Lecture 9
October 13, 2025

ECS 235A, Computer and Information Security

Administrative Stuff

- Zoom office hour on Wednesday moved to 4:00pm–4:50pm
  - Same Zoom meeting ID and password
- Homework 2 and Extra Credit 2 are now posted
  - They are due on October 22, a week from Wednesday

A Real Homework Assignment (Do Not Do It)

*This is not an assignment for this class. I am **only** asking what you think of it.*

Student is to perform a remote security evaluation of one or more computer systems. The evaluation should be conducted over the Internet, using tools available in the public domain.

What the student must submit.

In conducting this work, you should imagine yourself to be a security expert contracted by the owner of the computer system(s) to perform a security evaluation.

The student must provide a written report which has the following sections: Executive summary, description of tools and techniques used, dates and times of investigations, examples of data collected, evaluation data, overall evaluation of the system(s) including vulnerabilities.

RSA

- First described publicly in 1978
  - Unknown at the time: Clifford Cocks developed a similar cryptosystem in 1973, but it was classified until recently
- Exponentiation cipher
- Relies on the difficulty of determining the number of numbers relatively prime to a large integer *n*

Background

- Totient function ϕ(n)
  - Number of positive integers less than *n* and relatively prime to *n*
    - *Relatively prime* means with no factors in common with *n*
- Example: $\phi(10) = 4$
  - 1, 3, 7, 9 are relatively prime to 10
- Example: $\phi(21) = 12$
  - 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

Algorithm

- Choose two large prime numbers p, q
  - Let $n = pq$; then $\phi(n) = (p–1)(q–1)$
  - Choose $e < n$ such that e is relatively prime to $\phi(n)$.
  - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n); private key: d
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$

Example: Confidentiality

- Take $p = 181$, $q = 1451$, so $n = 262631$ and $\phi(n) = 261000$
- Alice chooses $e = 154993$, making $d = 95857$
- Bob wants to send Alice secret message PUPPIESARESMALL (152015 150804 180017 041812 001111); encipher using public key
  - $152015^{154993} \bmod 262631 = 220160$
  - $150804^{154993} \bmod 262631 = 135824$
  - $180017^{154993} \bmod 262631 = 252355$
  - $041812^{154993} \bmod 262631 = 245799$
  - $001111_{154993} \bmod 262631 = 070707$
- Bob sends 220160 135824 252355 245799 070707
- Alice uses her private key to decipher it

Example: Authentication/Integrity

- Alice wants to send Bob the message PUPPIESARESMALL in such a way that Bob knows it comes from her and nothing was changed during the transmission
  - Same public, private keys as before
- Encipher using private key:
  - $152015^{95857} \bmod 262631 = 072798$
  - $150804^{95857} \bmod 262631 = 259757$
  - $180017^{95857} \bmod 262631 = 256449$
  - $041812^{95857} \bmod 262631 = 089234$
  - $001111^{95857} \bmod 262631 = 037974$
- Alice sends 072798 259757 256449 089234 037974
- Bob receives, uses Alice's public key to decipher it

Example: Both (Sending)

- Same *n* as for Alice; Bob chooses *e* = 45593, making *d* = 235457
- Alice wants to send PUPPIESARESMALL (152015 150804 180017 041812 001111) confidentially and authenticated
- Encipher:
  - $(152015^{95857} \bmod 262631)^{45593} \bmod 262631 = 249123$
  - $(150804^{95857} \bmod 262631)^{45593} \bmod 262631 = 166008$
  - $(180017^{95857} \bmod 262631)^{45593} \bmod 262631 = 146608$
  - $(041812^{95857} \bmod 262631)^{45593} \bmod 262631 = 092311$
  - $(001111^{95857} \bmod 262631)^{45593} \bmod 262631 = 096768$
- So Alice sends 249123 166008 146608 092311 096768

Example: Both (Receiving)

- Bob receives 249123 166008 146608 092311 096768
- Decipher:
  - $(249123^{235457} \bmod 262631)^{154993} \bmod 262631 = 152012$
  - $(166008^{235457} \bmod 262631)^{154993} \bmod 262631 = 150804$
  - $(146608^{235457} \bmod 262631)^{154993} \bmod 262631 = 180017$
  - $(092311^{235457} \bmod 262631)^{154993} \bmod 262631 = 041812$
  - $(096768^{235457} \bmod 262631)^{154993} \bmod 262631 = 001111$
- So Alice sent him 152015 150804 180017 041812 001111
  - Which translates to PUP PIE SAR ESM ALL or PUPPIESARESMALL

Security Services

- Confidentiality
  - Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key
- Authentication
  - Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner

More Security Services

- Integrity
  - Enciphered letters cannot be changed undetectably without knowing private key
- Non-Repudiation
  - Message enciphered with private key came from someone who knew it

Warnings

- Encipher message in blocks considerably larger than the examples here
  - If only characters per block, RSA can be broken using statistical attacks (just like symmetric cryptosystems)
- Attacker cannot alter letters, but can rearrange them and alter message meaning
  - Example: reverse enciphered message of text ON to get NO

Checksums

- Mathematical function to generate a set of *k* bits from a set of *n* bits (where $k \leq n$).
  - *k* is smaller than *n* except in unusual circumstances
- Example: ASCII parity bit
  - ASCII has 7 bits; 8th bit is “parity”
  - Even parity: even number of 1 bits
  - Odd parity: odd number of 1 bits

Example Use

- Bob receives “10111101” as bits.
  - Sender is using even parity; 6 1 bits, so character was received correctly
    - Note: could be garbled, but 2 bits would need to have been changed to preserve parity
  - Sender is using odd parity; even number of 1 bits, so character was not received correctly

Definition of Cryptographic Checksum

Cryptographic checksum h: $A \rightarrow B$:

1. For any $x \in A$, $h(x)$ is easy to compute
2. For any $y \in B$, it is computationally infeasible to find $x \in A$ such that $h(x) = y$
3. It is computationally infeasible to find two inputs x, $x' \in A$ such that $x \neq x'$ and $h(x) = h(x')$
   - Alternate form (stronger): Given any $x \in A$, it is computationally infeasible to find a different $x' \in A$ such that $h(x) = h(x')$.

Collisions

- If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*
  - Pigeonhole principle: if there are n containers for n+1 objects, then at least one container will have at least 2 objects in it.
  - Application: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

Keys

- Keyed cryptographic checksum: requires cryptographic key
  - AES in chaining mode: encipher message, use last *n* bits. Requires a key to encipher, so it is a keyed cryptographic checksum.
- Keyless cryptographic checksum: requires no cryptographic key
  - SHA-512, SHA-3 are examples; older ones include MD4, MD5, RIPEM, SHA-0, and SHA-1 (methods for constructing collisions are known for these)

HMAC

- Make keyed cryptographic checksums from keyless cryptographic checksums
- *h* keyless cryptographic checksum function that takes data in blocks of *b* bytes and outputs blocks of *l* bytes. *k*′ is cryptographic key of length *b* bytes
  - If short, pad with 0 bytes; if long, hash to length *b*
- *ipad* is 00110110 repeated *b* times
- *opad* is 01011100 repeated *b* times
- HMAC-*h*(*k*, *m*) = *h*(*k*′ ⊕ *opad* || *h*(*k*′ ⊕ *ipad* || *m*))
  - ⊕ exclusive or, || concatenation

Strength of HMAC-*h*

- Depends on the strength of the hash function *h*
- Attacks on HMAC-MD4, HMAC-MD5, HMAC-SHA-0, and HMAC-SHA-1 recover partial or full keys
  - Note all of MD4, MD5, SHA-0, and SHA-1 have been broken

Digital Signature

- Construct that authenticates origin, contents of message in a manner provable to a disinterested third party (a “judge”)
- Sender cannot deny having sent message (service is “nonrepudiation”)
  - Limited to *technical* proofs
    - Inability to deny one’s cryptographic key was used to sign
  - One could claim the cryptographic key was stolen or compromised
    - Legal proofs, *etc.,* probably required; not dealt with here

Common Error

- Symmetric: Alice, Bob share key *k*
  - Alice sends *m* || { *m* } *k* to Bob
  - { *m* } *k* means *m* enciphered with key *k*, || means concatenation

Claim: This is a digital signature

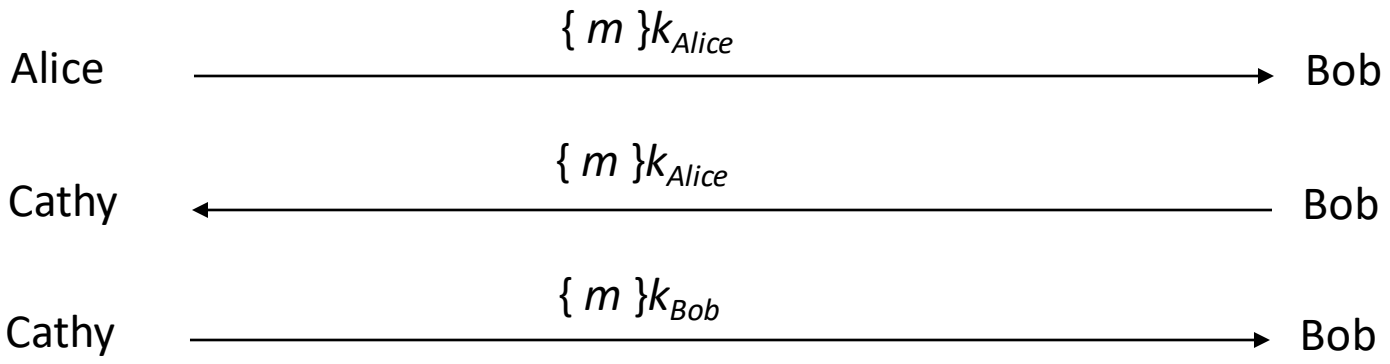
WRONG

This is not a digital signature

- Why? Third party cannot determine whether Alice or Bob generated message

Classical Digital Signatures

- Require trusted third party
  - Alice, Bob each share keys with trusted party Cathy
- To resolve dispute, judge gets $\{ m \} k_{Alice}$, $\{ m \} k_{Bob}$, and has Cathy decipher them; if messages matched, contract was signed



Public Key Digital Signatures

- Basically, Alice enciphers the message, or its cryptographic hash, with her private key
- In case of dispute or question of origin or whether changes have been made, a judge can use Alice's public key to verify the message came from Alice and has not been changed since being signed

RSA Digital Signatures

- Alice's keys are (e_{Alice},n_{Alice}) (public key), d_{Alice} (private key)
  - In what follows, we use e_{Alice} to represent the public key
- Alice sends Bob

$$m \,||\, \{ m \}\, d_{Alice}$$

- In case of dispute, judge computes

$$\{ \{ m \}\, d_{Alice} \}\, e_{Alice}$$

- and if it is m, Alice signed message
  - She's the only one who knows d_{Alice}!

RSA Digital Signatures

- Use private key to encipher message
  - Protocol for use is *critical*
- Key points:
  - Never sign random documents, and when signing, always sign hash and never document
  - Don’t just encipher message and then sign, or vice versa
    - Changing public key and private key can cause problems
    - Messages can be forwarded, so third party cannot tell if original sender sent it to her

Attack #1

- Example: Alice, Bob communicating
  - $n_A = 262631$, $e_A = 154993$, $d_A = 95857$
  - $n_B = 288329$, $e_B = 22579$, $d_B = 138091$
- Alice asks Bob to sign 225536 so she can verify she has the right public key:
  - $c = m^{d_B} \bmod n_B = 225536^{138091} \bmod 288329 = 271316$
- Now she asks Bob to sign the statement AYE (002404):
  - $c = m^{d_B} \bmod n_B = 002404^{138091} \bmod 288329 = 182665$

Attack #1

- Alice computes:
  - new message NAY (130024) by (002404)(225536) mod 288329 = 130024
  - corresponding signature (271316)(182665) mod 288329 = 218646
- Alice now claims Bob signed NAY (130024), and as proof supplies signature 218646
- Judge computes $c^{e_B} \bmod n_B = 218646^{22579} \bmod 288329 = 130024$
  - Signature validated; Bob is toast

Preventing Attack #1

- Do not sign random messages
  - This would prevent Alice from getting the first message
- When signing, always sign the cryptographic hash of a message, not the message itself

Attack #2: Bob's Revenge

- Bob, Alice agree to sign contract LUR (112017)
  - But Bob really wants her to sign contract EWM (042212), but knows she won't
- Alice enciphers, then signs:
  - $(m^{e_B} \bmod n_A)^{d_A} \bmod n_A = (112017^{22579} \bmod 288329)^{95857} \bmod 262631 = 42390$
- Bob now changes his public key
  - Computes r such that $042212^r \bmod 288329 = 112017$; one such $r = 9175$
  - Computes $re_B \bmod \phi(n_B) = (9175)(22579) \bmod 287184 = 102661$
  - Replace public key with (102661,288329), private key with 161245
- Bob claims contract was EWM
- Judge computes:
  - $(42390^{154993} \bmod 262631)^{161245} \bmod 288329 = 042212$, which is EWM
  - Verified; now Alice is toast

Preventing Attack #2

- Obvious thought: instead of encrypting message and then signing it, sign the message and then encrypt it
  - May not work due to surreptitious forwarding attack
  - Idea: Alice sends Cathy an encrypted signed message; Cathy deciphers it, re-enciphers it with Bob's public key, and then sends message and signature to Bob – now Bob thinks the message came from Alice (right) and was intended for him (wrong)
- Several ways to solve this:
  - Put sender and recipient in the message; changing recipient invalidates signature
  - Sign message, encrypt it, then sign the result