Applied Cryptography and Network Security

Adam J. Lee

adamlee@cs.pitt.edu

6111 Sennott Square

Lecture #25: Email Privacy

April 10, 2014

University of Pittsburgh

Outline

Overview of the email system

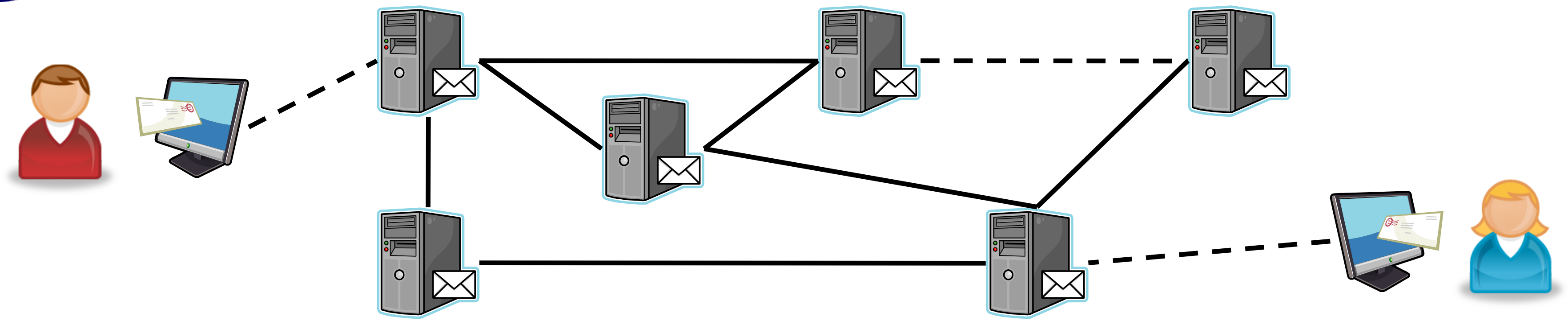
Group discussion: Desirable properties for secure email

Attaining these properties

- Basic properties
- Confidentiality properties
- Delivery properties

Case study: PGP/GPG

Electronic mail is forwarded through a mesh of servers



User agents (UAs) are used to compose, send, retrieve, and view mail

- ***Examples:*** Thunderbird, Pine, Mail, etc.
- UAs are not always connected to the email network

Mail transfer agents (MTAs) are used to route e-mail between users

- ***Example:*** The `sendmail` program
- MTAs are typically always online, but historically, they need not be

Question: Why might we have paths involving multiple MTAs? Why not just send email point to point?

Like most older Internet protocols, the email system was not built with security in mind

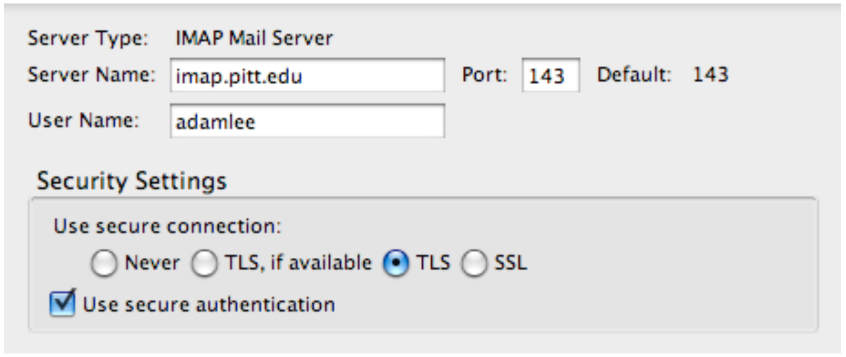
The Simple Mail Transfer Protocol (SMTP) is used to send mail from a UA to an MTA and to relay mail between MTAs

UAs typically use either the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP) to pull messages from their MTA

These protocols are very basic:

- Commands are sent as ASCII text strings
- Messages must be ASCII text
  - Non-ASCII text needs to be encoded prior to transmission
  - See MIME standards
- Authentication (if in place) is username/password authentication

To provide some sense of security, these protocols are often run over SSL/TLS



Server Type: IMAP Mail Server
Server Name: imap.pitt.edu Port: 143 Default: 143
User Name: adamlee

Security Settings
Use secure connection:
Never TLS, if available [✓ TLS] SSL
[✓ Use secure authentication]

SMTP Example

```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.org>
C: To: Alice Example <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 Jan 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice. This is a test.
C: Your friend, Bob
C: .
S: 250
Ok: queued as 12345
C: QUIT
S: 221 Bye {The server closes the connection}
```

Message is from Bob

Sent to Alice and the boss

Message is just plain text, terminated with a return, a period, and a return

Question: What is to stop users from pretending to be other users?

Discussion

What are some desirable properties that we might want an electronic mail system to have? Work in groups to identify properties, their definitions, and possible enforcement mechanisms.

Properties I (Basic)

Authentication: Is the sender who they claim to be?

- Might want Sender/MTA authentication to ensure appropriate use of an organization's mail facilities
- User to user authentication is nice if email has real world implications

Integrity: Users should be convinced that they receive the same message that was originally sent

Non-repudiation: The recipient should be able to prove to a third party that the sender really did send the message

- Why is this useful?
  - The email in question might authorize some sort of atypical action
  - The message might contain a contract or purchase order
  - …

Properties II (Data hiding)

Message privacy: The contents of a message should only be readable by the intended recipient

- Email is more like a postcard than a sealed letter
- Message privacy provides senders/receivers with the envelope that is otherwise missing from this analogy

Message flow confidentiality: The fact that two parties are communicating with one another should remain confidential

- Recall earlier discussions about traffic analysis
  - ➢ Lots of communication between the Pentagon and late-night delivery places implies that a military operation may be starting up
- The ability to protect message flow confidentiality hides potentially sensitive behaviors from prying eyes

Anonymity: The recipient of a message should have no way to determine who sent the message

When might this be a useful property?

Properties III (Delivery)

Proof of submission: The sender should be able to prove to a third party that a particular message was sent

- This is like certified mail in the physical world
- Useful for proving that you took required actions in a timely manner
  - i.e., It's not my fault! I sent the message and the delivery system failed!

Proof of delivery: The sender should be able to obtain proof that a message was delivered to its intended recipient

- Analogous to delivery confirmation at the post office
- Useful when messages need to be tracked

Sequencing: The system should provide assurance that messages arrive in their intended order

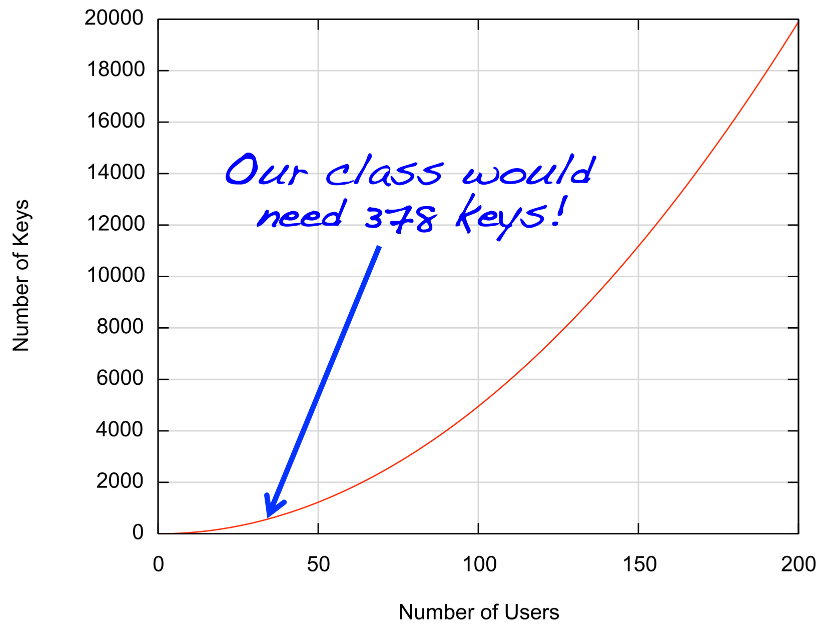
Why should we care about message sequencing?

How can we attain these properties?

By using cryptography, of course!

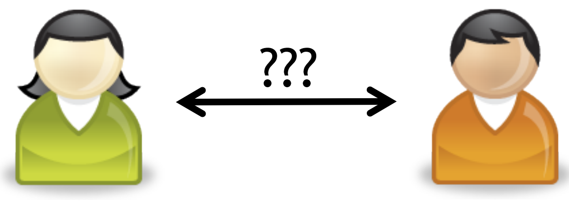
Problem 1: Key management

- In a network with *n* participants, $\binom{n}{2}$ = n(n-1)/2 keys are needed!
- This number grows very rapidly!



Problem 2: Key distribution

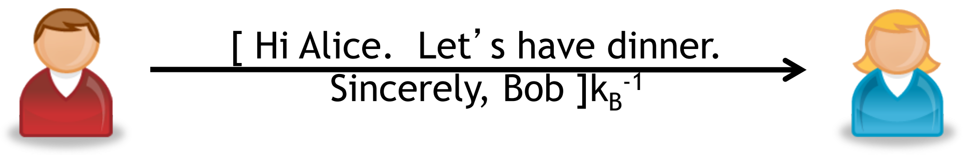
- How do Alice and Bob share keys in the first place?



- What if Alice and Bob have never met in person?
- What happens if they suspect that their shared key K_{AB} has been compromised?

Most secure email systems are based on public key cryptography, but use hybrid cryptosystems for efficiency

Digital signatures can provide us with many of our basic properties



Authentication

- The message body says that the letter is from Bob
- Alice can verify that Bob signed the message by using k_B
- If the signature checks out, only Bob could have produced it
- Note: Digital signatures can also provide UA → MTA authentication (How?)

Integrity

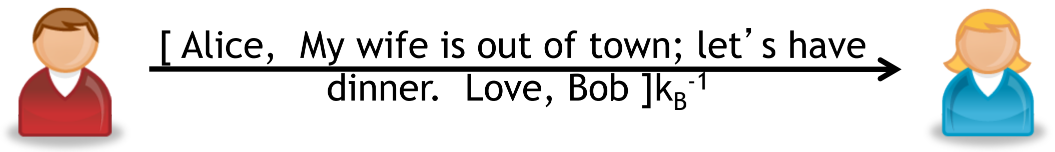
- If the message is modified in transit, the signature will not check out!
- This protects against transient failures and malicious modifications

Non-repudiation

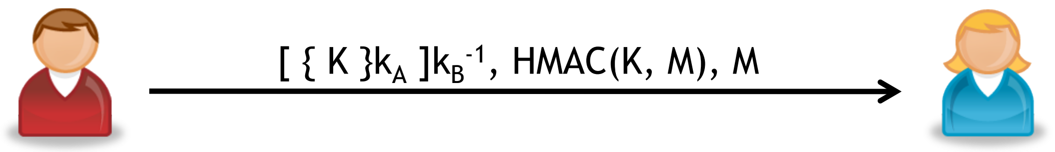
- All that is needed to verify Bob's signature is his public key
- Since the public key is public knowledge, anyone can verify Bob's signature

Interesting twist: Plausible deniability using public key cryptography

Bob may not always want message non-repudiation…


[Alice, My wife is out of town; let's have dinner. Love, Bob]k_B^{-1}

Hybrid cryptography gives us a means of achieving both authentication and deniability:



Question: Why does this authenticate Bob?

- Alice knows that she received the message from Bob
- The digital signature verifies this fact

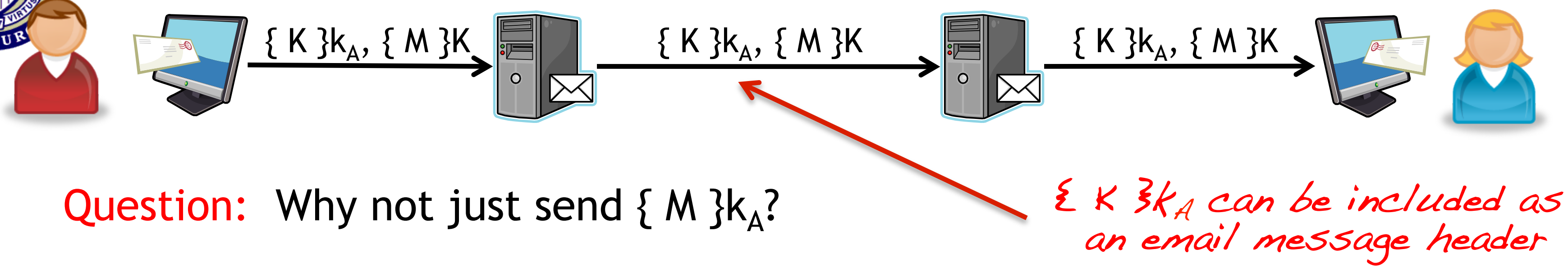
Question: Why does this protocol provide message integrity?

- The key K is only known to Alice and Bob

Question: Why does this protocol provide deniability?

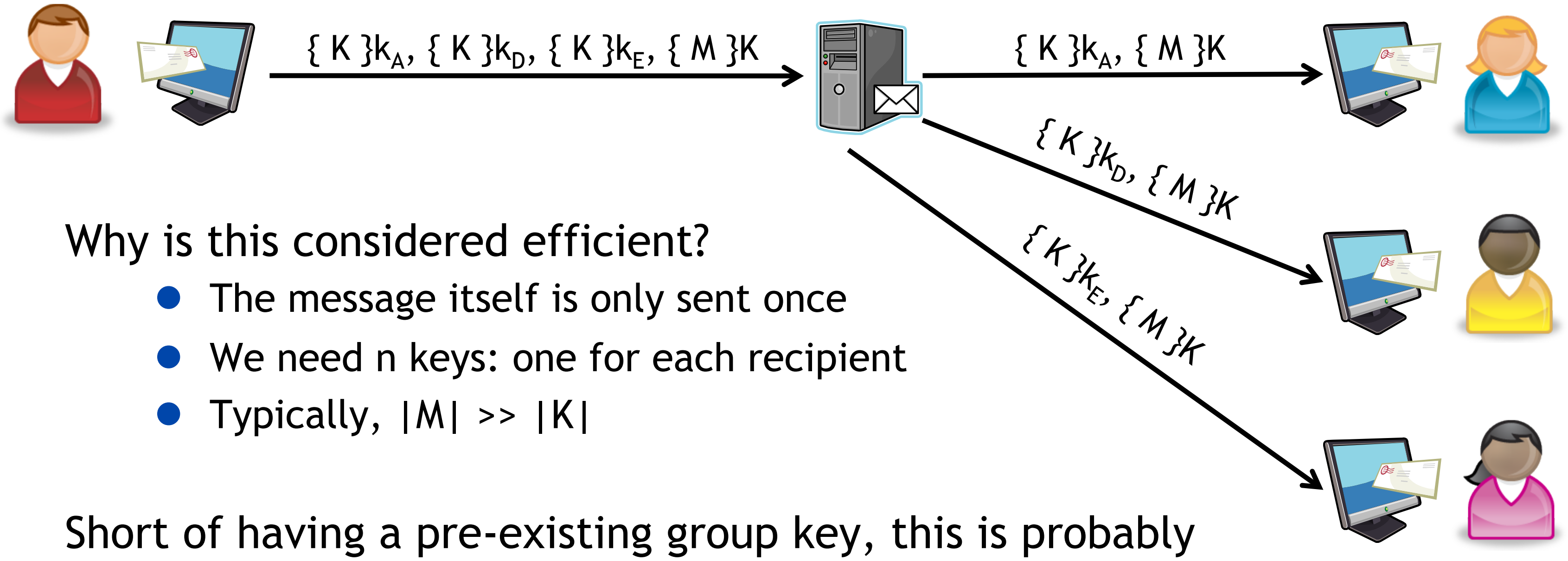
- After Alice recovers the key K, she can compute the HMAC for any message

Given what we've seen so far, achieving message-level privacy is actually pretty easy



Question: Why not just send { M }k$_A$?

This also allows us to efficiently send mail to multiple recipients!

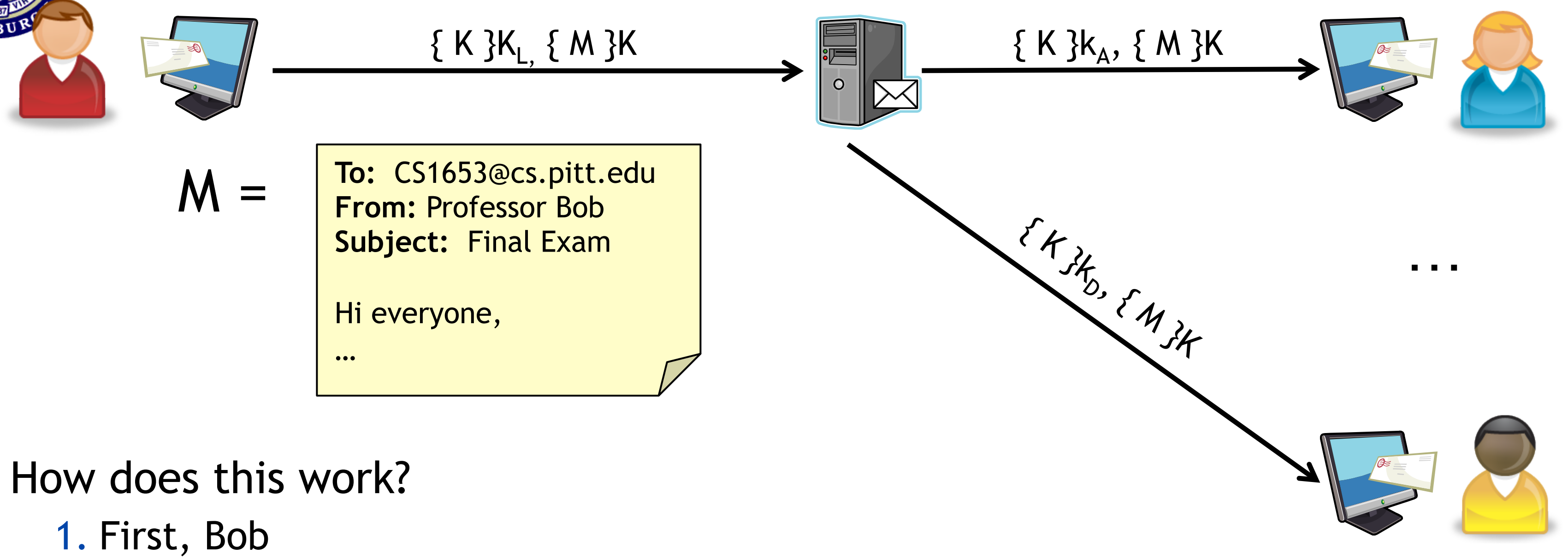


Why is this considered efficient?

- The message itself is only sent once
- We need n keys: one for each recipient
- Typically, |M| >> |K|

Short of having a pre-existing group key, this is probably as good as we could expect to do

This can even be extended to work with mailing lists!



To: CS1653@cs.pitt.edu
From: Professor Bob
Subject: Final Exam

Hi everyone,
…

How does this work?

1. First, Bob
   - Generates a random symmetric key K
   - Computes $\{ M \}K$ and $\{ K \}k_L$
   - Sends these values to the list server

2. Then, the list server
   - Decrypts $\{ K \}k_L$ to recover K
   - Explodes the list "CS1653@cs.pitt.edu"
   - Encrypts the key K to each member
   - Transmits $\{ K \}k_i$ $\{ M \}K$ to each principal p_i

3. Finally, each principal p_i
   - Decrypts the $\{ K \}k_i$ to recover K
   - Decrypts $\{ M \}K$

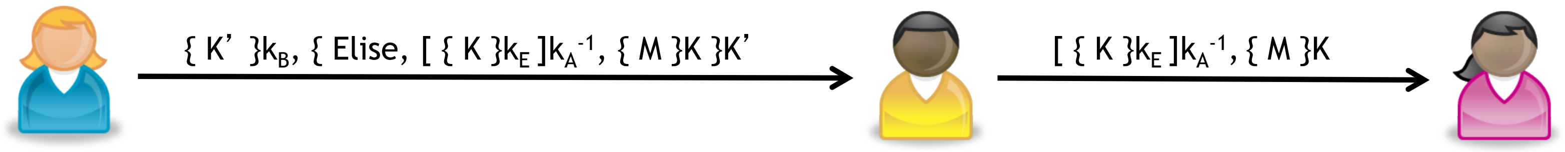
Discussion

Work in groups to develop an extension of the previous "confidential mailing list" setup that (i) uses symmetric keys instead of public/private key pairs and (ii) provides both privacy and integrity.

Message flow confidentiality and anonymity are a little bit harder to attain…

Note: All MTAs along the delivery path for a message know both the sender and the recipient of that message

One way to hide message flows is to use a trusted intermediary


$\{ K' \}k_B, \{ \text{Elise}, [\{ K \}k_E]k_A^{-1}, \{ M \}K \}K'$
$[\{ K \}k_E]k_A^{-1}, \{ M \}K$

Why does this work?

- Correctness: Bob is able to decrypt Alice's original message and forward the message that it contains
- Confidentiality: Bob cannot recover the key K, since he does not know k_E^{-1}
- Integrity: Elise can verify the signature on K
- Flow confidentiality: Only Bob knows that Alice and Elise are talking

Question: Why might Alice want to use multiple levels of intermediary

- No single intermediary knows the whole path
- Messages can be batched to hide flows from global monitors
- This is how Mixminion/Mixmaster work

Discussion

How could trusted intermediaries be used to facilitate anonymous email?

Case study: PGP

Pretty Good Privacy (PGP) is a hybrid encryption program that was first released by Phil Zimmerman in 1991

In the PGP model

- Users are typically identified by their email address
- Users create and manage their own digital certificates
- Certificates can be posted and discovered by using volunteer “key servers”
- Key servers also serve a function similar to an OLRS

Email addresses are GUIDs, so they effectively disambiguate identities

However, note that there are no CAs in the system! As such, users can create certificates for any email address or identity that they want!

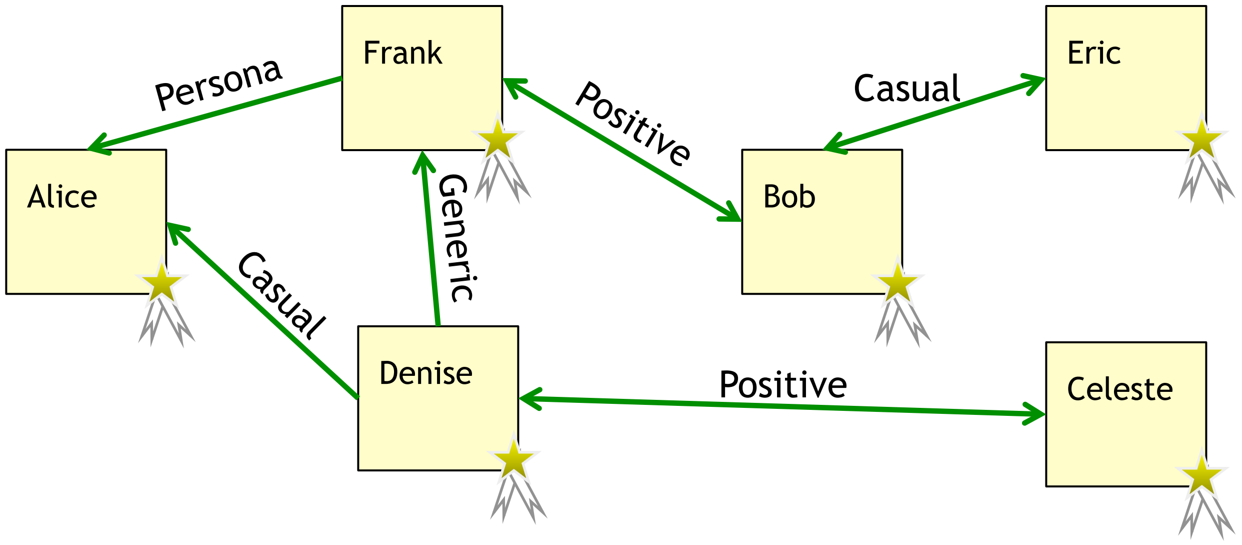
How can we deal with this anarchy?!

PGP takes a grassroots approach to certificate validation

Users create their own certificates, and sign the certificates of others

- "I am Alice and I have verified that this certificate belongs to Bob"
- Four levels of certification: Generic, Persona, Casual, and Positive

This is essentially a cryptographic social network!



Question: Can Celeste trust Frank? Can Eric trust Frank?

GPG is an implementation of the OpenPGP standard

GPG stands for the Gnu Privacy Guard

- The OpenPGP standard is defined in RFC 4880
- As such, GPG interoperates with current versions of PGP

GPG supports a number of unencumbered cryptographic algorithms

- Symmetric key ciphers: CAST5, Triple DES, AES, Blowfish, and Twofish
- Asymmetric key ciphers: RSA and ElGamal
- Hash functions: MD5 (ack!), SHA-1, RIPEMD-160, and Tiger
- Digital signatures: DSA

The core GPG program is command-line driven, and is available for a number of popular operating systems

Graphical front ends for GPG do exist

How does GPG work?

Step 1: Adding a signature

- Compute the hash, H(M), of the message
- Append the name of the hash function and a signature over H(M) to the message

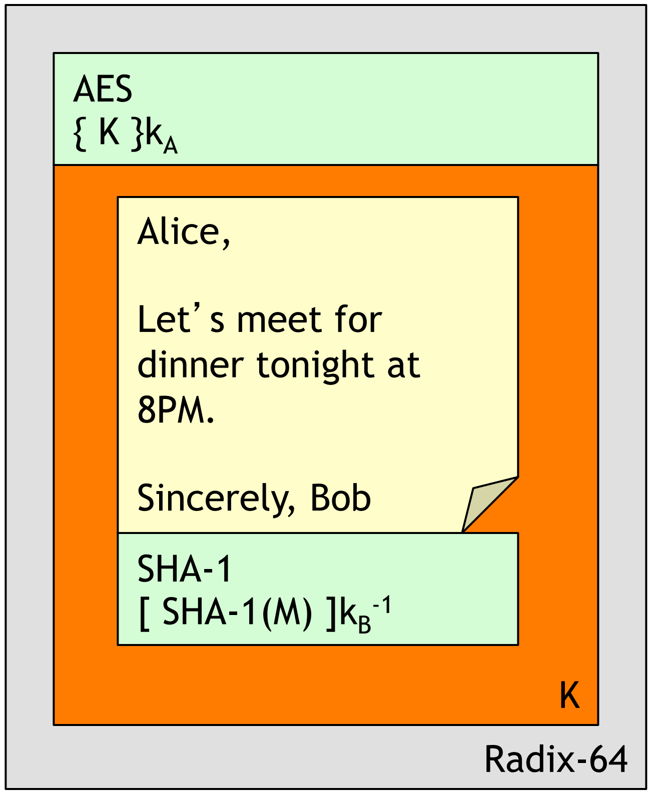
Step 2: Compress the message

Step 3: Encryption

- Generate a random key K
- Encrypt K and its corresponding algorithm name with the recipients public key
- Encrypt the whole message with K
- Prepend the encrypted key block to the newly encrypted message

Step 4: Encoding and Transmission

- Radix-64 encode the message (Why?)
- Send via email


AES
$\{ K \}k_A$
Alice,
Let's meet for dinner tonight at 8PM.
Sincerely, Bob
SHA-1
$[\text{SHA-1}(M)]k_B^{-1}$
K
Radix-64

Receiving a Message

Step 1: Decode the message

- Undo Radix-64 encoding to recover bytes

Step 2: Key recovery

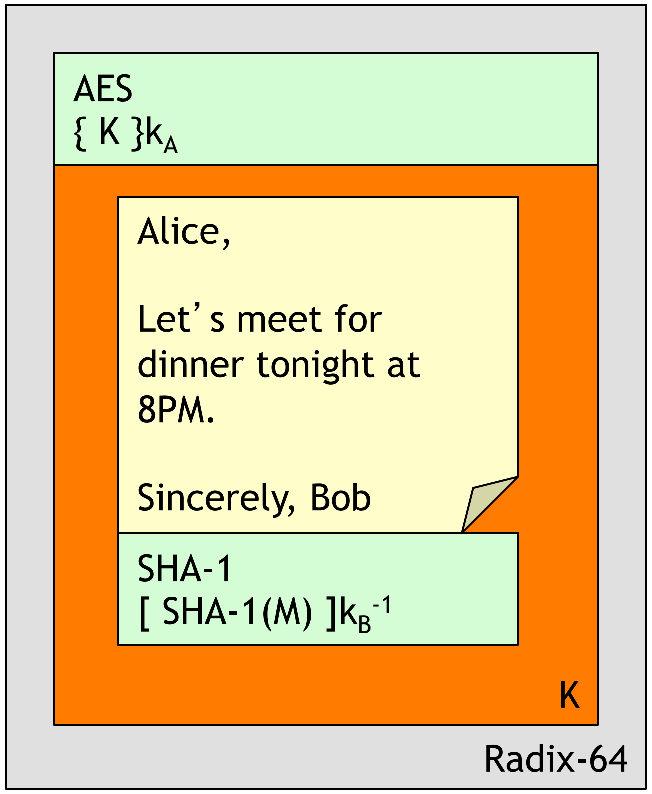
- Extract symmetric key algorithm name
- Recover session key K

Step 3: Decrypt and decompress

- Decrypt the message block
- Decompress message block

Step 4: Verification

- Extract hash function name
- Compute H(M) and validate sender signature



AES
$\{ K \}k_A$
Alice,
Let's meet for dinner tonight at 8PM.
Sincerely, Bob
SHA-1
$[\text{SHA-1}(M)]k_B^{-1}$
K
Radix-64

Conclusions

The email system was designed in more trustworthy times

There are many properties beyond best effort delivery that we would like to be provided by the email infrastructure

- End to end confidentiality
- Message path confidentiality
- Integrity
- Non-repudiation
- …

Many of these goals can be attained using cryptographic means

GPG is an open-source implementation of the OpenPGP standard

- One might claim that it offers pretty good privacy…

Next time: Data Privacy