of prescribed key length.
- ComputePublicKey(MPK, id_string)
  outputs PK, a public key that corresponds to the character string id_string within the domain indicated by MPK. id_string is simply a string representation of a user's identity, usually in a well-defined format that includes an email address and additional validity parameters.
- ComputeSecretKey(MSK, id_string)
  outputs SK, the secret key that corresponds to the public key PK derived from the input id_string.
- VerifySecretKey(MPK, PK, SK)
  outputs True or False, depending on whether SK matches the PK in the given master domain indicated by MPK. It's important to note that this verification can be performed without the master secret MSK.
- Sign($\mathcal{M}, SK_1, PK_1, MPK_1, PK_2, MPK_2$)
  outputs σ, a group signature on $\mathcal{M}$ with group members identified by PK_1 and PK_2, their respective domain public keys MPK_1 and MPK_2, and the corresponding secret key SK_1 for PK_1 in domain MPK_1. For simplicity of notation, only two members are represented here, but this function can take a group of any size, as long as at least one secret key is provided.
- GroupMembers(σ)
  outputs a list of (PK_i, MPK_i), the public keys of members in the group and their corresponding master public keys.
- Verify($\mathcal{M}, \sigma, PK_1, MPK_1$)
  outputs True or False depending on whether σ verifies as a proper group-signature on message $\mathcal{M}$ using a group that includes the user with public key PK_1 in domain indicated by MPK_1.

We note that, in a number of schemes of this type, and specifically in the one considered [7], the identity of the signer is information-theoretically secure: even if all secret keys are revealed (for example, in the case of a subpoena), one cannot ever determine which group member computed the signature.

3.4 Repudiability from Group Signatures.

As has been suggested in the literature [43], we can use group signatures to enable repudiability without forgeability. Specifically, an ad-hoc group signature crafted 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 did not author the email himself.
2. the sender can repudiate his message at a later date, 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 –, but completely 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 6).

4 The Protocol

4.1 Email Domain Setup

In the basic setup, an email user will obtain his secret key from a master authority dedicated to managing the security of that user's email domain. For example, if Alice's email address is `alice@wonderland.com`, she will obtain her secret key from the key server responsible for `wonderland.com`. Thus, every email domain becomes an identity-based master domain. The actual master authority keyserver may be the incoming mail server itself.

The setup procedure for a master authority for domain `wonderland.com` is defined as follows:

1. use Generate to obtain a master keypair (MPK, MSK).
2. define key issuance policy $Policy$ (details of this policy are defined in section 5.1).
3. publish MPK and $Policy$ in a DNS record within subdomain `_lta.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.

If a mail domain is large and requires, for administrative purposes, multiple keyservers each with independent cryptographic keys, additional master public key records can be added to the DNS subdomain

_lta.wonderland.com. While a particular user – e.g. alice@wonderland.com – may routinely obtain her secret key against only one of these master public keys, it is important that, if requested, she be allowed to obtain a secret key against *any* of the advertised master public keys. This capability will ensure repudiability in all cases.

One should note that MPK might eventually be certified by a stronger mechanism than DNS records. Our current scheme fully supports this improvement. In fact, the DNS record could contain, in addition to MPK, a certificate of some kind for MPK. However, the size of such a certificate makes this approach somewhat risky, given assumptions that current DNS servers make about the limited size of DNS records. Instead, a more immediate improvement of this key-distribution method will likely occur using DNSSEC [13], an authentication extension to DNS.

4.2 User Identities

Per the identity-based construction, a user's public key PK can be derived from any character string id_string using ComputePublicKey, where id_string is some representation of 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 5), Bob might obtain a secret key for bob@foo.com from a domain *other than* foo.com.

For this purpose, we build a key_domain parameter into the user identity character string. This parameter indicates the domain whose master public key will issue the users's corresponding secret key. 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 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. The identity character strings we specify here are meant for repudiable signatures. Keys derived from other identity character strings within the same master domain may be used for encryption, while others still may be used for both purposes. In order to ensure that keys are used only for their intended purpose, we propose including type information in id_string. Thus, our identity character string format becomes reusable by other cryptographic schemes.

Consider $type$, a character string composed only of lowercase alphanumeric characters including only a-z and 0-9. This type is part of the overall identity string. For the purposes of our application, we define a single type: ltasig.

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 [25].

Constructing Identity Character Strings. Putting the previous elements together, id_string is 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,ltasig

If Bob were to obtain his secret key from a master authority for a different domain, e.g. lta.org, his public key would necessarily be derived from a different identity character string:

lta.org,bob@foo.com,2005-12-31,ltasig

Notice that lta.org will happily issue a secret key for that identity string, even though the email address is not within the lta.org domain. This is legitimate, because key_domain matches the issuing key server.

4.3 Delivering User Secret Keys

Each domain keyserver will choose to issue user secret keys at certain regular intervals, possibly weekly or monthly. These keys are delivered by email, with a well-defined format – e.g. XML – 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 signature scheme's VerifySecretKey operation allows any mail client to check the correctness of the secret key it receives against its domain's master public key.

We note that, given that a keyserver is domain-specific, the connection between the keyserver and incoming mail server can be secured against external network sniffing attacks. In particular, an upgraded mail server can eventually integrate keyserver capability and deliver keys by direct injection into users' mailboxes. Alternatively, the master domain keyserver can connect to the mail server using SMTP/TLS [40].

Back-Dated Keys. In addition to these regularly issued keys with predictably spaced expiration dates, a user should have the ability to request secret keys corresponding to his email address with a back-dated expiration date. Though few users will actually proceed with this key retrieval in practice, their ability to do so at any time contributes to the repudiability of emails they receive.

Thus, each keyserver should allow a user to request a key for a given email address at that domain and a given back-dated expiration date, maybe via a web interface or an email interface. Because these requests for back-dated keys may not be authenticated, typical measures of IP address, email address, and number-of-keys-per-user throttling should be used to prevent Denial-of-Service attacks against given email addresses. Back-dated keys are delivered in the usual way, via email to the appropriate recipient.

4.4 Signing & Verifying Messages

Consider Alice, `alice@wonderland.com`, and Bob, `bob@foo.com`. On date `2005-02-04`, 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. obtain MPK_B, the master public key for Bob's email domain `foo.com`, using DNS record lookup.
2. assemble id_string_B, an identity string for Bob using `2005-02-04` as the $expiration\_date$: `foo.com,bob@foo.com,2005-02-04,ltasig`
3. compute PK_B from MPK_B and id_string_B using ComputePublicKey.
4. prepare a message $\mathcal{M}$ to sign:

`From:` `alice@wonderland.com`
`To:` `bob@foo.com`
`Subject:` $\langle subject \rangle$
`Timestamp:` $\langle timestamp \rangle$
`Body:` $\langle body \rangle$

5. apply Sign to $\mathcal{M}$, SK_A, PK_A, PK_B, MPK_A, MPK_B, and obtain σ
6. include σ in base64-encoding in SMTP header `X-LTASignature`, the exact id_string corresponding to PK_A in SMTP header `X-LTASenderIDString`, the exact id_string corresponding to PK_B in SMTP header `X-LTARecipientIDString`, and the timestamp used in the signature in SMTP header `X-LTATimestamp`

Upon receipt, Bob needs to verify the signature:

1. determine the sender's email address according to the email's `From:` field, and extract the sender's domain name.
2. obtain MPK_A, using DNS record lookup on the email domain name.
3. extract the public keys from the group signature using GroupMembers(σ). Ensure that there are only two group member public keys, PK_A and PK_B.
4. ensure that PK_B is correctly computed from the claimed `X-LTARecipientIDString` and MPK_B, and that this id_string is acceptable (includes Bob's email address, a valid expiration date, a valid type).
5. ensure that PK_A is correctly computed from the claimed `X-LTASenderIDString` and MPK_A, and that this id_string is acceptable (includes Alice's email address exactly as indicated in the `From` field, a valid expiration date, a valid type).

   (We assume, at this point, that PK_A's key_domain matches the email domain for Alice: `wonderland.com`. Some variants are described in Section 6.)
6. 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-LTATimestamp`.
7. apply Verify to $\mathcal{M}$, σ, PK_A, MPK_A to verify that Alice is in the signatory group.
8. apply Verify to $\mathcal{M}$, σ, PK_B, MPK_B to verify 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_A produced σ. If Alice's master domain keyserver is behaving correctly, that signer can only be Alice.

4.5 The Non-Repudiability Option.

Alice may want to sign her email to Bob with a stronger, non-repudiable signature. This is, of course, her decision. Our solution enables this option immediately: Alice can simply choose to apply a signature σ to her message $\mathcal{M}$ with a signatory group including only her public key.

This leads to an important observation: *our identity-based key distribution architecture functions just as well for simple identity-based signatures and encryption*. It can be employed for any number of identity-based cryptographic operations. Repudiability is an important feature of the system, but the sender may certainly opt out.

5 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 Trust. We also show how our solution lends itself quite well to commercial adoption, a property which bodes well for its long-term viability and the immediate goal of eliminating email-based phishing fraud.

5.1 Bootstrapping Lightweight Trust

Alice wishes to send an email to Bob. If both have upgraded email clients, and both domains are LTA-enabled, they can proceed as described in section 4 to achieve Lightweight Trust. What happens, however, if one of these elements is not yet in place?

Alternate Master Domains. The preeminent bootstrap situation occurs when Alice wishes to adopt LTA 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. `lta.org`. `lta.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 obtains such a key, Alice will have to explicitly sign up with `lta.org`, most likely via a web interface. Her id_string will read:
`lta.org,alice@wonderland.com,2005-12-31,ltasig`
Note that we do not expect `lta.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 `lta.org`, much like MIT runs the MIT PGP Keyserver [37], where anyone can freely use the service. Another possibility is that existing identity services – e.g. Microsoft Passport [35] or Ping Identity [41] – will issue LTA keys for a small fee. Where PGP requires large, mostly centralized services like the MIT PGP Keyserver, LTA users may choose from any number of keyservers.

Domain Policies. Once an email domain deploys LTA by adding the proper keyserver and specifying the appropriate DNS records, the domain's users may not all be using the upgraded mail client with LTA support. Others may be using the upgraded mail client, but might have already obtained a key from an alternate master domain. In certain cases, like that of small Internet Service Providers whose users are free to choose their level of email security, this variety of user setups may be perfectly acceptable. In other cases, like that of banking institutions, it is important that all users comply with a stricter domain policy in order to ensure that customers are protected.

The DNS LTA $Policy$ parameter exists for this purpose. Once an email domain deploys LTA, it can advertise, through the same DNS means used to deploy the domain's MPK, the domain's policy on issuing secret keys. Specifically, two parameters should be declared within $Policy$: $ExternalPolicy$ and $InternalPolicy$.

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

$InternalPolicy$ determines the requirements made of that domain's email users. Three possible values should be considered:

- None: users of this domain are not required to sign their emails. This option is generally chosen during the testing phase, before a domain has fully deployed LTA.
- Basic: users of this domain are required to sign their emails, but may do so with a key issued by any key server.
- Strong: users of this domain are required to sign their emails with a key issued by this domain.

Sender Options. When Alice wishes to send an email to Bob, she may notice that Bob's domain `foo.com` does not support LTA. 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 a service like `lta.org` that has an advertised DNS policy of issuing keys for external users. Thus, Alice may use:
`lta.org,bob@foo.com,2005-02-05,ltasig`
as id_string from which to derive a PK for Bob.

Note that Bob need not ever actually retrieve a secret key from the `lta.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 LTA 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.

Recipient Options. 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 with 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, 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 trust).
  - If the policy is either Basic or Strong, Bob can confidently reject 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 Basic, then Bob can trust the email as much as he is willing to trust this alternate master authority.
  - If $InternalPolicy$ is Strong, then Bob can confidently reject the message.

Note that Bob need not be concerned if the signatory group does not include one of his own public keys, as this lack of recipient signature only hinders Alice's ability to repudiate, something Bob cannot control in the first place. However, if the signatory group includes third parties, Bob will have to trust *all* group members. Specifically, the requirements found in `wonderland.com`'s $InternalPolicy$ will have to succeed for every signatory group member.

This question of additional signatory group members is addressed in greater detail in Section 6.

5.2 Commercial Adoption

With these bootstrapping techniques and the inherent deployment advantages of identity-based cryptography, it is sensible for commercial entities to become early adopters of Lightweight Trust. In fact, it is through these commercial entities that the full deployment of Lightweight Trust can occur.

A bank, e.g. `bigbank.com`, can rapidly set up Lightweight Trust for all of its internal users, including the delivery of secret keys and the publication of its master public key in DNS. Customers of that bank can then be instructed to upgrade their email client, guaranteeing that any email received from `bigbank.com` is either verifiably authentic, or worth throwing away. In other words, phishing attacks from `bigbank.com` can be eradicated by simple software updates.

Most importantly, though each domain can deploy Lightweight Trust without any coordination with other domains, the software upgrade required for recipients is non-specific: an email user who performs the upgrade is immediately protected from emails spoofing any compliant institution. And, as we will see in section 6, many Internet Service Providers will be able to deploy the upgrade at the server level: the average consumer can be protected immediately and tranparently.

6 Extensions and Optimizations

The protocol defined in Section 4 works in simple, person-to-person email situations. In practice, not every email situation is that simple: some emails are addressed to multiple people, while others still are addressed to mailing lists.

6.1 Evaporating Keys

Lightweight Trust offers repudiability because someone in possession of a secret key *other than the message author's* might have signed the message. Another approach to achieving this same end-goal is to set up an **evaporating key**, a public key derived from a particular id_string, whose secret key counterparts are publicly revealed at regular intervals.

An evaporating key has an expiration date, like other keys defined for LTA. 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 deniable at evaporation time, since the secret key is available for anyone to use.

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.

In general, if Alice wants to use an evaporating key, she should obtain such a key from her own domain `wonderland.com`, or from another domain `lta.org` that she trusts for this purpose, regardless of her email recipient's domain. The reason for this choice

is repudiability: the point of an evaporating key is repudiation, a service that benefits the email sender more than anyone else. Thus, Alice must trust the evaporating key issuer to eventually publish the secret counterpart to the evaporating public key she is selecting. To ensure trust, Alice should choose her evaporating domain. The best way is to have Alice's domain itself, `wonderland.com`, provide this service.

Mailing Lists For mailing list messages, the end-to-end nature of our proposed sender signature guarantees that the recipient can trust the authenticity of the email, regardless of the amount of forwarding the message endures along its path to eventual delivery. Where repudiation is concerned, however, the recipient address rewriting may prevent the sender from achieving repudiability against the proper public key.

This is precisely where evaporating keys come in. When Alice sends an email to a mailing list, she may not know the eventual recipients of the email. By using a signatory group that includes her public key and an evaporating key, recipients of the email can trust Alice's authorship, and Alice gains repudiability as soon as the evaporation interval comes to an end.

One complication exists because email clients (and sometimes individuals using the email client) are unaware of which email addresses are mailing lists, and which are individuals. Email users who want to ensure repudiability in all cases can create a signatory group that includes both the recipient public key and an evaporating key.

More Repudiation The repudiability presented so far is important, but it is incomplete: a third party can still determine, even days after an email was sent, that the signer must have been either the sender or recipient. Some users will want stronger deniability.

By including an additional public key in their signatory group, specifically an evaporating public key, senders of email can guarantee that emails are initially sender-deniable, and, eventually, completely repudiable. Thus, a user can include an additional evaporating key in every email signature they send. This additional key becomes an additional avenue for repudiation.

Weaknesses of Evaporating Keys Given evaporating keys' utility, 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. If the recipient timestamps his incoming message with some trustworthy timestamping service before the evaporating key expiration, he can effectively cancel the repudiability of the signature.

For this purpose, it's important to always include the recipient public key in the signatory group as an additional avenue for repudiability. The evaporating key and the recipient key play two different roles: repudiability in time and author, respectively. Each is important in its own way.

6.2 Optimizations

Though the Lightweight Trust Architecture respects the end-to-end nature of email and does not require a mail server upgrade, numerous optimizations can be performed with an upgraded incoming or outgoing mail server. Specifically, the trust relationship that inherently exists between a user and his incoming mail server is significant: an incoming mail server already has access to all incoming email, makes some spam-related decisions, etc... Similarly, many outgoing mail servers also require authentication to prevent spam mail relaying.

Thus, in this situation, we assume that the incoming mail server, and the domain master authority are one and the same. In some situations, we assume that this new system includes outgoing mail server functionality, too. We explore the optimizations that result.

DNS MPK Lookups. The most important optimization for clients is the DNS lookup of remote MPK records. An incoming mail server can easily look up the MPK records 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 Verifications. The incoming mail server can go even further and perform the signature check, 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 low-computation-power clients, like cell phones or PDAs.

Signature Creation. If a user sends email through his regular outgoing mail server, the repudiable signature can be applied by an appropriately upgraded outgoing mail server. This optimization is also particularly useful for 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 LTA. If Internet Service Providers begin implementing LTA, they can combine the previous optimizations to provide all LTA functionality at the server. In other words, without ever upgrading their mail client, home users can get all the benefits of LTA if signature creation, verification, and lookups are performed by their mail server. This is particularly appealing to Internet Service Providers who can deploy this solution for their users 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 doesn't 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.

6.3 Multiple Recipients

If Alice explicitly sends an email to multiple recipients Bob, Charlie, and David, her mail client will need to generate a separate repudiable signature for each recipient. For maximal repudiability, it is important that Alice not send all signatures to all recipients. Instead, she should send only the appropriate signature to each recipient.

Unfortunately, this approach may reduce the common optimization of mail clients which send a single copy of such messages to their outgoing mail server and expect it to perform the appropriate copying and routing to all recipients. One approach to reestablishing this optimization is the Transparent LTA method described previously: if the outgoing mail server is given the ability to sign emails on behalf of its users, then the sender can once again offload the distribution work onto the server. Alternatively, the sender could use an evaporating key approach to provide rapid repudiability in a single signature.

In any case, the authenticity of the sender is never questioned. The tradeoff involves only the amount of repudiability the sender desires.

7 Reputation Management

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.

Such identification enables the deployment of reliable reputation management systems, both for individual email addresses and email domains as a whole. It is through these reputation management techniques that certain complex phishing attacks can truly be thwarted. Specifically, certain phishing attacks may pass the initial LTA signature test because they originate at a domain that is different from but closely resembles a known institution – e.g. `cit1bank.com`. With reputation management, these techniques can be countered, but only if LTA forces phishing attacks to resort to these social engineering approaches.

It is also with reputation management that spam can be mitigated: email addresses and entire domains that are responsible for spam can be marked as spamming senders, while legitimate email users are protected from impersonation through LTA signatures.

In the end, *LTA provides accountability*. It is this accountability which provides a solid foundation for making trust decisions.

8 Performance Estimates

Recent work on repudiable, identity-based signatures [7] provides flexibility in one's choice of underlying cryptographic technology. In particular, the identity-based component can be implemented using well-proven RSA [42] or Discrete-Log assumptions. Alternatively, the newer identity-based schemes based on bilinear maps [7] offer the same signature functionality using keys that also provide identity-based encryption capability [8, 51].

Even these latest schemes – which are slower due to their use of bilinear maps – offer very usable performance on today's hardware. We assume a very conservative key size of 256 bits for these elliptic-curve cryptography operations (160 bits is often considered comparable to 1024-bit RSA).

Using performance estimates from the MIRACL cryptographic libraries [45] and its recent uses [6], we estimate that, on a typical 1.8Ghz AMD Athlon processor, a signature operation performed in software will take 20ms, as will a verification. These numbers indicate perfectly acceptable performance for client-based computing. Even a server can process 50 such signatures per second. With an optimized server – using either higher processing power or dedicated cryptographic hardware – it's likely that a medium-sized mail server could process upwards of 100 signatures or verifications per second.

9 Conclusion

In this paper, we present two novel proposals. First, we describe *the design and deployment of an identity-based public key infrastructure using today's tools*, including DNS and email-based authentication. Second, we de-

scribe *the novel use in email of an identity-based signature scheme which prevents forgeability but maintains repudiability*. Taken together, these two concepts form a promising approach to defeating phishing attacks and mitigating spam.

Most importantly, our approach is practical. All existing email infrastructure is preserved, and no heavyweight public-key infrastructure is required. Bootstrapping is natural, and the adoption path for commercial entities is promising.

General-Purpose Lightweight Trust. One should also note that, though our original motivation was the defeat of phishing attacks, *our identity-based public-key infrastructure may be useful for numerous other security functions*. Specifically, our proposal presents a compelling reason and mechanism to deploy public-keys on a wide scale. Numerous existing public-key applications may become far more practical by boostrapping onto Lightweight Trust.

This development is natural: security infrastructure is often deployed out of need, in response to specific threats or to serve a particular commercial purpose. Whereas web-based security was a basic requirement of online commerce as early as the mid-1990s, there has been, until recently, little real need to secure email. Email is, of course, a much more personal communication medium than the Web. Thus, our proposed defense against email-based phishing attacks may well provide a general-purpose mechanism for securing individual, personal traffic on the Internet.

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.

10 Acknowledgements

The authors wish to thank Rob Miller and Min Wu for their helpful pointers to and explanations of user-interface-related solutions to the phishing problem.

References

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

[2] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In Y. Zheng, editor, *Advances in Cryptology — ASIACRYPT '02*, volume 2501 of *LNCS*, pages 415–432. Springer Verlag, 2002.

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

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

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

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In the full version of the paper that will appear in the *proceedings of the 12th Annual Network and Distributed System Security Symposium*, 2005.

[7] B. Adida, S. Hohenberger and R. L. Rivest. Separable Identity-Based Ring Signatures: Theoretical Foundations for Fighting Phishing Attacks. In submission, contact authors for a copy of the paper.

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

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

[10] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In M. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pages 413–430. Springer Verlag, 1999.

[11] Christine E. Drake and Jonathan J. Oliver and Eugene J. Koontz. Anatomy of a Phishing Attack. In *in proceedings of Conference on Email and Anti-Spam 2004*, July 2004.

[12] U. S. Congress. Electronic Signatures in Global and National Commerce Act, Pub. L. No. 106-229, 114 Stat. 464, 2000.

[13] D. Eastlake. RFC 2535: Domain Name System Security Extensions, Mar. 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. C. et. al. Internet X.509 Public Key Infrastructure (latest draft). *IETF Internet Drafts*, Jan. 2005.

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

[17] Federal Trade Commission. How Not to Get Hooked by a 'Phishing' Scam. `http://www.ftc.gov/bcp/conline/pubs/alerts/phishingalrt.htm`.

[18] Gartner. Phishing Attacks Up Against US Consumers, May 2004. `http://www.nwfusion.com/news/2004/0506gartnphish.html`.

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

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

[21] A. Herzberg and A. 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.

[22] F. Hess. Efficient identity based signature schemes on pairings. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography — SAC '02*, volume 2595 of *LNCS*, pages 310–324. Springer Verlag, 2002.

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

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

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

[26] J. Levine and A. DeKok and et al. Lightweight MTA Authentication Protocol (LMAP) Discussion and Comparison, Feb. 2004. http://asrg.kavi.com/apps/group_public/download.php/31/draft-irtf-asrg-%lmap-discussion-01.txt.

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

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

[29] M. Lentczner and M. Wong. Sender Policy Framework: Authorizing Use of Domains in MAIL FROM, 2004. http://www.ozonehouse.com/mark/spf/draft-lentczner-spf-00.txt.

[30] M. Crispin. RFC 1730: Internet Mail Access Protocol - Version 4.

[31] M. Sahami and S. Dumais and 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.

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

[33] 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%.

[34] MessageLabs. Annual Email Security Report, Dec. 2004. http://www.messagelabs.com/intelligence/2004report.

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

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

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

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

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

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

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

[42] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[43] R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT '01*, volume 2248 of *LNCS*, pages 552–565. Springer Verlag, 2001.

[44] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-To-End Arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov. 1984.

[45] M. Scott. MIRACL library. Indigo Software. http://indigo.ie/∼mscott/#download.

[46] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology — CRYPTO '84*, volume 196 of *LNCS*, pages 47–53. Springer Verlag, 1985.

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

[48] T.A. Meyer and B. Whateley. SpamBayes: Effective open-source, Bayesian based, email classification system.

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

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

[51] B. Waters. Efficient Identity-Based Encryption Without Random Oracles, July 2004. Available at http://eprint.iacr.org/2004/180.

[52] F. Zhang and K. Kim. ID-Based Blind Signature and Ring Signature from Pairings. In Y. Zheng, editor, *Advances in Cryptology — ASIACRYPT '02*, volume 2501 of *LNCS*, pages 533–547. Springer Verlag, 2002.

[53] P. Zimmerman. Pretty Good Privacy. http://www.pgp.com.