Announcements

• Project 4 is Due Friday May 2nd at 11:59 PM

• Final exam:
  – Friday, May 12th. Noon - 2:00pm   DRLB A6

• Today:
  – Last details about voting
  – Secret sharing
  – Anonymity / Onion Routing / Crowds
    • Slides adapted from some by Levente Buttyán
Blind Signatures

- Digital signature scheme equipped with a commutative *blinding* operation
  - Signer never learns what they signed
  - Like signing an envelope with a window (or with carbon paper)
  - I.e.: \( \text{unblind}(\text{sign}(\text{blind}(m))) = \text{sign}(m) \)

- Voting scheme:
  - Voter prepares vote \( v \), blinds, and authenticates to Authorization server, and sends vote. Server checks off voter, signs vote, and sends back to voter. Voter unblinds and now has \( \text{sign}(v) \).
  - Voter anonymously sends \( \text{sign}(v) \) to Tabulation server. Server checks signature, then counts vote.
Homomorphic Encryption

- A *homomorphic* encryption scheme has an operator ★ such that Enc(m) ★ Enc(n) = Enc(m ★ n). ★ is usually either + or ×, never both.
  - E.g. both RSA and El Gamal have ×.

- Voting scheme:
  - Suppose scheme has + as homomorphism and votes are either 0 or 1.
  - Voter prepares Enc(0) or Enc(1) as vote, authenticates to Tabulation server, and submits vote.
  - Tabulation server sums all the votes, then decrypts result. Individual votes never decrypted.

- Need additional checks to ensure that the voters don't cheat by submitting Enc(2) or Enc(-17)
  - Civitas Solution: use a zero knowledge proof that shows the vote is a re-encryption of either 0 or 1.
  - Theory due to Hirt & Sako
Secret Sharing

• How to share a secret among N+1 players:
  – Owner of the secret generates N random bitstrings R1 ... RN
  – Player 0 gets S ⊕ R1 ⊕ ... ⊕ RN
  – Player j > 0 gets Rj
  – All N players can cooperate to recover S -- they just XOR their shares.

• Threshold schemes allow k-out-of-N players to recover the secret:
  – Owner of the secret picks a random polynomial f with degree (k-1) such that f(0) = S
  – Player j > 0 gets f(j)
  – If any k players get together, they can use Lagrange interpolation to calculate f(0)
  – If fewer than k players get together, there's no information about f(0).
Lagrange Interpolation

The Lagrange interpolating polynomial is $P(x)$ that passes through $n$ points: $(x_1, y_1 = f(x_1)), \ldots, (x_n, y_n = f(x_n))$

$$P(x) = \frac{(x - x_2)(x - x_3)\cdots(x - x_n)}{(x_1 - x_2)(x_1 - x_3)\cdots(x_1 - x_n)} y_1 + \frac{(x - x_1)(x - x_3)\cdots(x - x_n)}{(x_2 - x_1)(x_2 - x_3)\cdots(x_2 - x_n)} y_2 + \cdots + \frac{(x - x_1)(x - x_2)\cdots(x - x_{n-1})}{(x_n - x_1)(x_n - x_2)\cdots(x_n - x_{n-1})} y_n.$$
Example: 3-out-of-N Secret

- Suppose the secret is $S = 7$
- I generate (at random) $f(x) = 2x^2 - 3x + 7$
- Then $S = f(0) = 7$
  - Share $s_1 = f(1) = 6$
  - Share $s_2 = f(2) = 9$
  - Share $s_3 = f(3) = 16$
  - Share $s_4 = f(4) = 27$

- To recover secret and obtain 3 shares:
  - Example: given $s_2, s_3, s_4 = (2,9) (3,16) (4,27)$
  - Calculate $P(x)$ as on the previous slide [see blackboard]
Mix networks: Anonymity

• Chaum 1981: Basic Mix network
• Suppose that there are $N$ servers with public keys $K_1 \ldots K_N$.
• A mix message $M_a$ looks like: $K_1\{K_2\{\ldots K_N\{m_a\}\}\}$
• To anonymize a set of messages $M_1$, $M_2$, ..., $M_j$:
  – Server $i$ decrypts the messages, permutes them, and forwards them to server $i+1$
  – The last server will reveal $m_1$, $m_2$, ..., $m_j$ in some random permutation:

![Diagram of mix network](attachment:diagram.png)
Mix Networks

• Original Chaumian decryption mix:
  – Implemented with set of servers
  – Input: list of encrypted values
    • Enc(Enc(Enc(…c…)))
  – Output: same list, decrypted
    • But order of list permuted
  – Each server in mix permutes list and removes one layer of encryption

• Civitas based on a re-encyprtion mix network
  – Input: List of encrypted messages
  – Output: Permutated list of re-encrypted messages
  – Re-encryption in El Gamal requires only the public key
Mix Network Voting Schemes

• Voting scheme:
  – Voter encrypts vote, authenticates to Ballot Box server, submits vote.
  – Set of tabulation tellers run a mixnet over the encrypted votes, resulting in random permutation of votes.
  – Permuted list is decrypted and tallied.
Preserving web privacy

- Your IP address may be visible to web sites
  - This may reveal your employer, ISP, etc.
  - Can link activities on different sites, different times

- Can you prevent sites from learning about you?
  - Anonymizer
    - Single site that hides origin of web request
  - Crowds
    - Distributed solution
  - Onion Routing
    - Unlinkability of sender and receiver
Anonymity?

• Sender anonymity:
  – The identity of the sender is hidden, while the receiver (and message) might not be

• Receiver anonymity:
  – The identity of the receiver is hidden (message and sender might not be)

• Unlinkability of sender and receiver:
  – Although the sender and receiver can be identified as participating in communication, they cannot be identified as communicating \textit{with each other}. 
Browsing Anonymizers

- Anonymizer.com
- Web Anonymizer hides your IP address

What does anonymizer.com know about you?

www.anonymizer.com/cgi-bin/redirect.cgi?url=...
Related approach to anonymity

- Hide source of messages by routing them randomly
- Routers don’t know for sure if the apparent source of the message is the actual sender or simply another router
  - Only secure against local attackers!
- Existing systems: Freenet, Crowds, etc.
Crowds

Sender randomly chooses a path through the crowd
Some routers are honest, some corrupt
After receiving a message, honest router flips a coin
  - With probability $P_f$ routes to the next member on the path
  - With probability $1 - P_f$ sends directly to the recipient

http://avirubin.com/crowds.pdf
What Does Anonymity Mean?

- **Degree of anonymity:**
  - Ranges from absolute privacy to provably exposed

- **Beyond suspicion**
  - The observed source of the message is no more likely to be the actual sender than anybody else

- **Probable innocence**
  - Probability <50% that the observed source of the message is the actual sender

  Guaranteed by Crowds if there are sufficiently few corrupt routers

- **Possible innocence**
  - Non-trivial probability that the observed source of the message is not the actual sender
A real-time MIX network – Onion routing

• general purpose infrastructure for anonymous communications over a public network (e.g., Internet)
• supports several types of applications (HTTP, FTP, SMTP, rlogin, telnet, …) through the use of application specific proxies
• operates over a (logical) network of onion routers
  – onion routers are real-time Chaum MIXes (messages are passed on nearly in real-time → this may limit mixing and weaken the protection!)
  – onion routers are under the control of different administrative domains → makes collusion less probable
• anonymous connections through onion routers are built dynamically to carry application data
• distributed, fault tolerant, and secure
Overview of OR architecture

Application (initiator) -> application proxy
- prepares the data stream for transfer
- sanitizes appl. data
- processes status msg sent by the exit funnel

Application proxy
- opens connection via the OR network
- encrypts/decrypts data

Onion router

Entry funnel
- multiplexes connections from onion proxies

Exit funnel
- demultiplexes connections from the OR network
- opens connection to responder application and reports a one byte status msg back to the application proxy

Long-term socket connections

Application (responder)
OR network setup and operation

• long-term socket connections between “neighboring” onion routers are established → links
• neighbors on a link setup two DES keys using the Station-to-Station protocol (one key in each direction)
• several anonymous connections are multiplexed on a link
  – connections are identified by a connection ID (ACI)
  – an ACI is unique on a link, but not globally
• every message is fragmented into fixed size *cells* (48 bytes)
• cells are encrypted with DES in OFB mode (null IV)
  – optimization: if the payload of a cell is already encrypted (e.g., it carries (part of) an onion) then only the cell header is encrypted
• cells of different connections are mixed, but order of cells of each connection is preserved
Anonymous connection setup

• the application is configured to connect to the application proxy instead of the real destination

• upon a new request, the application proxy
  – decides whether to accept the request
  – opens a socket connection to the onion proxy
  – passes a *standard structure* to the onion proxy
  – standard structure contains
    • application type (e.g., HTTP, FTP, SMTP, …)
    • retry count (number of times the exit funnel should retry connecting to the destination)
    • format of address that follows (e.g., NULL terminated ASCII string)
    • address of the destination (IP address and port number)
  – waits response from the exit funnel before sending application data
Anonymous connection setup (2)

- upon reception of the standard structure, the onion proxy
  - decides whether to accept the request
  - establishes an anonymous connection through some randomly selected onion routers by constructing and passing along an onion
  - sends the standard structure to the exit funnel of the connection
  - after that, it relays data back and forth between the application proxy and the connection

- upon reception of the standard structure, the exit funnel
  - tries to open a socket connection to the destination
  - it sends back a one byte status message to the application proxy through the anonymous connection (in backward direction)
  - if the connection to the destination cannot be opened, then the anonymous connection is closed
  - otherwise, the application proxy starts sending application data through the onion proxy, entry funnel, anonymous connection, and exit funnel to the destination
Onions

- An onion is a multi-layered data structure
- It encapsulates the route of the anonymous connection within the OR network
- Each layer contains
  - Backward crypto function (DES-OFB, RC4)
  - Forward crypto function (DES-OFB, RC4)
  - IP address and port number of the next onion router
  - Expiration time
  - Key seed material
    - Used to generate the keys for the backward and forward crypto functions
- Each layer is encrypted with the public key of the onion router for which data in that layer is intended
Anonymous connection setup
Anonymous connection setup

onion proxy

bwd: entry funnel, crypto fns and keys
fwd: blue, ACI = 12, crypto fns and keys

application (responder)
Anonymous connection setup

onion proxy

ACI = 12

onion

application (responder)
Anonymous connection setup

bwd: magenta, ACI = 12, crypto fns and keys
fwd: green, ACI = 8, crypto fns and keys
Anonymous connection setup

onion proxy

onion

ACI = 8

application (responder)
Anonymous connection setup

- Onion proxy
- Application (responder)
- Onion proxy

Bwd: blue, ACI = 8, crypto fns and keys

Fwd: exit funnel
Anonymous connection setup

bwd: entry funnel, crypto fns and keys
fwd: blue, ACI = 12, crypto fns and keys

bwd: magenta, ACI = 12, crypto fns and keys
fwd: green, ACI = 8, crypto fns and keys

bwd: blue, ACI = 8, crypto fns and keys
fwd: exit funnel

open socket
application (responder)
Data movement

- **forward direction**
  - the onion proxy adds all layers of encryption as defined by the anonymous connection
  - each onion router on the route removes one layer of encryption
  - responder application receives plaintext data

- **backward direction**
  - the responder application sends plaintext data to the last onion router of the connection (due to sender anonymity it doesn’t even know who is the real initiator application)
  - each onion router adds one layer of encryption
  - the onion proxy removes all layers of encryption