Massachusetts Institute of Technology
6.857: Network and Computer Security
Professor Ronald L. Rivest

Handout 13
September 26, 2002

Problem Set 1 Solutions

Problem 1-1. PGP

(a) PGP baby steps

The majority of students successfully mailed the staff using PGP. As far as we know, the only legitimate staff keys are for Kevin Fu and Thien-Loc Nyguen. Kevin has two keys; Thien-Loc has three.

Most people used the following public key because it was older and had many signatures in the "Web of Trust." A few students recognized signatures from their peers. One person doubted the key's authenticity because of the strange 2047-bit key. Believe it or not, this is an artifact of a bug in an old version of PGP that had an off-by-one error. One student also commented that the creation timestamp leads to the question of how long someone has been hanging around MIT....

```
Type Bits KeyID      Created    Expires    Algorithm       Use
sec+ 2047 0xE0FD2589 1995-02-25 ---------- RSA             Sign & Encrypt
f16    Fingerprint16 = E1 83 68 C5 1D 46 17 17  9C D5 E9 AB D3 3B 1E B5
uid  Kevin E. Fu <fubob@mit.edu>
```

Another valid key is the following, created when DSS keys became in vogue. It, however, has fewer signatures.

```
sec+ 1024 0xD6354960 2001-08-12 ---------- DSS             Sign & Encrypt
f20    Fingerprint20 = 257D 2532 B896 BCBB E4A9  418F E992 5479 D635 4960
sub  2048 0x2C72A0B9 2001-08-12 ---------- Diffie-Hellman
f20    Fingerprint20 = 57A0 4F2B C82D A0D3 4AA6  36DE 2A6A 3652 2C72 A0B9
uid  Kevin Fu <fubob@mit.edu>
```

Thien-Loc created three PGP key pairs shortly before releasing the problem set. None of the keys had signatures (except for self-signatures). For this reason, many people did not trust the keys.

```
Type Bits KeyID      Created    Expires    Algorithm       Use
pub  1024 0x3DE13338 2002-09-09 ---------- DSS             Sign & Encrypt
f20    Fingerprint20 = C001 78BD 6805 E1EA DB85  EAD2 9246 3D1B 3DE1 3338
sub  2048 0x010D7B51 2002-09-09 ---------- Diffie-Hellman
f20    Fingerprint20 = 63B0 34BF 10EF 41FD F302  739E 8B87 9C12 010D 7B51
uid  Thien-Loc Nguyen <nguyentl@mit.edu>

pub  1024 0xDDA950C0 2002-09-09 ---------- DSS             Sign & Encrypt
f20    Fingerprint20 = B789 3843 C449 5DC8 CC56  E0CC 209C 2D4C DDA9 50C0
sub  2048 0x86C60D8B 2002-09-09 ---------- Diffie-Hellman
f20    Fingerprint20 = D786 6C68 D9DD 0205 47C2  0854 10B2 683F 86C6 0D8B
uid  Thien-Loc Nguyen <nguyentl@theory.lcs.mit.edu>

pub  1024 0x9129436B 2002-09-09 ---------- DSS             Sign & Encrypt
f20    Fingerprint20 = 858B 67E4 D191 96FA CD34  D984 2919 FED2 9129 436B
sub  2048 0x5E927C64 2002-09-09 ---------- Diffie-Hellman
f20    Fingerprint20 = AA43 C0EC BEA7 7119 A22A  0357 7628 C750 5E92 7C64
uid  Thien-Loc Nguyen <thien-loc.nguyen@polytechnique.org>
```

(b) Fingerprints

1. A PGP public key fingerprint is a shorthand way for naming a public key. It's useful for establishing trust because reading an entire public key over the phone is cumbersome compared to reading a short fingerprint. While a key fingerprint is shorter than the public key, it's semantically the same with respect to trust under a computationally-limited adversarial model.
2. Fingerprints rely on a collision-resistant hash function. In PGP versions < 4, fingerprints use the MD5 hash function. In later versions, PGP uses SHA-1. While most cryptographic hashes provide one-wayness, this is not necessary for PGP fingerprints. All the data hashed is public information. Some students said that one-wayness is required to prevent the private key from being derived. While we do not want to reveal the private key, this is tangential to the public key fingerprint concept. Public-key techniques (not necessarily hashing) ensure that a computationally limited adversary cannot derive the private key given the public key.
3. If we trust that a fingerprint does speak for a person (e.g., by verifying over the phone or in person) and that finding a hash collision is difficult, then we can also assume a public key obtained insecurely speaks for a person if its recomputed fingerprint matches the already trusted fingerprint.

While the term "fingerprint" loosely means a hash of an object, different products have different definitions of a fingerprint. In PGP, the email address is not part of the object hashed into a fingerprint. In the Self-Certifying File System (SFS)[1], HostIDs (aka fingerprints) are the SHA1 hashes of server public keys.

Several students said that fingerprints are "hashes of certificates." We tracked down this misleading information to a document from Network Associates. We tried to obtain a PGP x.509 certificate, but the CAs had either gone bankrupt or no longer supported PGP certificates. A certificate is generally a public key along with meta information such as names, time stamps, and a signature from a centralized certification authority (CA).

We phoned Marc Horowitz, one of the designers of the PGP key server protocol. We agreed that the document is misleading. While there is nothing wrong with creating fingerprints of a certificate, it makes less sense for PGP because it relies more on a Web of Trust model than on a centralized Public Key Infrastructure (PKI). If using hashes of certificates instead of public keys, a single new signature would cause the fingerprint to change. A user's public key fingerprint should not change if the public key remains the same. For the definitive documentation of the PGP standards, see RFC 2440 on `http://web.mit.edu/rfc/rfc2440.txt`. Section 11.2 discusses the PGP key fingerprint format and inputs.

Two clever students asked the TAs in-person or by phone for PGP fingerprints. Kudos to them.

(c) President who??

Below is what we found when searching for `president@whitehouse.gov` on `http://pgp.mit.edu/`. It should be clear that these keys are fake or have little chance of being authentic. This illustrates that while PGP key servers are useful for *finding and distributing* keys, the key server itself provides *no guarantee of key authenticity.* The key server does not act as a CA nor does it authenticate any keys. Some keys might have signatures to vouch for key authenticity, but this is orthogonal to the

[1]`http://www.fs.net/`

services provided by a key server. As a result, anyone can post a fake key under anyone's name. Some keys even have signatures from fake Presidents....

```
           Public Key Server -- Index ‘‘president@whitehouse.gov ’’

Type bits/keyID    Date       User ID
pub  1024/C087CB59 2000/09/17 billy bob clinton <president@whitehouse.gov>
pub  1024/5F3FDC7E 2000/09/06 Forgery?  Of Course it's a Forgery! <forged-presi
dent@forgeries.whitehouse.gov>
pub  1024/1220B09F 2000/09/06 Bill Clinton <president@whitehouse.gov>
pub  1024/75E5EC44 2000/06/04 Bill Clinton <president@whitehouse.gov>
pub  1024/F66E22F0 1998/11/24 William Clinton <president@whitehouse.gov>
pub  1024/E6E7E3B3 1997/02/20 Cody <president@whitehouse.gov>
pub   384/DE09D78D 1980/01/01 Albert Gore <vice-president@whitehouse.gov>
pub   384/23BDC6F9 1980/01/01 William J. Clinton <president@whitehouse.gov>
```

Problem 1-2. Hotel security

This solution has been submitted by John Cogliano, Anthony Degangi, Alex Patino, Shan Sinha. The format and structure have been preserved. Some edits/additions have been made to the contents.

The Singapore Ritz Hotel Security Policy & Requirements

Introduction The Singapore Ritz is seeking to upgrade its door locking mechanism to facilitate a more convenient user experience. We are submitting this document to several manufacturers to facilitate the bidding process.

Purpose and Scope The purpose of this document is to establish a hotel security policy that will be used to guide the manufacturing of a new in-hotel security system. The scope of this document will pertain strictly to the security issues of our new door locking system. Any features or requirements pertaining to the user experience will be outside the scope of this document, unless directly relevant from a security perspective.

Users and Roles The following enumerates the various users that will be interfacing with our new system:

- Guests and their parties.
  - Need access to the rooms they have reserved.
  - Are allowed to use the hotel Gym, Spa and any other rooms the hotel has granted them access to.
- Hotel managers.
  - Print access cards on computer system.
  - Reprogram door locks.
- Front desk agents.
  - Print access cards through computer system.
- Other hotel staff (cleaning staff, room service, etc).

  - Require access to certain rooms based on their duties.
- Security system administrators.
  - Manages accounts on computer system.
  - Controls access privileges.
  - Maintains computer system.

Locations for Consideration Our hotel contains several types of locations, all of which will need to be considered in the design of the various apparatus that will be in our system.

- Standard hotel rooms.
- Gyms, Spa Rooms, Concierge Floors.
- Manager-only rooms.
- Shipping and receiving docks.
- Package rooms - for incoming large mail.
- Employee-only facilities.
- Restrooms.
- Restaurants.

Functionality The security system will consist of three broad areas of functionality:

- Access cards.
- Door locks.
- Computer system.

The remainder of this document will elaborate on the security requirements of each of those functions.

Access Cards

- Access cards will be the primary mechanism used to access any locked facility in the hotel.
  - Upon arriving and checking in, guests will receive an access card to their rooms, and some common rooms. They will keep this access card for the duration of their stay at our hotel.
- Access cards will not be reproducible.
  - A potential adversary should not be able to arbitrarily make a copy of an access card or be able to deduce from a card the method by which to print a new card.
- Access cards will not leak information.
  - A potential adversary should not be able to retrieve any information (e.g. type of card, rooms it grants access to, etc) from a lost/stolen card, be it by physically looking at it or reading its electronic data.
- Access cards will support single card-multiple room access.
  - Managers and hotel staff will typically need access to several rooms.
  - Guests frequently reserve multiple rooms under one name.

- Access cards will come in multiple versions.
  - Open-only – They will only allow to open doors. The list of doors they can open and dates at which they can open these doors will be programmed into the card.
  - Reprogram – Given the assumption that there will be no communication between the door locks and the computer system; we will require the ability to reprogram door locks. Only managers should be able to print this kind of card. They should support single-card, multiple-door reprogramming.
- Access cards will have a short expiration date.
  - To minimize the consequences of lost/stolen/unreturned cards, *open-only* cards should not be valid longer than they need to: guest cards should expire soon after the scheduled checkout date/time, staff cards should expire soon after the scheduled end of duty, etc.
  - As *reprogram* cards serve a critical function in the system, they should be printed on a need-only basis and have a very short expiration date: this expiration date should correspond to a little more than the duration a typical round of door reprogramming. The expiration date for *reprogram* cards should not be programmable, but set once and for all at the setup of the infrastructure.

Door Locks

- Door locks will be reasonably resistant to forms of physical or electrical tampering.
- Door locks will need to support reject access card list.
  - Given that access cards may be lost, stolen or otherwise compromised, door locks will need to keep the information of which cards to reject. This memory can be tagged by the expiration date of the particular card to reject, to ease memory requirements.
- Door locks will remain closed from the outside when their batteries are running out of power.
  - Door locks should have a indicator revealing the state of their batteries. This indicator can be a small light emitting diode: the absence of light would mean the end of the battery.
- Door locks will need to be able to be opened manually at all times from the inside.
  - People should not be prevented to get out when they don't have the proper Access cards (lost/stolen).
  - In case of emergency, people should be able to get out in the shortest possible time.

Computer System

- Maintains a list of users and login data.
- Allows for different levels of access.
  - Manager – should have access to printing all kinds of cards, and have access to log files.
  - Front Desk – should only be able to print guest cards.
  - System Admin – should only be able to access system administration functions.
- Prints/Generates Access cards.
- Records an audit log of all transactions with detailed information.
  - Hotel staff should be accountable for its use/misuse of the system.

–Insiders should not be able to give away Access cards to unauthorized persons.

Problem 1-3. Cookie security

In all the problem, U denotes a user you want to forge a cookie for.

(a) This solution has been adapted from solutions of the following groups:
Michael Kahan, Anthony Kim, Melanie Moy, Mark Rosen
Brent Buddensee, Ariel Segall, David Wilson
Levente Jakab, Joshua Marron, Paul Youn, Enrique Zolezzi

The scheme is actually secure against selective forgery.

Suppose you want to forge a cookie for user U.

Let ϵ be a small number so that $u = U \oplus \epsilon$ does not already exist in the system[2].

Log in as u to get a cookie (u, e, d) where $d = KNC_{\mathcal{K}}(u \oplus e)$.

ϵ being small, $E = e \oplus \epsilon$ is still a valid date in the future[3].

Besides

$$\begin{aligned} \delta(U, E) &= KNC_{\mathcal{K}}(U \oplus E) \\ &= KNC_{\mathcal{K}}(u \oplus \epsilon \oplus e \oplus \epsilon) \\ &= KNC_{\mathcal{K}}(u \oplus e) \\ &= d \end{aligned}$$

The cookie (U, E, d) is our valid forged cookie.

(b) The existential forgery is really simple in this case.

Let e_0 be any valid time in the future, such that $u = e_0$ does not already exist in the system.

Log in as u to get a cookie (u, e, d) where $d = KNC_{\mathcal{K}}(u) \oplus KNC_{\mathcal{K}}(e)$.

Let $(U, E) = (e, u)$.

Then $U = e$ is a user we did not create (which may or may not already exist) and $E = u = e_0$ is a valid time in the future.

Besides

$$\begin{aligned} \delta(U, E) &= KNC_{\mathcal{K}}(U) \oplus KNC_{\mathcal{K}}(E) \\ &= KNC_{\mathcal{K}}(e) \oplus KNC_{\mathcal{K}}(u) \\ &= KNC_{\mathcal{K}}(u) \oplus KNC_{\mathcal{K}}(e) \\ &= d \end{aligned}$$

The cookie $(U, E, d) = (e, u, d)$ is our valid forged cookie.

[2]In practice, you should find some $\epsilon \leq 2^8 - 1 = 255$ by trying all integers starting from 1: that means trying all usernames with the same characters as u but the last character.

[3]Since ϵ has 8 bits or less, E and e have the same bits, but maybe the last 8 ones, so that they are within $2^8 - 1 = 255$ seconds (4 minutes 15 seconds) of each other.

Note: You can also use the forged cookie $(e_0, e_0, 0)$, in which case you don't even have to create a new user!

(c) The right solution to this problem is kind of tricky if we want to get a reasonable valid expiration date.

The bottom line is that you need a way to compute $KNC_{\mathcal{K}}(U+E) \oplus KNC_{\mathcal{K}}(U-E)$ without querying U.

You need to log in 3 users[4] u_1, u_2 and u_3 at appropriate times. You get then 3 cookies (u_1, e_1, d_1), (u_2, e_2, d_2) and (u_3, e_3, d_3), such that

$$\begin{aligned} d_1 &= KNC_{\mathcal{K}}(u_1+e_1) \oplus KNC_{\mathcal{K}}(u_1-e_1) \\ d_2 &= KNC_{\mathcal{K}}(u_2+e_2) \oplus KNC_{\mathcal{K}}(u_2-e_2) \\ d_3 &= KNC_{\mathcal{K}}(u_3+e_3) \oplus KNC_{\mathcal{K}}(u_3-e_3) \end{aligned}$$

Now, in order to have $\delta(U,E) = d_1 \oplus d_2 \oplus d_3$, we can choose $u_1, u_2, u_3, e_1, e_2, e_3$ such that

$$\begin{aligned} U+E &= u_1+e_1 && (1) \\ u_1-e_1 &= u_2-e_2 && (2) \\ u_2+e_2 &= u_3+e_3 && (3) \\ u_3-e3 &= U-E && (4) \end{aligned}$$

Can we find $u_1, u_2, u_3, e_1, e_2, e_3$ satisfying equations (1) to (4)?

We have 4 independent linear equations with 6 unknowns. So we know we can express all solutions using two parameters α and β.

After some algebra, we get

$$\begin{aligned} u_1 &= U+\alpha && (5) \\ u_2 &= U+\alpha-\beta && (6) \\ u_3 &= U-\beta && (7) \\ e_1 &= E-\alpha && (8) \\ e_2 &= E-\alpha-\beta && (9) \\ e_3 &= E-\beta && (10) \end{aligned}$$

Since we want also u_1, u_2, u_3 to be distinct from U (we cannot query U), we need to have $\alpha \neq \beta$.

Now here is how the process works.

Let α, β be two small numbers such that $u_1 = U+\alpha$, $u_2 = U+\alpha-\beta$, $u_3 = U-\beta$ do not already exist in the system[5].

Log in:

[4]you could also do it with two, but you would end up with a date in 2035

[5]In practice, you should find some $\alpha, \beta \leq 2^8 - 1 = 255$ by trying all small pairs of distinct integers. Note that the condition here is stronger than in part (a) since we need u_1, u_2 AND u_3 to be nonexistent in the system, but we get to try $\frac{255*(255-1)}{2} = 32,385$ triplets of usernames with the same characters as u but the last character.

1. using u_2 at some time t_0 to get a cookie (u_2, e_2, d_2) where $e_2 = t_0 + \tau$.
2. using u_3, $\alpha > 0$ seconds after step 1 (at time $t_0 + \alpha$) to get a cookie (u_3, e_3, d_3) where $e_3 = t_0 + \tau + \alpha$.
3. using u_1, $\beta > 0$ seconds after step 1 (at time $t_0 + \beta$) to get a cookie (u_1, e_1, d_1) where $e_1 = t_0 + \tau + \beta$.

Let $E = t_0 + \tau + \alpha + \beta$. Then $U, E, u_1, u_2, u_3, e_1, e_2, e_3$ satisfy equations (4) to (10).

The cookie $(U, E, d_1 \oplus d_2 \oplus d_3)$ is our valid forged cookie.

(d) We assume here that we can query the XORcism server in response to intercepted communications.

Suppose we eavesdrop on user U. As he logs in at time t_0, he gets a cookie (U, e_0, d_0), where $t_0 = e_0 + \tau$.

Let's first explain how we get a cookie (u, e_1, d_1) such that $e_1 = e_0$.

- First, choose any user u such u does not already exist in the system.
- Log in as u whenever we intercept a secure communication from U to the server. We get a set of cookies (u, e_1, d_1).

  Since U logged in using a secure communication, in our cookie set, there is a cookie (u, e_1, d_1) that we obtained at time t_0. Then $e_1 = t_0 + \tau = e_0$.

  However, since we don't know the contents of the communications, we don't know either t_0 nor e_0 at this point. Therefore, we don't know which cookie is the right one.
- When user U logs out, as specified in the email clarification, the cookie (U, e_0, d) transits in clear, and we can intercept it.
- Now, since we know the value of e_0, we can select the cookie (u, e_1, d_1) in our cookie set such that $e_1 = e_0$.

Then, having a cookies (U, e_0, d_0) and (u, e_0, d_1), we can compute

$$\begin{aligned}
\omega &= d_0 \oplus d_1 \\
&= KNC_{\mathcal{K}_\infty}(U) \oplus KNC_{\mathcal{K}_\in}(e_0) \oplus KNC_{\mathcal{K}_\infty}(u) \oplus KNC_{\mathcal{K}_\in}(e_0) \\
&= KNC_{\mathcal{K}_\infty}(U) \oplus KNC_{\mathcal{K}_\infty}(u)
\end{aligned}$$

Now, we create a forged cookie for user U at any time in the following manner.

Log in as u to get a cookie (u, e, d) where $d = KNC_{\mathcal{K}_\infty}(u) \oplus KNC_{\mathcal{K}_\in}(e)$.

Then e is a valid time in the future and

$$\begin{aligned}
\delta(U, e) &= KNC_{\mathcal{K}_\infty}(U) \oplus KNC_{\mathcal{K}_\in}(e) \\
&= KNC_{\mathcal{K}_\infty}(U) \oplus KNC_{\mathcal{K}_\infty}(u) \oplus KNC_{\mathcal{K}_\infty}(u) \oplus KNC_{\mathcal{K}_\in}(e) \\
&= \omega \oplus d
\end{aligned}$$

The cookie $(U, e, \omega \oplus d)$ is our valid forged cookie.