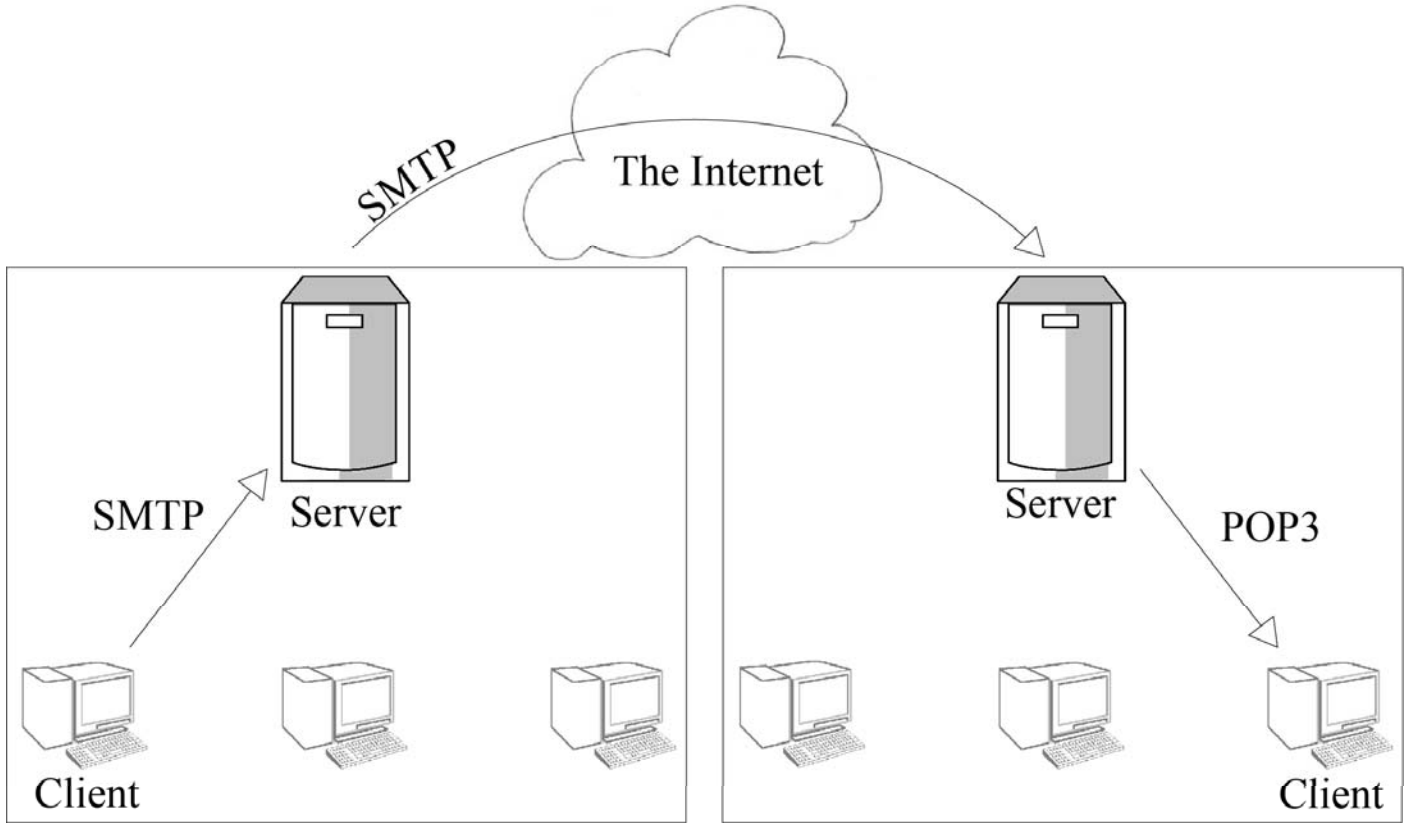
How to Secure Your Email Address Books and Beyond

Erhan J Kartaltepe, Paul Parker, and Shouhuai Xu
Department of Computer Science
University of Texas at San Antonio

Outline

- Email: A Brief Overview
- The Malicious Impostor Email Problem and a Solution Framework
- Encrypted Address Book: This Paper
- Conclusion and Future Work

Email At a High Level



SMTP: Simple Mail Transfer Protocol

- Published and popularized in the early 1980s, SMTP is the *de facto* standard for email transmission across the Internet.
- SMTP is a "push" protocol, allowing a client to send an email to its server, which in turn sends it to the recipient's server.
- SMTP cannot retrieve an email *from* the email server; for that, other "pull" protocols are needed.

SMTP Commands

Command	Description
HELO	Identify the SMTP sender to the SMTP receiver (obsoleted by RFC 2821).
EHLO	Identify the SMTP sender to the SMTP receiver under Extended SMTP.
MAIL	Set the envelope return path and clear the list of envelope recipient addresses.
RCPT	Add one address to the list of envelope recipient addresses.
DATA	Consider the lines following the command to be email from the sender.
RSET	Reset the envelope.
NOOP	Ask the receiver to send a valid reply (but specifies no other action).
QUIT	Ask the receiver to send a valid reply, and then close the transmission channel.

Email Headers

- Each received email contains a ***header*** with the email ID, sender, recipient, time, subject, and path it took from server to server.
- They are easy to create and falsify by the sender or a relay.
- Due to a lack of security safeguards, SMTP headers are a security vulnerability that are simple for adversaries to exploit.

Existing Threats to Email: Less Harmful

- **Spam**
  - unsolicited advertising emails
  - commercially-motivated → mass-produced and distributed
- **Phishing**
  - social engineering attack by impersonating an authority
  - gleans passwords, account numbers, cryptographic keys...

Existing Threats to Email: More Harmful

- **Email Viruses**
  - malevolent program sent by email as an attachment
  - inserts itself into other executables and corrupts files
- **Email Worms**
  - self-replicating, self-propagating program
  - harms the network and consumes bandwidth

Outline

- Email: A Brief Overview
- **The Malicious Impostor Email Problem and a Solution Framework**
- Encrypted Address Book: This Paper
- Conclusion and Future Work

The Malicious Impostor Email Problem

- Email has become an indispensable part of most people's daily routines, but abuses such as spam, worms, and viruses have undermined its utility.

- We envision a next generation attack, much more powerful than previous ones, called **malicious impostor emails**.

What is a Malicious Impostor Email?

A **malicious impostor email**

- looks perfectly legitimate *in every way* (e.g., it can pass any statistical filter and human inspection)
- has a harmful executable as an attachment
- is dangerous because it appears so authentic that the recipient would have no qualms with opening its attachment

Malicious Impostor Email: Formal Definition

Definition: *A malicious impostor email is an email sent to a recipient* U *with mechanisms* WhiteList_U, Filter_U, and Scanner_U *such that*

$\Pr[\text{sender}(\textit{email}) \in \text{WhiteList}_U] = 1$, meaning that the email possesses a sender address that is on U's whitelist.

$\Pr[\text{Filter}_U(\textit{email})$ outputs "suspicious"$] = 0$, meaning that the non-attachment content cannot be detected as malicious, even by a human being.

$\Pr[\text{Scanner}_U(\textit{email})$ outputs "suspicious"$] = \theta$ for some $0 \leq \theta < 1$.

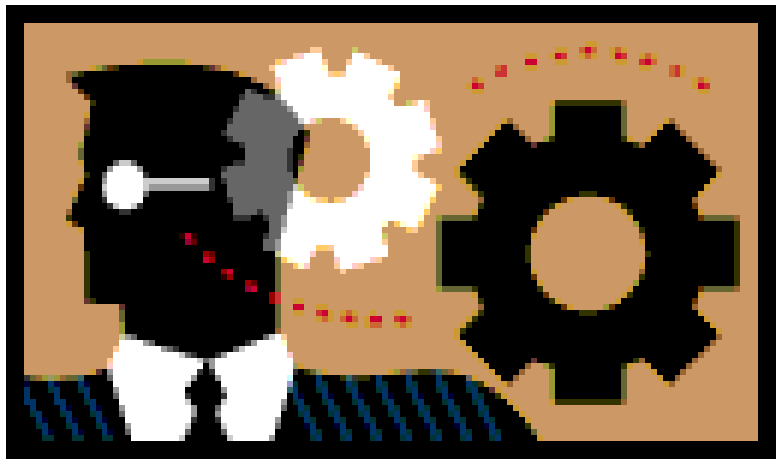
Consequences of Malicious Impostor Emails

Consequences of these attacks are severe.

- The attachment payload could exploit any system vulnerability.
- A PKI could be rendered ineffective.
- They may go unnoticed for long periods of time.

Motivation

How do we deal with malicious impostor emails?



A Solution Framework

- First Line of Defense (this paper): *Prevention*
  - **EAB** can significantly slow down malicious email spreading.
- Second Line of Defense: *Containment*
  - **CAMEL** counters attackers using the local outgoing server.
- Third Line of Defense: *Detection*
  - **MAUDE** counters attackers using their own outgoing server.

Outline

- Email: A Brief Overview
- The Malicious Impostor Email Problem
- **Encrypted Address Book: This Paper**
- Conclusion and Future Work

Key Observation

- Many email viruses and malicious impostor emails proliferate by exploiting the email address books on the infected hosts.
- Thus, encrypting those email addresses may theoretically defeat their self-spreading.
- However, straightforward implementations of the above are either insecure or not deployable.

Building-Blocks

- Building-block I: Embedding passphrases onto pictures
- Building-block II: Encryption scheme and its security requirement (IND-CPA may be too strong; a residing adversary can already compromise the key)
- Building-block III: How to encrypt so as to avoid offline dictionary attack (e.g., @ etc.)

An Encrypted Address Book

- Our mechanism for encrypting email addresses in address books and folders is called **EAB**.
  - each address book entry is encrypted with a unique key.
  - the users are relieved of memorizing any passwords.
  - there is no need for any special purpose hardware.

- **EAB** takes advantage of current hard AI problems such as  and image recognition puzzles.

EAB Under the Hood

- An address book consists of records $A = (A_0, A_1, A_2)$, where A_0 is the address, A_1 is the username, and A_2 are other attributes.
- **EAB** substitutes A_0 with several new attributes, but keeps A_1 and A_2 intact.
- **EAB** uses symmetric key encryption to encrypt each A_0 (RC4 stream cipher in practice) with a unique passphrase as the key.

Securely Encrypting an Email Address

- Email addresses are well-formed and follow a specified format.
- **EAB** encrypts each address, although top-level domains, the first character, and "@" and "`.`" characters are not encrypted.
- **EAB** maps valid email characters (`A-Z`, `a-z`, `0-9`, "`-`" and "_") to the first six bits per character, with the remaining set to "`0`".

An Entry in an Address Book

- An icon is associated with a contact, with the passphrase (RC4's key) embedded in the icon.
- Correctly typing the passphrase decrypts the email address.
- To avoid typos, a second encrypted image may be used, with the passphrase as its key. If the image is consistent with the icon, the passphrase was correctly typed.

Functionalities of Scheme

eab.setup(ces.ab) {
  eab $\leftarrow \emptyset$
  For each $(a_0, a_1, a_2) \in$ ces.ab {
    select a random string r_0
    user picks a passphrase pa
    user picks a picture image
    icon $\leftarrow$ **embed**$(pa,$ image$)$
    $c_0 \leftarrow \mathsf{Enc}_{h(r_0,pa)}(a_0)$
    eab $\leftarrow$ eab $\cup \{($icon$, r_0, c_0, a_1, a_2)\}$
  }
}

eab.addaddr(eab, $addr, user$) {
  select a random string r_0
  user picks a passphrase pa
  user picks picture image
  user enters $info$ for $addr$
  icon $\leftarrow$ **embed**$(pa,$ image$)$
  $c_0 \leftarrow \mathsf{Enc}_{h(r_0,pa)}(addr))$
  $c_1 \leftarrow user$
  $c_2 \leftarrow info$
  eab $\leftarrow$ eab $\cup \{($icon$, r_0, c_0, c_1, c_2)\}$
}

eab.modifyaddr(eab) {
  users clicks an icon $\in$ eab
  user enters passphrase pa from icon
  user enters new address $newaddr$
  $c_0 \leftarrow \mathsf{Enc}_{h(r_0,pa)}(newaddr)$
  update (icon, r_0, c_0, c_1, c_2) in eab
}

eab.getaddr(eab, icon) {
  If (icon, r_0, c_0, c_1, c_2) $\in$ eab {
    user enters pa from icon
    return $\mathsf{Dec}_{h(r_0,pa)}(c_0))$
  }
  Else return NULL
}

eab.deleteaddr(eab, icon) {
  If (icon, r_0, c_0, c_1, c_2) $\in$ eab
    eab $\leftarrow$ eab $- \{($icon$, r_0, c_0, c_1, c_2)\}$
}

eab.geticon(eab, $user$) {
  If (icon, $r_0, c_0, user, c_2$) $\in$ eab
    return icon
  Else return NULL
}

Functionalities of the Prototype System

```
initialize(ces.ab, ces.folders) {
  eab.setup(ces.ab)
  For each folder ∈ ces.folders {
    For each email ∈ folder {
      For each u ∈ email
        u.addr ← eab.geticon(eab, u.name)
    }
  }
  erase ces.ab
}
```

```
deleteaddr(eab, new.folders, icon) {
  For each folder ∈ new.folders {
    For each email ∈ folder {
      If icon = email.addr {
        eab.deleteaddr(eab, icon)
        return
      }
    }
  }
}
```

```
receive(eab) {
  EMAILS ← ces.receive()
  table ← projection(eab, icon, c_1)
  For each em ∈ EMAILS {
    For each u ∈ email {
      If (icon, u.name) ∈ table
        u.addr ← icon
      Else {
        user picks an icon for u.name
        eab.addaddr(eab, u.addr, u.name)
        u.addr ← eab.geticon(eab, u.name)
      }
    }
  }
}
```

```
insertaddr(eab, email) {
  If user clicks "insert address" button {
    user clicks on an icon ∈ eab
    addr ← cab.getaddr(eab, icon)
    ces.insertaddr(addr)
  }
  If user types in an email address addr {
    user enters a username user
    If eab.geticon(eab, user) = NULL
      cab.addaddr(eab, addr, user)
  }
}
```

Example Email Address Book Entry

Original Address Book	Encrypted Address Book
bill@clinton.net Bill Clinton Birth Date: 08-19-1946 Phone number: (212) 555-1248	**b**5410FD@47B17CA3C3280C.**net** FB1BACCAEFB9DF669BC92D160A8E 48A278113EE239CA43DBD719A33B   Bill Clinton Birth Date: 08-19-1946 Phone number: (212) 555-1248

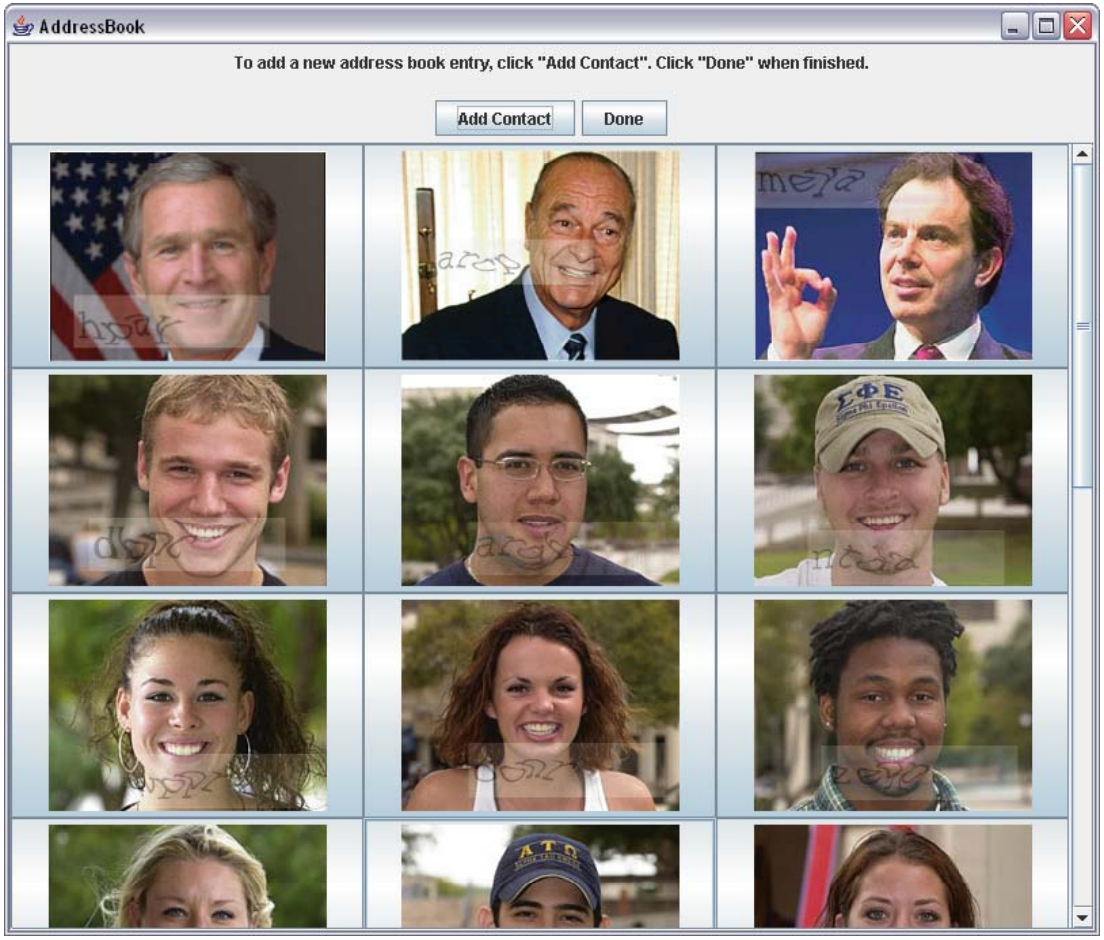
Example Email in the Inbox

Original Email	EAB's Email
From: bill@clinton.net Date: April 25, 2006 Subject: You're awesome Hi Erhan, Thanks for helping me with my taxes. You saved my life! —Bill	 Date: April 25, 2006 Subject: You're awesome Hi Erhan, Thanks for helping me with my taxes. You saved my life! —Bill

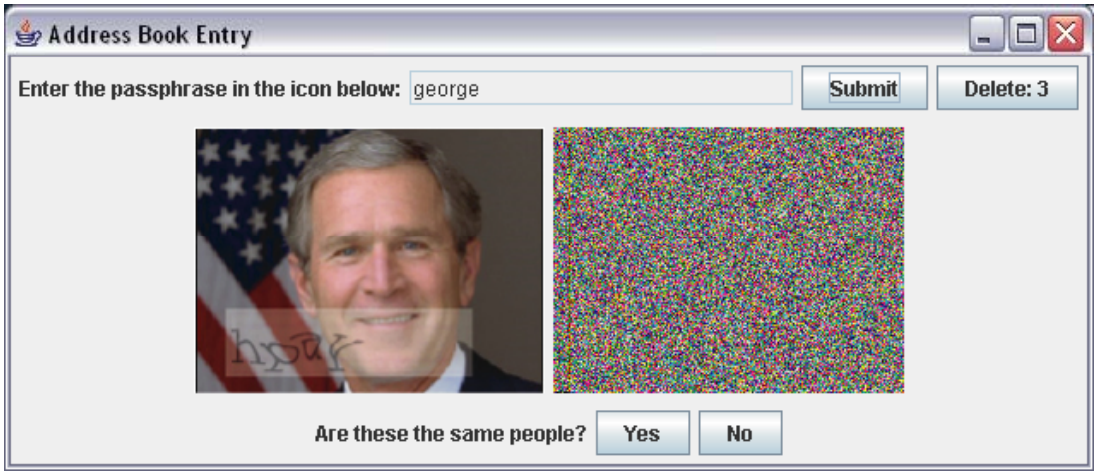
Security Assumptions

- Image recognition is a difficult problem. Specifically, given two images, do they correspond to the same person or not?
- CAPTCHAs are hard for programs to decipher; conversely, CAPTCHAs are easy for a human to decipher.
- Users will type low-entropy passphrases.

Encrypted Address Book at a Glance



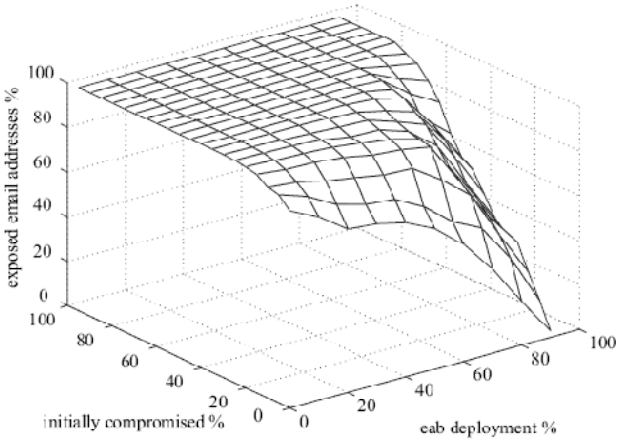
Verifying the CAPTCHA

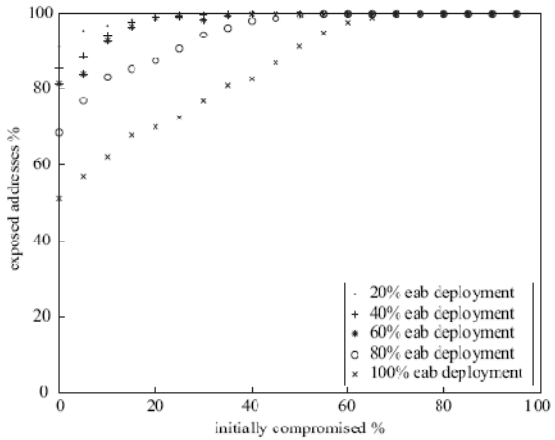


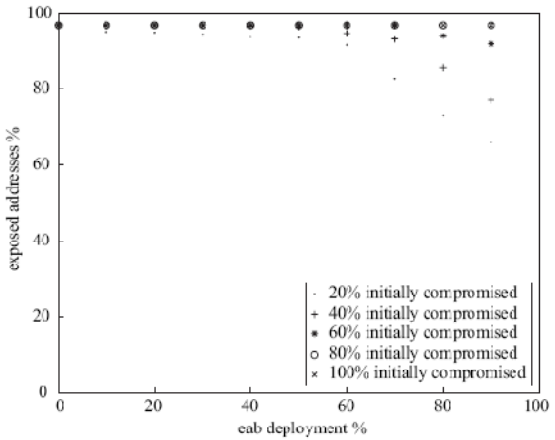
Experiments

- Simulations were run using data from an actual email network (This dataset contained 41,991 anonymized senders and recipients and 406,600 emails.).
- The simulation replayed 16 weeks of emails with varying degrees of initially infected accounts and **EAB** deployments.
- Exposed email addresses were tracked, with the worst case scenario that each user emailed everyone in his address book.

Exposed Accounts After the Experiment







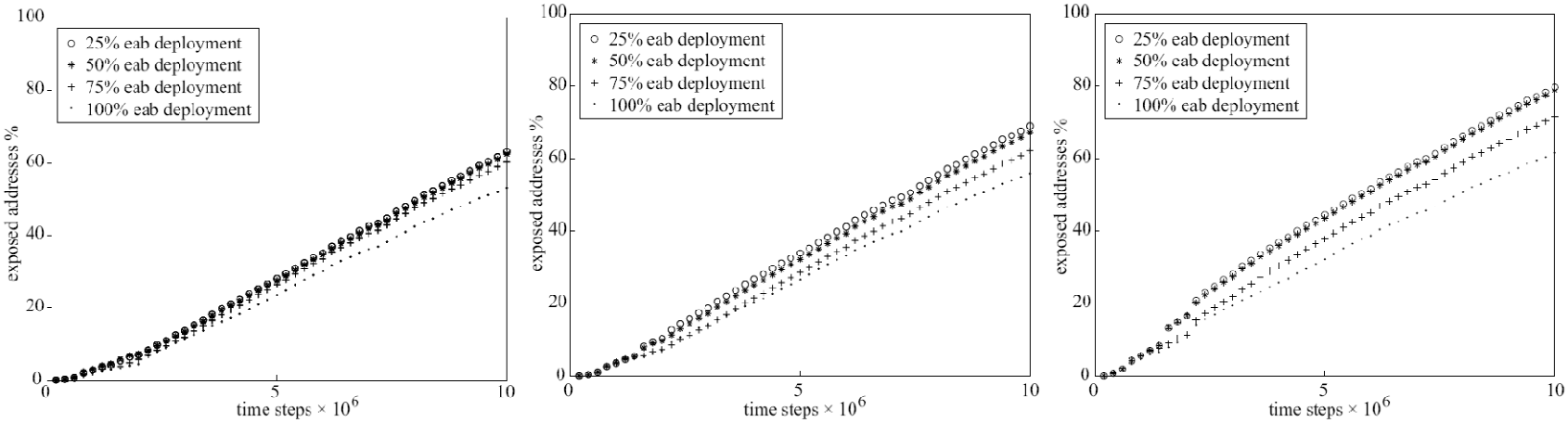
- As initially infected accounts *increase*, exposed addresses *increase*.
- As EAB deployments *increase*, exposed addresses *decrease*.

Initially Compromised Percentage: 1%

- Suppose initially that 1% of the nodes are compromised (which would already be pretty high in real life).

To keep the number of exposed addresses under	The deployment percentage should be
20%	96%
40%	76%
60%	44%

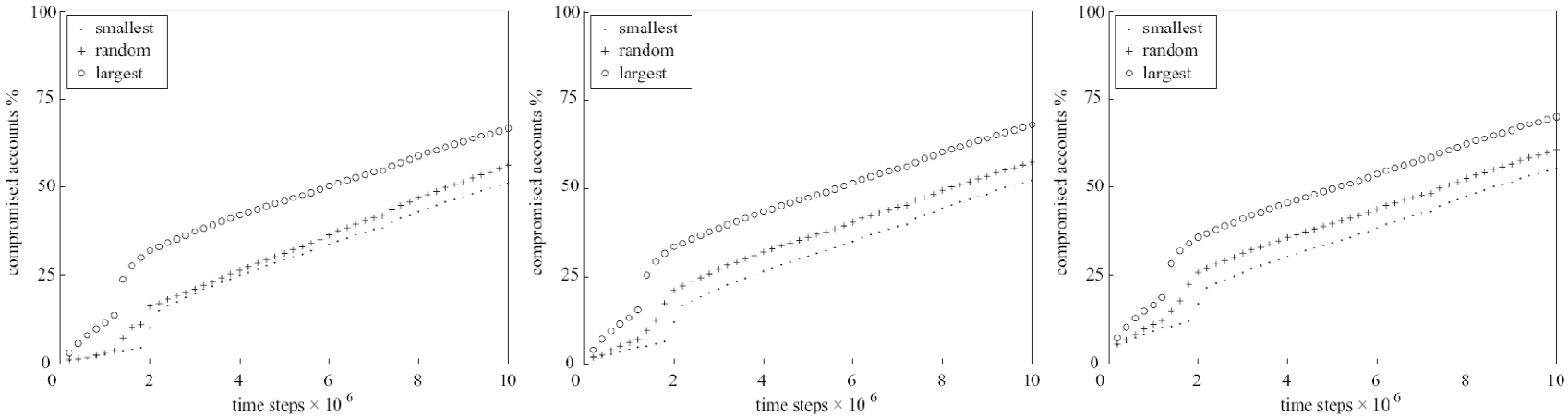
Exposed Accounts Over Time



(a) 1% initially compromised (b) 2% initially compromised (c) 5% initially compromised

- As **EAB** deployment *increases*, exposed addresses *decrease*.
- The larger the initially compromised percentage, the more significant the above tendency is.

Compromised Accounts Over Time



(a) 1% initially compromised (b) 2% initially compromised (c) 5% initially compromised

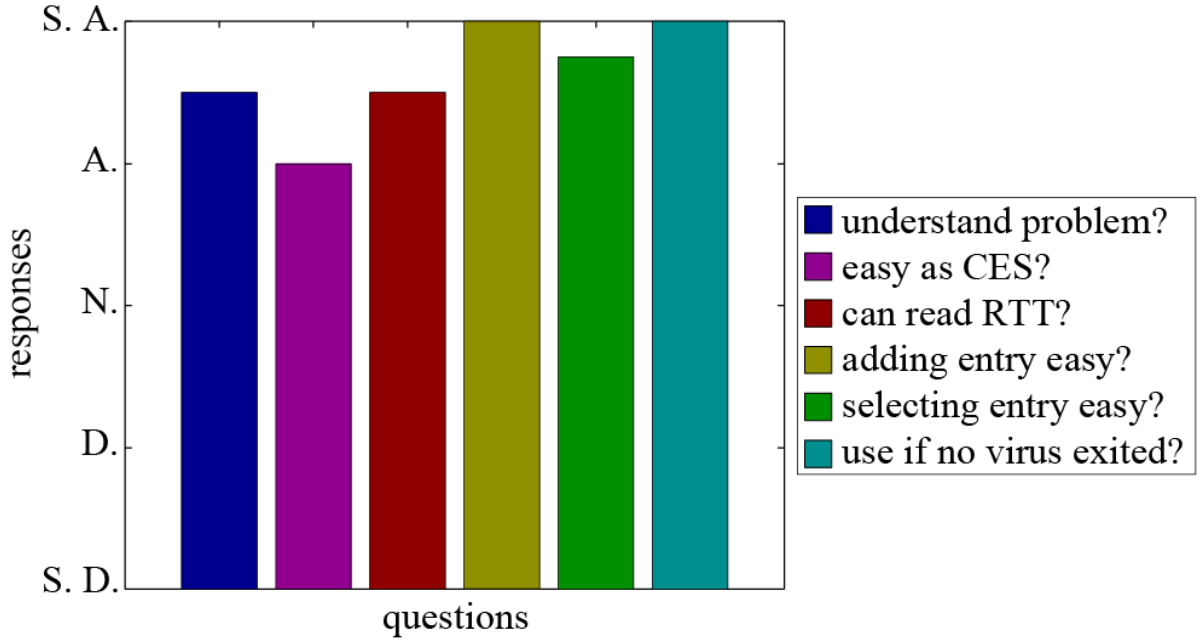
Growth of the number of compromised accounts is affected by

- initial infection
- degree of nodes affected
- **EAB** deployment (not in this experiment)

Usability Testing

- To deploy **EAB** effectively, it must be easy to use.
- Preliminary usability testing with computer scientists and "lay people" were polled to determine if **EAB** was deployable.
- Results were computed using the five-point Likert scale.

Usability Testing Results



- Preliminary results indicate **EAB** is reasonably easy to use.
- Most difficult/time consuming: getting pictures for contact.

Related Prior Works

- Actually some commercial software, which however "encrypts all addresses with a single password"
- But can be integrated with our so that we get "two layers of encryption" (EAB for the email addresses you care most)
- Complementary to other protections such as virus throttling

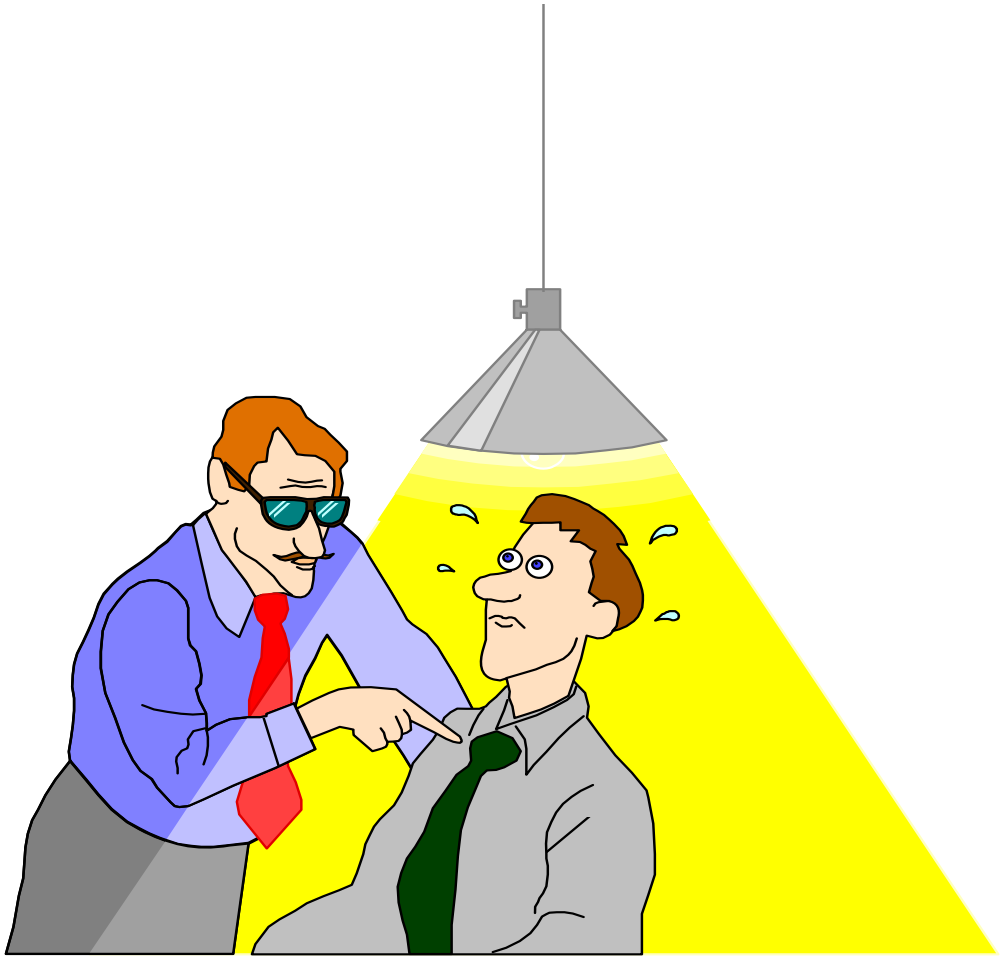
Conclusions

- We systematically explored the feasibility of encrypting email address books so that malicious impostor emails and email viruses cannot automatically spread themselves within a short period of time.
- Beyond: Email folders are also protected

Future Work

- Large-scale experimental study for usability
- Mathematical model for explaining the linear (not exponential) increasing, nor is there phase transition

Questions and Answers



Functionalities of Scheme II

eab.setup(ces.ab) {
  eab $\leftarrow \emptyset$
  For each $(a_0, a_1, a_2) \in$ ces.ab {
    select random strings r_0 and r_1
    user picks a passphrase pa
    user picks pictures image_1 and image_2
    icon $\leftarrow \mathbf{embed}(pa, \text{image}_1)$
    $c_0 \leftarrow \text{Enc}_{h(r_0,pa)}(a_0)$
    $c_3 \leftarrow \text{Enc}_{h(r_1,pa)}(\text{image}_2)$
    eab $\leftarrow$ eab $\cup \{(\text{icon}, r_0, r_1, c_0, a_1, a_2, c_3)\}$
  }
}

eab.addaddr(eab, $addr$, $user$) {
  select random strings r_0 and r_1
  user picks a passphrase pa
  user picks pictures image_1 and image_2
  user enters $info$ for $addr$
  icon $\leftarrow \mathbf{embed}(pa, \text{image}_1)$
  $c_0 \leftarrow \text{Enc}_{h(r_0,pa)}(addr)$
  $c_1 \leftarrow user$
  $c_2 \leftarrow info$
  $c_3 \leftarrow \text{Enc}_{h(r_1,pa)}(\text{image}_2)$
  eab $\leftarrow$ eab $\cup \{(\text{icon}, r_0, r_1, c_0, c_1, c_2, c_3)\}$
}

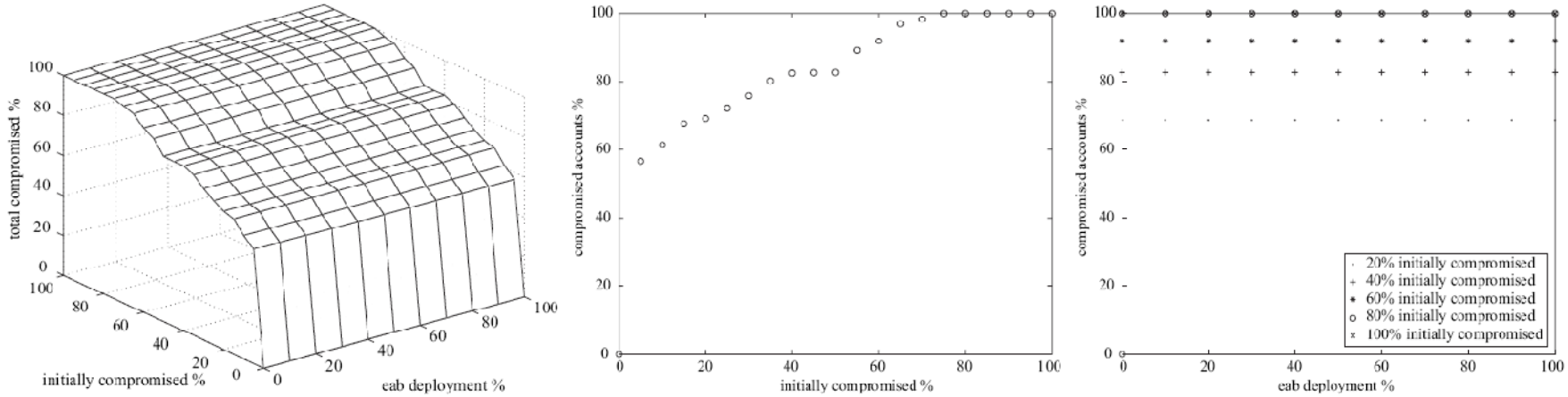
eab.modifyaddr(eab) {
  users clicks an icon $\in$ eab
  user enters passphrase pa from icon
  If icon and $\text{Dec}_{h(r_1,pa)}(c_3)$ concur {
    user types in new address $newaddr$
    $c_0 \leftarrow \text{Enc}_{h(r_0,pa)}(newaddr)$
    update $(\text{icon}, r_0, r_1, c_0, c_1, c_2, c_3)$ in eab
  }
  Else return "wrong passphrase"
}

eab.getaddr(eab, icon) {
  If $(\text{icon}, r_0, r_1, c_0, c_1, c_2, c_3) \in$ eab {
    user enters passphrase pa from icon
    If icon and $\text{Dec}_{h(r_1,pa)}(c_3)$ concur {
      return $\text{Dec}_{h(r_0,pa)}(c_0)$
    }
    Else return "wrong passphrase"
  }
  Else return NULL
}

eab.deleteaddr(eab, icon) {
  If $(\text{icon}, r_0, r_1, c_0, c_1, c_2, c_3) \in$ eab
    eab $\leftarrow$ eab $- \{(\text{icon}, r_0, r_1, c_0, c_1, c_2, c_3)\}$
}

eab.geticon(eab, $user$) {
  If $(\text{icon}, r_0, r_1, c_0, user, c_2, c_3) \in$ eab
    return icon
  Else return NULL
}

Compromised Accounts



- **EAB** was not deployed and this experiment reflects this.
- If **EAB** were deployed, no malicious email would leave the client.