Homework #5 Solutions

Brian A. LaMacchia
bal@cs.washington.edu
bal@microsoft.com

Question 1 - Timestamping

- Clients send the timestamping service a hash value.
- The service signs the hash value together with the current time, producing a *timestamping receipt*.
- The timestamping receipt is then sent back to the client, who can do whatever he wants with it (typically, archive it and/or send it along with the signature).

Question 1a

- What information would you include in the receipt?
- Why are you including it?
- What's the minimum size in bytes of the information you have to include?
- Assumptions:
  - Hash values sent to you by clients are all SHA2-256 hashes and are thus 32 bytes in size.
  - Time is expressed using the GeneralizedTime structure, which is 15 bytes in length.

Question 1a

- Minimum set:
  - The hash value the client submitted to the timestamp service (32 bytes)
  - The timestamping time (15 bytes)
  - An identifier for the signature algorithm the timestamp service is going to use to sign the receipt
    - This includes the algorithm the service uses to hash the receipt contents (e.g. "RSA-SHA2-256")
    - Variable length, could be as small as 1 byte, but is probably at least 4 bytes. Anything reasonable is OK so long as you justified it.
- The timestamp server's signature will take another 256 bytes (assuming RSA with 2048-bit keys)
  - Per my e-mail, since by "included" I had intended "included in the to-be-signed part", you didn't have to include the signature (it's OK if you did)

Question 1a

- What additional information would you include? Some possibilities:
  - Version numbers
    - E.g. For the receipt format (~4 bytes typically)
  - Public key of the timestamp authority
    - Assume ~260 bytes (256 bytes for a 2048-bit public key, 4 bytes for e if it's short)
  - Certificates for the timestamp authority
    - These would be at least 512 bytes each (subject public key + issuer signature), probably 1K or more in practice.

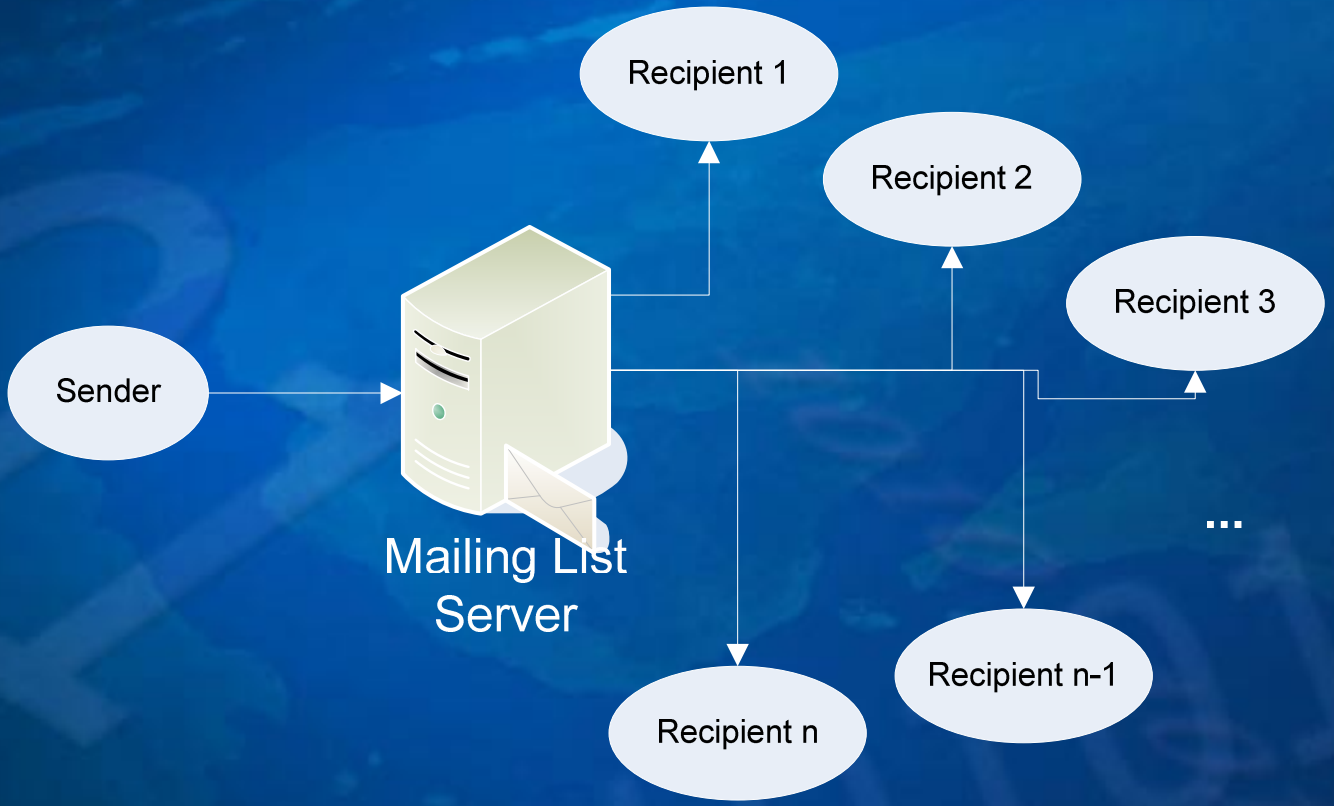
Question 1b

- Describe how you could modify the operation of your timestamping service to defend against fraudulent insertion of timestamps "after the fact".
- What additional information do you have to add to the timestamping receipt to effect this change?

Question 1b

- One way to do this is to link the timestamping receipts together by “hash chaining”
  - Include the hash of the nth receipt within the to-be-signed info of the (n+1)st receipt.
  - Every so often (daily, weekly) widely publish (e.g. NYT classifieds) the hash of the last issued timestamp.
  - Only adds 32 bytes to the receipt (size of a SHA2-256 hash)
- "How to Timestamp a Digital Document," S. Haber, W. Stornetta, Journal of Cryptology, Vol. 3, No. 2, pp. 99-111, 1991. Available at http://www.surety.com/dataintegrity.php

Question 2 – Encrypting Mail to Mailing Lists



Question 2a

- What information does the sender have to know about each mailing list recipient if it wants to be able to send them encrypted messages?
  - Nothing—in this scenario the sender doesn't need to know anything about the ultimate recipients of mailing list messages.
  - The sender just has to know the public encryption key of the central mailing list server.

Question 2a

- What information does each recipient have to know about the mailing list server?
  - Nothing in order to decrypt messages sent to them.
  - If the server signs messages in any of your protocols, then recipients would have to know the public part of the server's signing key pair.

Question 2a

- If there are M members of the mailing list, how many public keys does each member need to know and how many keys does the server need to know?
  - Each member needs to know 1 public encryption key (the public key of the server).
  - They also need to know their own public & private key.
  - The central server needs to know M public encryption keys (one per subscriber), plus his own private key pair

Question 2b

- When the sender encrypts a message that he wants to send to the mailing list, how many RecipientInfos will his S/MIME message have?
  - One (for the mailing list server)
- Does it make a difference if the sender wants to archive a copy of his encrypted message in his "Sent Items" folder in case he wants to look at it later?
  - Yes, in this case the sender needs to include a second RecipientInfo for himself.

Question 2c

- Describe one way that the server could verify that the message came from a mailing list subscriber.
  - The server could maintain a database of members public signing keys along with their e-mail addresses and require that senders digitally sign all messages sent to the mailing list.

Question 2c

- Does the server need to know any additional information about the sender beyond what you already indicated in you answer to Question 2(a)?
  - Yes, in the general case. The server would need to know a public signature key for each mailing list participant. While this could conceivably be the same as the encryption key if RSA is being used, in general subscribers would have separate signing and encryption keys.

Question 2d

- If the mailing list has M members, how many public key encryptions does the server have to perform to prepare the message for sending?
  - The server needs to perform at least M-1 public key encryptions – one for every recipient except the sender (assuming the sender included a second RecipientInfo for himself). If the sender didn't do that then it's M encryptions.
  - For each recipient the server needs to create a RecipientInfo (content encryption key encrypted with the recipient's public key).

Question 2d

- Does it make a difference if the server prepares one message with many RecipientInfos vs. a separate message for every recipient?
  - No, not in terms of the number of public key encryptions required. The server has to perform the same number of PK encryptions whether it's one message to M recipients or M separate messages.

Question 2e

- How much symmetric key decryption and encryption does the server need to do in order to properly relay the message?
  - This depends on your answer to 2(c). If you said in 2(c) "have the signer sign the message", then the server is going to have to decrypt the message once in order to reveal and verify the signer's signature.

Question 2e

- Re-encryption of the message body:
  - If the mailing list is not anonymous, then the server can re-use the encrypted content from the inbound message, and no further symmetric operations are required.
  - If the mailing list *is* anonymous, then the server will have to re-encrypt the message body after removing the signature, so one symmetric encryption of the content will be required.

Question 3 – Subscribe Protocol

- Describe a protocol that the new subscribing user and the server can use to send this information to the server and authenticate that it came from the entity that receives e-mail at the subscribing address.
  - That is, the server needs to know that the subscriber isn't maliciously signing someone else up to the mailing list.

Question 3 – Solution

1. Subscribing client C randomly generates a new public/private encryption key pair (K_{Pub}, K_{Priv})
2. C → Server S: K_{Pub} along with his subscribing e-mail address
3. S randomly generates a nonce N
4. S → C the nonce encrypted with the public key K_{Pub}: $\{N\}K_{Pub}$

Question 3 – Solution

5. C decrypts $\{N\}K_{Pub}$ to obtain N.
6. C encrypts the hash H(N) to S using S's public key K_{SPub}
7. C → S: $\{H(N)\}K_{SPub}$
8. S decrypts $\{H(N)\}K_{SPub}$ to obtain H(N) and verifies that H(N) is the hash of the nonce originally sent to C.

Question 4 – Group Symmetric Key

- Design the "group symmetric key" system. Describe how:
  1. The server can ensure that new subscribers get the current group symmetric key
  2. The group symmetric key is used to encrypt a particular message
  3. What happens to the current group symmetric key when a member unsubscribes from the mailing list.

Question 4 – Group Symmetric Key

1. Describe how the server can ensure that new subscribers get the current group symmetric key
   - The server can communicate the GSK as part of the registration process. We can modify the solution to Q3 above as follows

Question 4

[Steps 1-6 same as in Q3]

7. C → S: $\{H(N)\}K_{SPub}$
8. S decrypts $\{H(N)\}K_{SPub}$ to obtain H(N) and verifies that H(N) is the hash of the nonce originally sent to C.
9. S → C: $\{GSK\}K_{Pub}$
10. If we want confirmation,
    C → S: H(GSK,N), or
    C → S: $\{H(GSK)\}K_{SPub}$

Question 4 – Group Symmetric Key

2. **Describe how the group symmetric key is used to encrypt a particular message.**
   - When the server receives an inbound message, it will decrypt the RecipientInfo addressed to it to obtain the content encryption key (CEK).
   - Now, instead of performing a public key encryption for each recipient, the server encrypts the content encryption key with the group symmetric key. The result is stored in a RecipientInfo that's common to all recipients.
   - We use the "KEKRecipientInfo" form of RecipientInfo in S/MIME, which allows us to encrypt a content encryption key with another symmetric key.

Question 4 – Group Symmetric Key

3. **What happens to the current group symmetric key when a member unsubscribes from the mailing list.**
   - **When a user unsubscribes, the current GSK becomes invalid and the server has to generate a new GSK and send it to the remaining list members.**
   - **One way to do this is to have the server send a separate message to each list member containing the new GSK encrypted with their respective public encryption keys.**
   - **Clients could then send a confirmatory message to the server to indicate that they've received the new GSK. [Optional step]**

Question 4 – Group Symmetric Key

- Another possibility is to have the server send the GSK (encrypted to each recipient) as part of the next mailing list message.
  - Form the S/MIME message using the GSK to encrypt the CEK.
  - Include the KEKRecipientInfo with {CEK}GSK (CEK encrypted with GSK)
  - Include M additional RecipientInfos, one per member i, containing $\{GSK\}_{KPub,i}$