Lecture 13

Access Control

1

1

Recall: Security Services

- Confidentiality: to assure information privacy and secrecy
- Authentication: to assert who created or sent data
- Integrity: to show that data has not been altered
- Access Control: prevent misuse of resources = control access to them

  e.g., files, directories, accounts, printers, computers, IoT devices, etc.
- Availability: to offer access to resources, permanence, non-erasure

2

2

Access Control (AC)

- **A "language" for expressing access control policies: who can access what, how and when …**
- **Enforcement of access control**
  - Identify all resources (objects) and their granularity
  - Identify all potential users (subjects)
  - Specify rules for subject/object interaction
  - Guard them in real time

3

3

Model and Terminology

- **Subjects:** users or processes
- **Objects:** resources (files, memory, printers, routers, plotters, disks, processes, etc., etc.,...)

4

4

Focus of Access Control

- **What a subject is allowed to do**

- **What may be done with an object**

5

5

Access Modes

- **"Look" at an object, e.g.:**
  - Read file
  - Check printer queue
  - Read screen
  - Query database
  - Turn on/use microphone, etc., etc.

- **"Change" an object, e.g.:**
  - Write/append/erase file
  - Print on a printer
  - Display on screen
  - Use speakers (audio out)
  - Send packets via WiFi/Bluetooth, etc., etc.

6

6

Access Modes: Bell-Lapadula model

4 mode of access: execute, read, append, and write

	Execute	Read	Append	Write
Observe		X		X
Alter			X	X

7

7

UNIX/Linux/*x Operating Systems

- **execute:** execute (program) file, search directory
- **read:** read from file, list directory
- **write:** write (re-write or append) file, create or rename file in directory

8

8

AC Types

Who is in charge of setting AC policy?

- **Discretionary Access Control (DAC):** resource owner
- **Mandatory Access Control (MAC):** system-wide policy

9

9

Access Control Structures

i. **MAC: Access Control Matrix**

ii. **MAC: Access Control Lists**

iii. **DAC: Capabilities**

10

10

Access Control Matrix

Object

Subject	Bill.doc	Edit.exe	Fun.com
Alice	{0}	{execute}	{execute,read}
Bob	{read,write}	{execute}	{execute,read,write}

11

11

Access Control Lists 1/2

Keep access rights to an object with that object:

- **ACL for bill.doc:**
  - **Bob: read, write**
- **ACL for edit.exe:**
  - **Alice: execute;**
  - **Bob: execute**
- **ACL for fun.com:**
  - **Alice: execute, read;**
  - **Bill: execute, read, write**

- As many ACLs as there are objects
- Each ACL either signed or stored in protected place
- Faster when subject # < object #

12

12

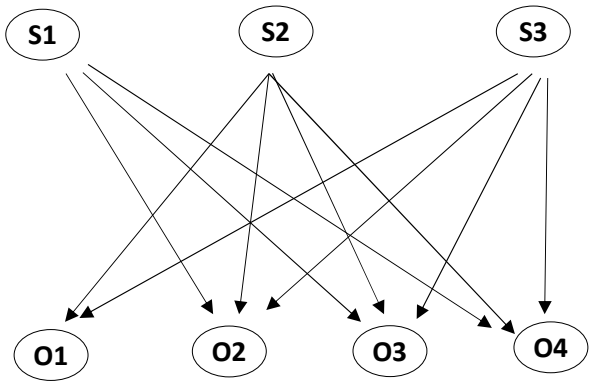
Access Control Lists 2/2

- **Managing access rights can be difficult**
- **Groups can be helpful …**
- **Groups simplify definition of access control policies**

13

13

Access Control Lists w/o grouping

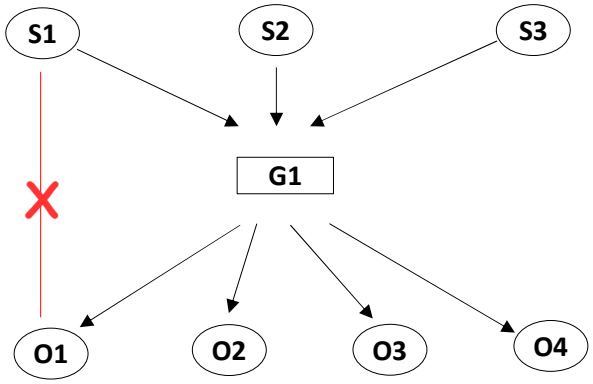


# of rules: (4*3 – 1) = 11

14

14

Access Control Lists with grouping



Reduce # of rules from 11 to (4 + 1) = 5

15

15

Capabilities 1/2

- **Capabilities are associated with Discretionary Access Control (DAC)**
- **Reason: difficult to get full view of who has permission to access an object**

16

16

Capabilities 2/2

Keep access rights with the subject:

- **Alice's capabilities:**
  - **[edit.exe:execute];**
  - **[fun.com:execute,read]**
- **Bob's capabilities:**
  - **[bill.doc:read,write]**
  - **[edit.exe:execute]**
  - **[fun.com:execute,read,write]**

- As many capabilities as there are subject/object pairs
- Each capability either signed or otherwise protected
- Hard to revoke in a distributed setting --- owners and objects have to keep track of all issued capabilities
- Faster when subject # > object #

17

17

In Summary

- **Centralized Systems:**
  - **ACLs/MAC are better**
- **Distributed Systems:**
  - **Capabilities/DAC are better**

18

18

ROLE BASED ACCESS CONTROL (RBAC)

19

19

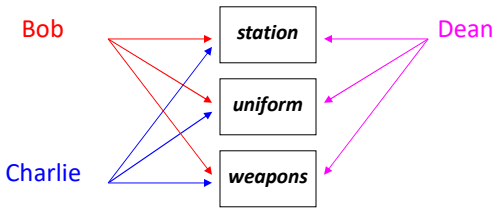
RBAC Basics

- Users are associated with roles
- Roles are associated with permissions
- A user has permission only if s/he has a role associated with that permission
- A bit similar to ACL that only has groups as subjects, no single user

20

20

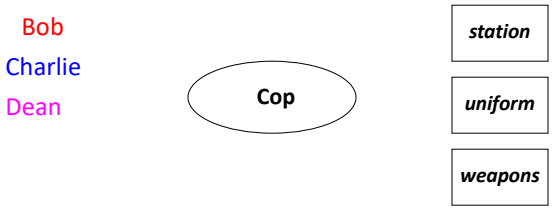
Example: Cops (aka Police Officers) (User/Permission Association)

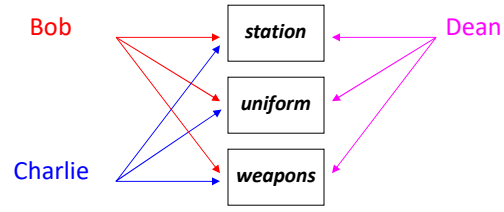


21

21

Example: RBAC

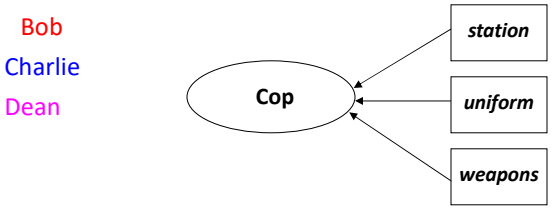


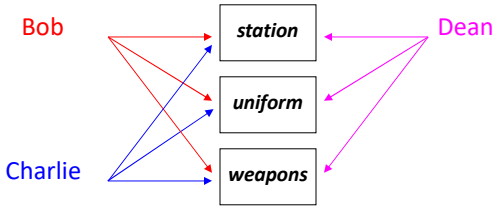


22

22

Example: RBAC

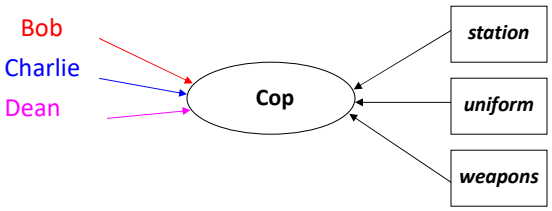




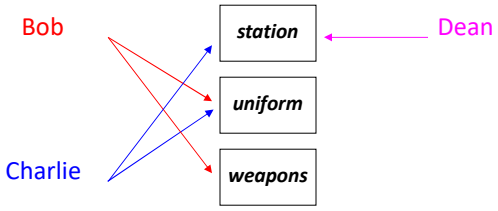
23

23

Example: RBAC



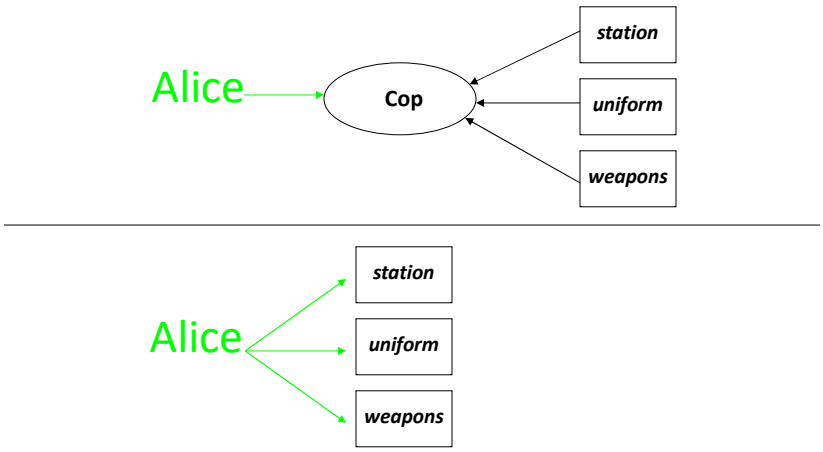
Here RBAC doesn't work …



24

24

Example: Alice becomes a Cop



25

25

CS 134

SPRING 2019

LECTURE 14

Privacy and Anonymity

26

26

Privacy

- Privacy and Society
  - Basic individual right & desire
  - Also relevant to corporations & government agencies
  - Recently increased awareness
    - But, public's perception of privacy is fickle
- Privacy and Technology in Recent Years
  - >>Information disclosed on the Internet
  - >>Handling and transfer of sensitive information
  - <<Privacy and accountability



(Image from geekologie.com)

27

27

Privacy on Public Networks

- The Internet is designed as a public network
  - Anyone on your LAN (wired or wireless) can see your traffic
  - Network routers see all traffic that passes through them
- Routing information is public
  - IP packet headers identify source and destination addresses
  - A passive observer can easily figure out who is talking to whom
- Encryption (e.g., SSL or IPSec) does not hide identities
  - Encryption hides payload, not routing+addressing information
  - Even IP-level encryption (tunnel-mode IPsec/ESP) reveals IP addresses of IPsec gateways

28

28

Applications of Anonymity (1)

- Privacy
  - Hide online transactions, web browsing, etc. from intrusive governments, marketers, archival/search entities (e.g., Google) as well as from criminals and snoops
- Untraceable Electronic Mail
  - Corporate whistle-blowers
  - Political dissidents in oppressive societies
  - Socially sensitive communications
  - Confidential business negotiations
- Law Enforcement and Intelligence
  - Sting operations
  - Secret communications on a public network
    - Informers, secret agents, etc.

29

29

Applications of Anonymity (2)

- Digital/Electronic Cash
  - Electronic currency with properties of paper money (online purchases unlinkable to buyer's identity)
- Anonymous Electronic Voting
- Censorship-Resistant Publishing
- Crypto-Anarchy
  - "Some people say that "anarchy won't work." That's not an argument against anarchy; that's an argument against work." – Bob Black

30

30

Applications of Anonymity (3)

- Human Trafficking
- Libel
- Disinformation = Fake News / Propaganda
- Sale of Illegal Substances (e.g., SilkRoad)
- Tax Avoidance (via Untraceable Payments)
- Incitement to Criminal Activity (e.g., Murder, Rioting, Genocide, Terrorism)

31

31

What is Anonymity?

- **Anonymity**: inability to identify someone within a set of subjects (size varies)
  - Different from PRIVACY – right to be left alone
  - To be anonymous, need to hide your activities among similar activities by others
  - One cannot be anonymous alone!
    - Big difference between anonymity and confidentiality
- **Unlinkability**: separation of action and identity performing that action
  - For example, sender and his email are no more related after observing communication than they were before
- **Unobservability**: inability to tell whether a certain action took place
  - very hard to achieve

32

32

Attacks on Anonymity

- Passive Traffic Analysis
  - Infer from network traffic who is talking to whom
  - To hide your traffic, must carry other people's traffic!
- Active Traffic Analysis
  - Inject packets or put a timing signature on a packet flow
- Compromise of Network Nodes (such as Routers)
  - Not obvious which nodes have been compromised
    - Attacker may be passively logging traffic
  - Do not fully trust any individual node
    - Assume that some fraction of nodes is good, but do not know which

33

33

Chaum's Mix
(David Chaum, ca. 1980-1981)

- Earliest proposal for anonymous email:
  - David Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms", Communications of the ACM, February 1981.
- Public-key crypto + trusted re-mailer (Mix)
  - Untrusted communication medium
  - Public-keys used as persistent pseudonyms

Before spam, people thought anonymous email was a good idea ☺

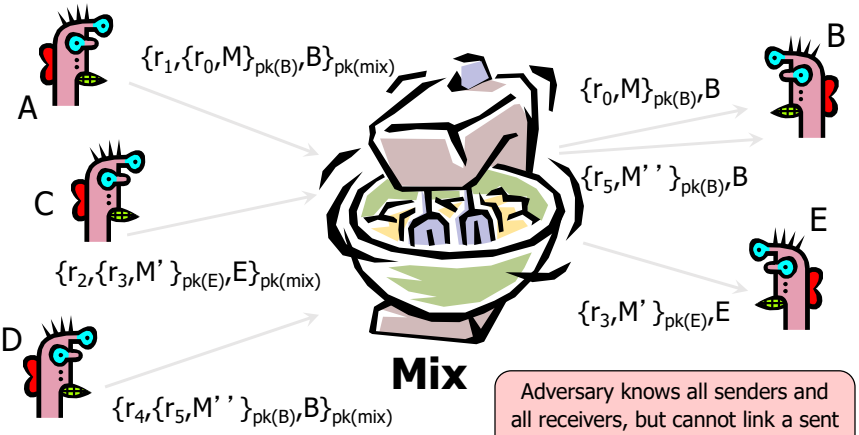
- Modern anonymity systems use Mix as the basic building block

34

34

Basic Mix Design

$\{M\}_{pk(mix)}$: Use Pk_{mix} to encrypt M



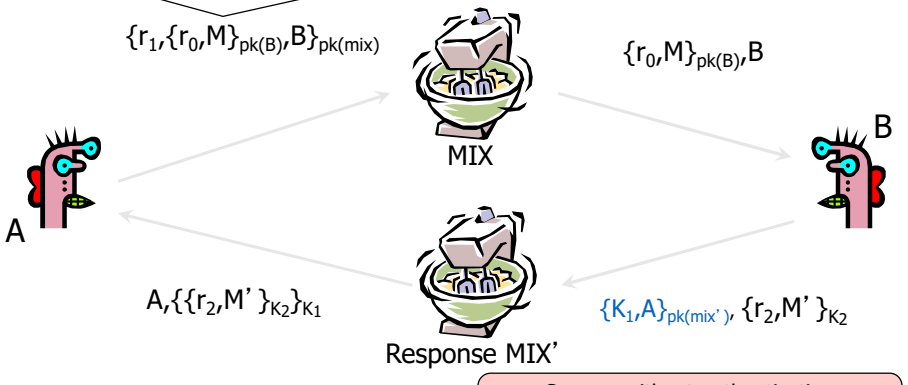
Adversary knows all senders and all receivers, but cannot link a sent message with a received message

35

35

Anonymous Return Addresses (When B does not know the sender)

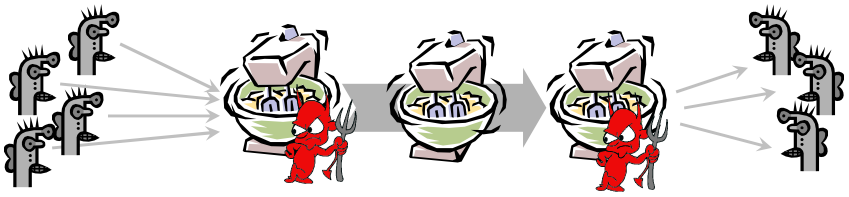
M includes $\{K_1,A\}_{pk(mix')}$, K_2, where K_2 is a fresh public key and MIX' is possibly different from MIX



Secrecy without authentication
(good for an online confession service)

36

Mix Cascade



- Messages are sent through a sequence of mixes
  - Can also form an arbitrary network of mixes ("mixnet")
- Some mixes may be controlled by attacker, but even a single good mix guarantees some anonymity
- Pad and buffer traffic to foil correlation attacks

37

37

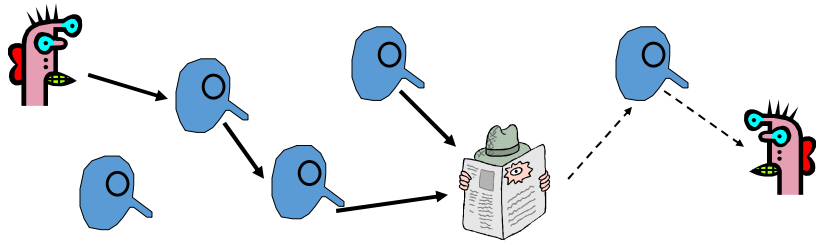
Disadvantages of Basic Mixnets

- Public-key encryption and decryption at each mix are computationally expensive
- Basic mixnets have high latency
  - Ok for email, but not for anonymous Web browsing
- Challenge: low-latency anonymity network
  - Use public-key cryptography to establish a "circuit" with pairwise symmetric keys between hops on the circuit
  - Then use symmetric decryption and re-encryption to move data messages along the established circuits
  - Each node behaves like a mix; anonymity is preserved even if some nodes are compromised

38

38

Another Idea: Randomized Routing



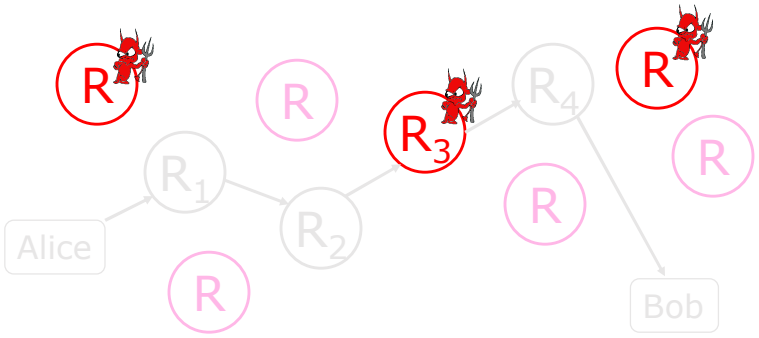
- Hide sources by routing messages randomly
  - Popular technique: Crowds, Freenet, Onion routing
- Routers do not know if the apparent source of a message is the true sender or another router

39

39

[Reed, Syverson, Goldschlag 1997]

Onion Routing (early Tor)



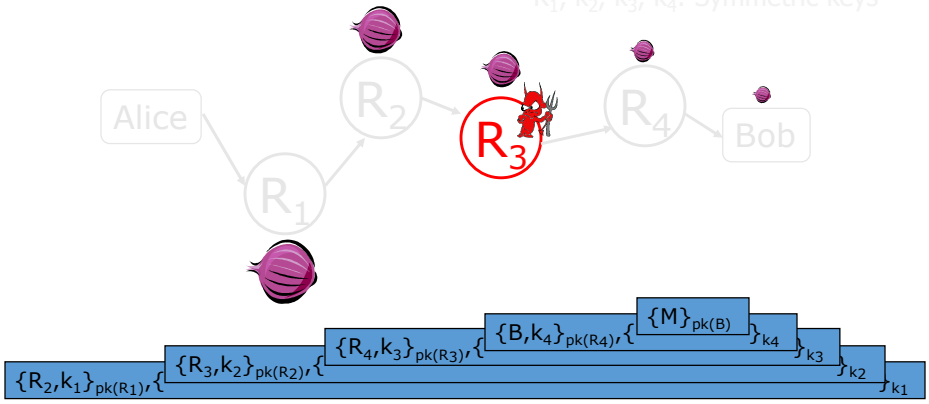
- Sender chooses a random sequence of routers
  - Some routers are honest, some are not
  - Sender controls path length

40

40

Route Establishment (Early Tor)

K_1, k_2, k_3, k_4: Symmetric keys



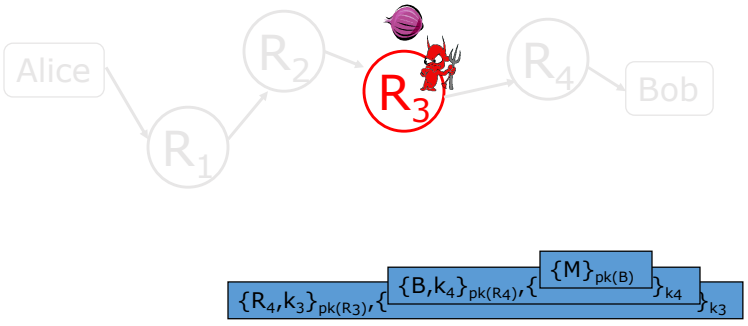
- Routing info for each link encrypted with router's public key

41

41

Route Establishment (Early Tor)

K_1, k_2, k_3, k_4: Symmetric keys



- Routing info for each link encrypted with router's public key
- Each router learns only the identity of the next router

42

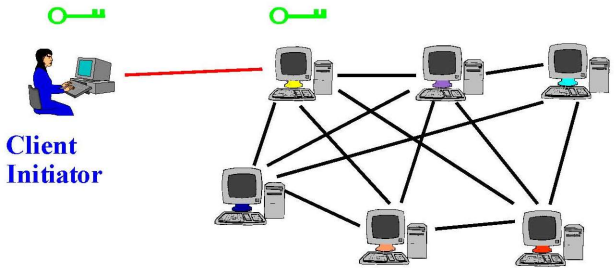
42

The Onion Router (current Tor)

- Second-generation onion routing network
  - http://tor.eff.org
  - Specifically designed for low-latency anonymous Internet communications (e.g., Web browsing)
  - Running since October 2003
- Hundreds of nodes on all continents
- 2.5 million users as of 05/2019
- "Easy-to-use" client proxy
  - Freely available, can use it for anonymous browsing
  - Available for smartphones and tablets too
  - TorBrowser – get it!!!

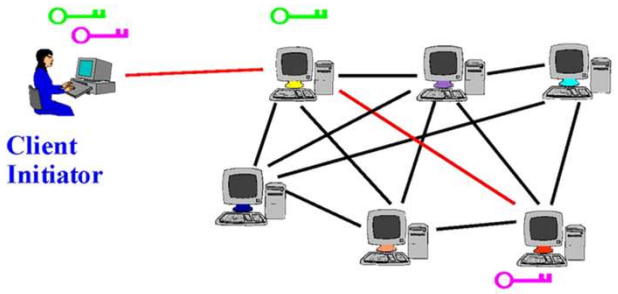
Tor Circuit Setup (1)

- Client proxy establishes a symmetric session key and circuit with Onion Router #1



Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #2
  - Tunnel through Onion Router #1

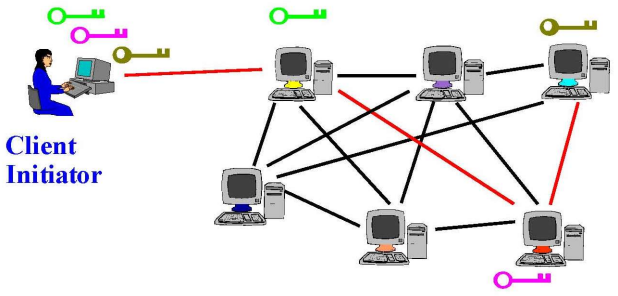


45

45

Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with Onion Router #3
  - Tunnel through Onion Routers #1 and #2

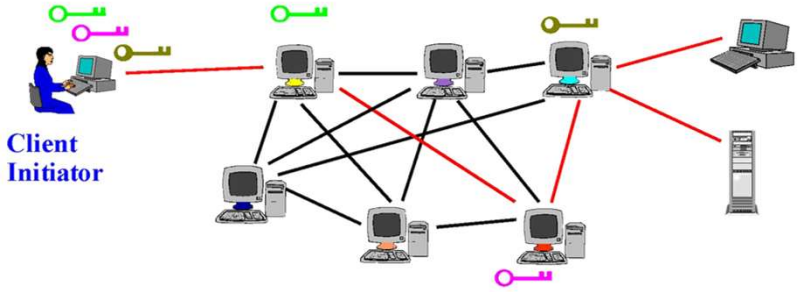


46

46

Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit (also to multiple dst-s)
  - Datagrams are decrypted and re-encrypted at each link



47

47

Tor Management Issues

- Many applications can share one circuit
  - Multiple TCP streams over one anonymous connection
- Tor router do not need root privileges
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- Directory servers
  - Maintain lists of active onion routers, their locations, current public keys, etc.
  - Control how new routers join the network
    - "Sybil attack": attacker creates a large number of routers
  - Directory servers' keys ship with Tor code

48

48

Deployed Anonymity Systems

- Free Haven project has an excellent bibliography on anonymity
  - http://www.freehaven.net/anonbib
- Tor (http://tor.eff.org)
  - Overlay circuit-based anonymity network
  - Best for low-latency applications such as anonymous Web browsing
- Mixminion (http://www.mixminion.net)
  - Network of mixes
  - Best for high-latency applications such as anonymous email

49

49

Dining Cryptographers

- How to make a message public, but in a perfectly untraceable manner
  - David Chaum. "The dining cryptographers problem: unconditional sender and recipient untraceability." Journal of Cryptology, 1988.
- Guarantees information-theoretic anonymity for message senders
  - VERY strong form of anonymity: defeats adversary who has unlimited computational power
- Difficult to make practical
  - In group of size **N**, need **N random bits to send 1 bit**

50

50

Three-Person DC Protocol

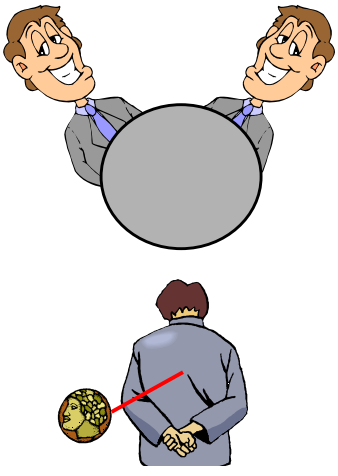
- Three cryptographers are having dinner.
- Either NSA is paying for the dinner, or one of them is paying, but wishes to remain anonymous.
- The cryptographers respect each other's right to make an anonymous payment, but want to find out whether the NSA paid
- So they decide to execute a two-stage protocol:

1. Each diner flips a coin and shows it to his left neighbor.
   - Every diner sees two coins: his own and his right neighbor's

51

51

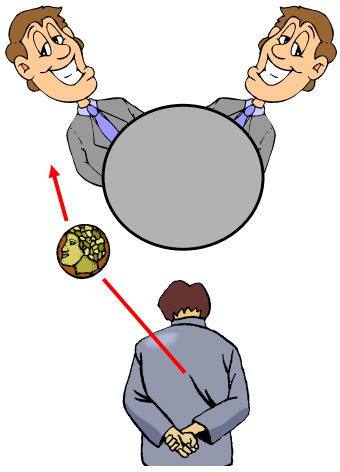
Each diner flips a coin and shows it to his left neighbor



52

52

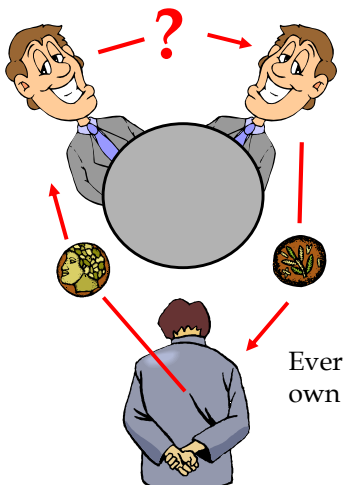
Each diner flips a coin and shows it to his left neighbor



53

53

Each diner flips a coin and shows it to his left neighbor



Every diner sees two coins: his own and his right neighbor's

54

54

Three-Person DC Protocol

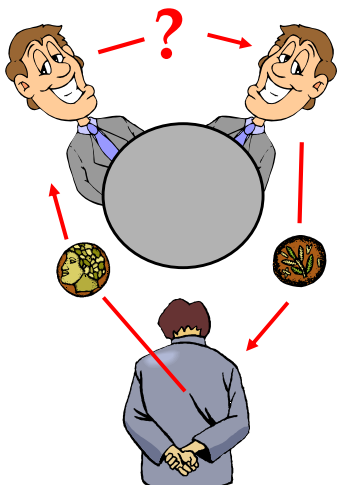
- Three cryptographers are having dinner.
- Either NSA is paying for the dinner, or one of them is paying, but wishes to remain anonymous.
- The cryptographers respect each other's right to make an anonymous payment, but want to find out whether the NSA paid
- So they decide to execute a two-stage protocol:

1. Each diner flips a coin and shows it to his left neighbor.
   - Every diner sees two coins: his own and his right neighbor's
2. Each diner announces whether the two coins are the same. If he is the payer, he lies (says the opposite).

55

55

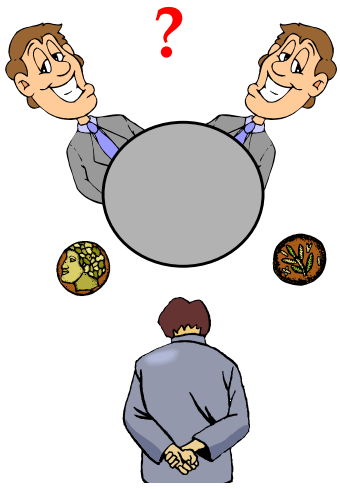
Each diner flips a coin and shows it to his left neighbor



56

56

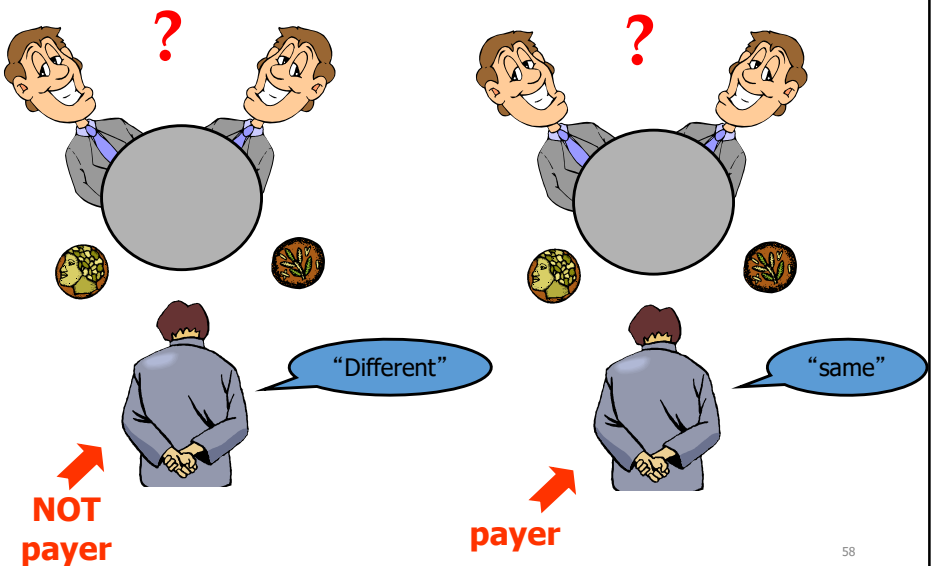
Each diner flips a coin and shows it to his left neighbor



57

57

If he is the payer, he lies (says the opposite)



58

58

Three-Person DC Protocol

- Three cryptographers are having dinner.
- Either NSA is paying for the dinner, or one of them is paying, but wishes to remain anonymous.
- The cryptographers respect each other's right to make an anonymous payment, but want to find out whether the NSA paid
- So they decide to execute a two-stage protocol:

1. Each diner flips a coin and shows it to his left neighbor.
   - Every diner sees two coins: his own and his right neighbor's
2. Each diner announces whether the two coins are the same. If he is the payer, he lies (says the opposite).
3. IF Number of "same"=1 or 3 $\Rightarrow$ NSA is paying

   IF Number of "same"=0 or 2 $\Rightarrow$ one of them is paying
   - But a non-payer cannot tell which of the other two is paying!

59

59

Let's see why it is correct

- 8 Possibilities

Coin 1	Coin 2	Coin 3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

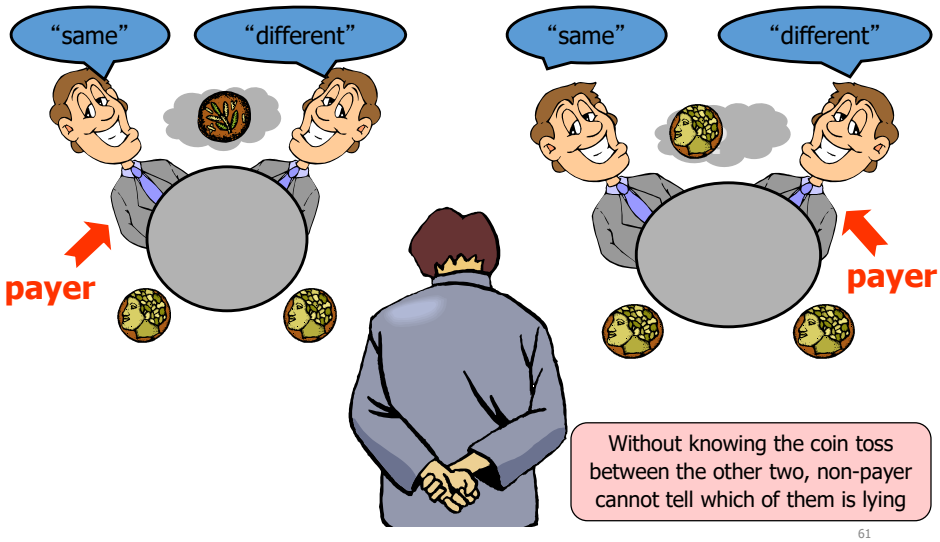
Coin 1, 2	Coin 2, 3	Coin 3, 1
Same	Same	Same
Same	Different	Different
Different	Different	Same
Different	Same	Different
Different	Same	Different
Different	Different	Same
Same	Different	Different
Same	Same	Same

If nobody is lying, # of "same" **can only be 1 or 3** -> NSA paid!
Otherwise, one of the cryptographers must have paid, but **impossible to tell who**!

60

60

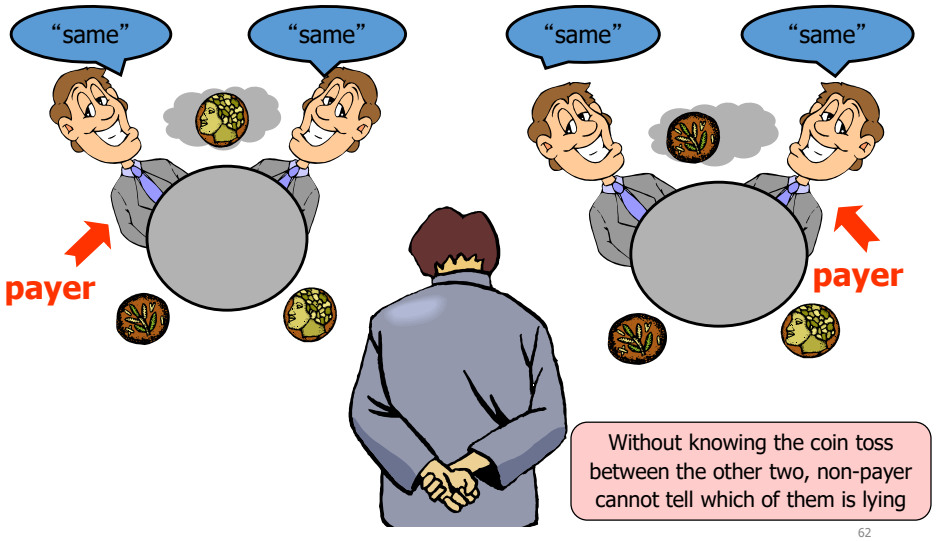
Non-Payer's View: Same Coins



61

61

Non-Payer's View: Different Coins



62

62

Super-posed Sending

- This idea generalizes to **any group of size N**
- For each bit of the message, every user generates 1 random bit and sends it to ONE neighbor
  - Every user learns 2 bits (his own and his neighbor's)
- Each user announces own bit XOR neighbor's bit
- Sender announces own bit XOR neighbor's bit XOR message bit
- XOR all announcements = message bit
  - Every randomly generated bit occurs in this sum twice (and is canceled by XOR), message bit occurs once

63

63

Android and DAC

- Restrict use of system facilities by apps
  - e.g., bluetooth, network, storage access
  - requires kernel modifications, "special" group IDs
- Isolate apps from each other
  - unique user and group ID per installed app
  - assigned to app processes and files
- Hardcoded, scattered "policy"

64

64

Security Enhanced (SE)Linux : What is it?

- Mandatory Access Control (MAC) for Linux
  - Defines and enforces a system-wide security policy
  - Over all processes, objects, and operations
  - Based on security labels
- Can confine flawed and malicious applications
  - Even ones that run as “root”/uid 0
- Can prevent privilege escalation

65

65

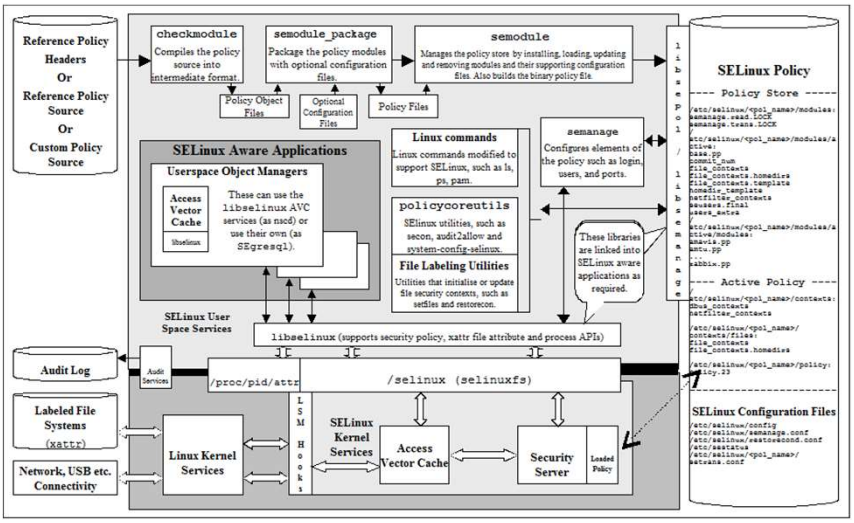
How Can SELinux Help Android?

- Confine privileged daemons
  - Protect them from misuse
  - Limit the damage that can be done via them
- Sandbox and isolate apps
  - Strongly separate apps from each other and from the system
  - Prevent privilege escalation by apps
- Provide centralized, analyzable policy

66

66

Security Enhanced (SE)Linux in Android



More details at:
http://selinuxproject.org/~jmorris/lss2011_slides/caseforseandroid.pdf

67

67