Lecture 15

- PKIs, certificates, and naming
  - X.509
  - PGP
- Policy composition approaches
- Noninterference
  - Access control matrix interpretation

Cryptographic Key Infrastructure

- Goal: bind identity to key
- Classical: not possible as all keys are shared
  - Use protocols to agree on a shared key (see earlier)
- Public key: bind identity to public key
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

Certificates

- Create token (message) containing
  - Identity of principal (here, Alice)
  - Corresponding public key
  - Timestamp (when issued)
  - Other information (perhaps identity of signer)

  signed by trusted authority (here, Cathy)

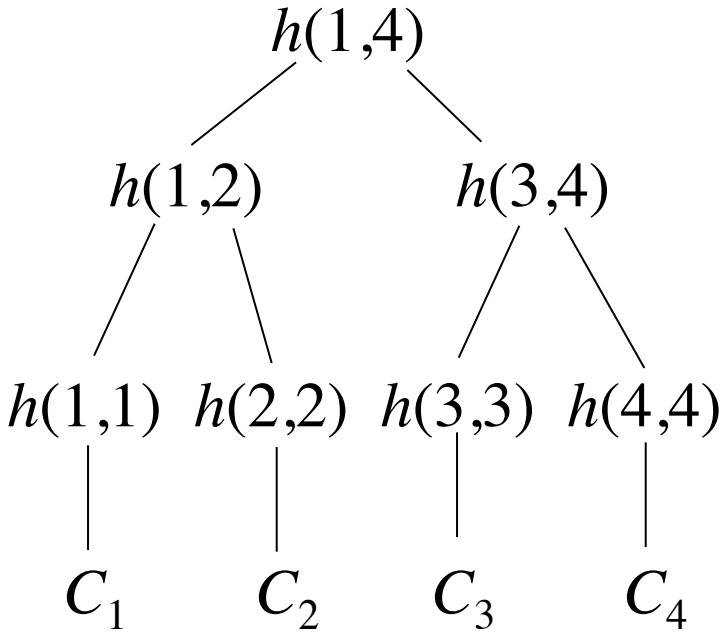
$$C_A = \{ e_A \,||\, \text{Alice} \,||\, T \}\, d_C$$

Use

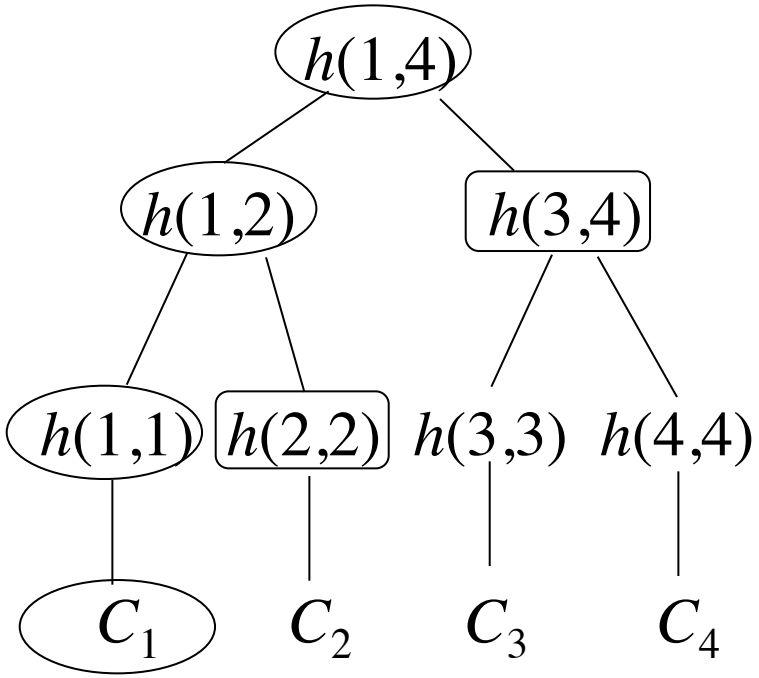
- Bob gets Alice's certificate
  - If he knows Cathy's public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key
- Problem: Bob needs Cathy's public key to validate certificate
  - Problem pushed "up" a level
  - Two approaches: Merkle's tree, signature chains

Merkle's Tree Scheme

- Keep certificates in a file
  - Changing any certificate changes the file
  - Use crypto hash functions to detect this
- Define hashes recursively
  - h is hash function
  - C_i is certificate i
- Hash of file ($h(1,4)$ in example) known to all



Validation



- To validate C_1:
  - Compute $h(1, 1)$
  - Obtain $h(2, 2)$
  - Compute $h(1, 2)$
  - Obtain $h(3, 4)$
  - Compute $h(1,4)$
  - Compare to known $h(1, 4)$
- Need to know hashes of children of nodes on path that are not computed

Details

- f: $D \times D \rightarrow D$ maps bit strings to bit strings
- h: $N \times N \rightarrow D$ maps integers to bit strings
  - if $i \geq j$, $h(i, j) = f(C_i, C_j)$
  - if $i < j$,

    $h(i, j) = f(h(i, \lfloor (i+j)/2 \rfloor), h(\lfloor (i+j)/2 \rfloor + 1, j))$

Problem

- File must be available for validation
  - Otherwise, can't recompute hash at root of tree
  - Intermediate hashes would do
- Not practical in most circumstances
  - Too many certificates and users
  - Users and certificates distributed over widely separated systems

Certificate Signature Chains

- Create certificate
  - Generate hash of certificate
  - Encipher hash with issuer's private key
- Validate
  - Obtain issuer's public key
  - Decipher enciphered hash
  - Recompute hash from certificate and compare
- Problem: getting issuer's public key

X.509 Chains

- Some certificate components in X.509v3:
  - Version
  - Serial number
  - Signature algorithm identifier: hash algorithm
  - Issuer's name; uniquely identifies issuer
  - Interval of validity
  - Subject's name; uniquely identifies subject
  - Subject's public key
  - Signature: enciphered hash

X.509 Certificate Validation

- Obtain issuer's public key
  - The one for the particular signature algorithm
- Decipher signature
  - Gives hash of certificate
- Recompute hash from certificate and compare
  - If they differ, there's a problem
- Check interval of validity
  - This confirms that certificate is current

Issuers

- *Certification Authority (CA)*: entity that issues certificates
  - Multiple issuers pose validation problem
  - Alice's CA is Cathy; Bob's CA is Don; how can Alice validate Bob's certificate?
  - Have Cathy and Don cross-certify
    - Each issues certificate for the other

Validation and Cross-Certifying

- Certificates:
  - Cathy<<Alice>>
  - Dan<<Bob>
  - Cathy<<Dan>>
  - Dan<<Cathy>>
- Alice validates Bob's certificate
  - Alice obtains Cathy<<Dan>>
  - Alice uses (known) public key of Cathy to validate Cathy<<Dan>>
  - Alice uses Cathy<<Dan>> to validate Dan<<Bob>>

PGP Chains

- OpenPGP certificates structured into packets
  - One public key packet
  - Zero or more signature packets
- Public key packet:
  - Version (3 or 4; 3 compatible with all versions of PGP, 4 not compatible with older versions of PGP)
  - Creation time
  - Validity period (not present in version 3)
  - Public key algorithm, associated parameters
  - Public key

OpenPGP Signature Packet

- Version 3 signature packet
  - Version (3)
  - Signature type (level of trust)
  - Creation time (when next fields hashed)
  - Signer's key identifier (identifies key to encipher hash)
  - Public key algorithm (used to encipher hash)
  - Hash algorithm
  - Part of signed hash (used for quick check)
  - Signature (enciphered hash)
- Version 4 packet more complex

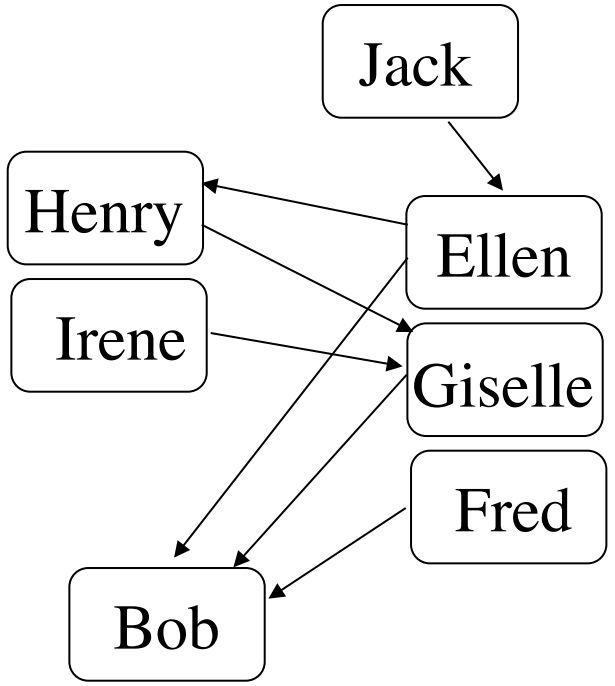
Signing

- Single certificate may have multiple signatures
- Notion of "trust" embedded in each signature
  - Range from "untrusted" to "ultimate trust"
  - Signer defines meaning of trust level (no standards!)
- All version 4 keys signed by subject
  - Called "self-signing"

Validating Certificates

- Alice needs to validate Bob's OpenPGP cert
  - Does not know Fred, Giselle, or Ellen
- Alice gets Giselle's cert
  - Knows Henry slightly, but his signature is at "casual" level of trust
- Alice gets Ellen's cert
  - Knows Jack, so uses his cert to validate Ellen's, then hers to validate Bob's

Arrows show signatures
Self signatures not shown



Naming and Certificates

- Certificates issued to a principal
  - Principal uniquely identified to avoid confusion
- Problem: names may be ambiguous
  - Does the name "Matt Bishop" refer to:
    - The author of this book?
    - A programmer in Australia?
    - A stock car driver in Muncie, Indiana?
    - Someone else who was named "Matt Bishop"

Disambiguating Identity

- Include ancillary information in names
  - Enough to identify principal uniquely
  - X.509v3 Distinguished Names do this
- Example: X.509v3 Distinguished Names
  - /O=University of California/OU=Davis campus/ OU=Department of Computer Science/CN=Matt Bishop/

    refers to the Matt Bishop (CN is *common name*) in the Department of Computer Science (OU is *organizational unit*) on the Davis Campus of the University of California (O is *organization*)

CAs and Policies

- Matt Bishop wants a certificate from Certs-from-Us
  - How does Certs-from-Us know this is "Matt Bishop"?
    - CA's *authentication policy* says what type and strength of authentication is needed to identify Matt Bishop to satisfy the CA that this is, in fact, Matt Bishop
  - Will Certs-from-Us issue this "Matt Bishop" a certificate once he is suitably authenticated?
    - CA's *issuance policy* says to which principals the CA will issue certificates

Example: Verisign CAs

- Class 1 CA issued certificates to individuals
  - Authenticated principal by email address
    - Idea: certificate used for sending, receiving email with various security services at that address
- Class 2 CA issued certificates to individuals
  - Authenticated by verifying user-supplied real name and address through an online database
    - Idea: certificate used for online purchasing

Example: Verisign CAs

- Class 3 CA issued certificates to individuals
  - Authenticated by background check from investigative service
    - Idea: higher level of assurance of identity than Class 1 and Class 2 CAs
- Fourth CA issued certificates to web servers
  - Same authentication policy as Class 3 CA
    - Idea: consumers using these sites had high degree of assurance the web site was not spoofed

Internet Certification Hierarchy

- Tree structured arrangement of CAs
  - Root is *Internet Policy Registration Authority*, or IPRA
    - Sets policies all subordinate CAs must follow
    - Certifies subordinate CAs (called *policy certification authorities*, or PCAs), each of which has own authentication, issuance policies
    - Does not issue certificates to individuals or organizations other than subordinate CAs
  - PCAs issue certificates to ordinary CAs
    - Does not issue certificates to individuals or organizations other than subordinate CAs
  - CAs issue certificates to organizations or individuals

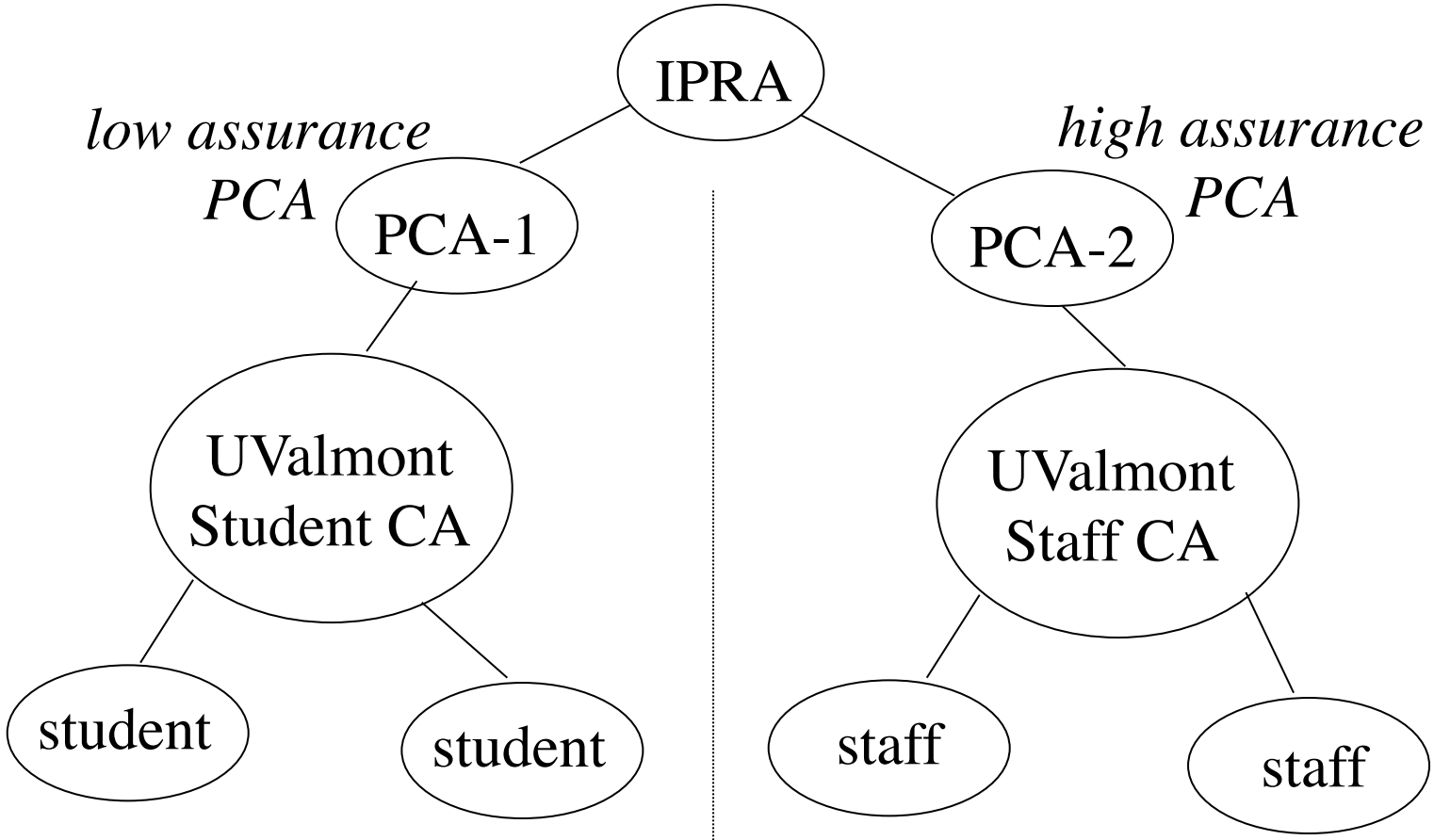
Example

- University of Valmont issues certificates to students, staff
  - Students must present valid reg cards (considered low assurance)
  - Staff must present proof of employment and fingerprints, which are compared to those taken when staff member hired (considered high assurance)

UValmont and PCAs

- First PCA: requires subordinate CAs to make good-faith effort to verify identities of principals to whom it issues certificates
  - Student authentication requirements meet this
- Second PCA: requires use of biometrics to verify identity
  - Student authentication requirements do not meet this
  - Staff authentication requirements do meet this
- UValmont establishes to CAs, one under each PCA above

UValmont and Certification Hierarchy



Certificate Differences

- Student, staff certificates signed using different private keys (for different CAs)
  - Student's signed by key corresponding to low assurance certificate signed by first PCA
  - Staff's signed by key corresponding to high assurance certificate signed by second PCA
- To see what policy used to authenticate:
  - Determine CA signing certificate, check its policy
  - Also go to PCA that signed CA's certificate
    - CAs are restricted by PCA's policy, but CA can restrict itself further

Types of Certificates

- Organizational certificate
  - Issued based on principal's affiliation with organization
  - Example Distinguished Name

    /O=University of Valmont/OU=Computer Science Department/CN=Marsha Merteuille/
- Residential certificate
  - Issued based on where principal lives
  - No affiliation with organization implied
  - Example Distinguished Name

    /C=US/SP=Louisiana/L=Valmont/PA=1 Express Way/ CN=Marsha Merteuille/

Certificates for Roles

- Certificate tied to a role
- Example
  - UValmont wants comptroller to have a certificate
    - This way, she can sign contracts and documents digitally
  - Distinguished Name

    /O=University of Valmont/OU=Office of the Big Bucks/RN=Comptroller

    where "RN" is *role name*; note the individual using the certificate is not named, so no CN

Certificate Principal Identifiers

- Need not be Distinguished Names
  - Example: PGP certificates usually have email addresses, not Distinguished Names
- Permits ambiguity, so the user of the certificate may not be sure to whom it refers
  - Email addresses change often, particularly if work email addresses used
- Problem: how do you prevent naming conflicts?

Naming Conflicts

- X.509v3, PGP silent
  - Assume CAs will prevent name conflicts as follows
    - No two distinct CAs have the same Distinguished Name
    - No two principals have certificates issued containing the same Distinguished Name by a single CA

Internet Certification Hierarchy

- In theory, none
  - IPRA requires each PCA to have a unique Distinguished Name
  - No PCA may certify two distinct CAs with same Distinguished Name
- In practice, considerable confusion possible!

Example Collision

- John Smith, John Smith Jr. live at same address
  - John Smith Jr. applies for residential certificate from Certs-from-Us, getting the DN of:

    /C=US/SP=Maine/L=Portland/PA=1 First Ave./ CN=John Smith/
  - Now his father applies for residential certificate from Quick-Certs, getting DN of:

    /C=US/SP=Maine/L=Portland/PA=1 First Ave./ CN=John Smith/

    because Quick-Certs has no way of knowing that DN is taken

Solutions

- Organizational certificates
  - All CA DNs must be superior to that of the principal
  - Example: for Marsha Merteuille's DN:

    /O=University of Valmont/OU=Computer Science Department/CN=Marsha Merteuille/

    DN of the CA must be either:

    /O=University of Valmont/

    (the issuer being the University) or

    /O=University of Valmont/OU=Computer Science Department/

    (the issuer being the Department)

Solutions

- Residential certificates
  - DN collisions explicitly allowed (in above example, no way to force disambiguation)

    /C=US/SP=Maine/L=Portland/PA=1 First Ave./CN=John Smith/

    Unless names of individuals are different, how can you force different names in the certificates?

Related Problem

- Single CA issues two types of certificates under two different PCAs
- Example
  - UValmont issues both low assurance, high assurance certificates under two different PCAs
  - How does validator know under which PCA the certificate was issued?
    - Reflects on assurance of the identity of the principal to whom certificate was issued

Solution

- CA Distinguished Names need *not* be unique
- CA (Distinguished Name, public key) pair *must* be unique
- Example
  - In earlier UValmont example, student validation required using first PCA's public key; validation using second PCA's public key would fail
  - Keys used to sign certificate indicate the PCA, and the policy, under which certificate is issued

Meaning of Identity

- Authentication validates identity
  - CA specifies type of authentication
  - If incorrect, CA may misidentify entity unintentionally
- Certificate binds *external* identity to crypto key and Distinguished Name
  - Need confidentiality, integrity, anonymity
    - Recipient knows same entity sent all messages, but *not* who that entity is

Persona Certificate

- Certificate with meaningless Distinguished Name
  - If DN is

    /C=US/O=Microsoft Corp./CN=Bill Gates/

    the real subject may not (or may) be Mr. Gates
  - Issued by CAs with persona policies under a PCA with policy that supports this
- PGP certificates can use any name, so provide this implicitly

Example

- Government requires all citizens with gene X to register
  - Anecdotal evidence people with this gene become criminals with probability 0.5.
  - Law to be made quietly, as no scientific evidence supports this, and government wants no civil rights fuss
- Government employee wants to alert media
  - Government will deny plan, change approach
  - Government employee will be fired, prosecuted
- Must notify media anonymously

Example

- Employee gets persona certificate, sends copy of plan to media
  - Media knows message unchanged during transit, but not who sent it
  - Government denies plan, changes it
- Employee sends copy of new plan signed using same certificate
  - Media can tell it's from original whistleblower
  - Media cannot track back whom that whistleblower is

Trust

- Goal of certificate: bind correct identity to DN
- Question: what is degree of assurance?
- X.509v3, certificate hierarchy
  - Depends on policy of CA issuing certificate
  - Depends on how well CA follows that policy
  - Depends on how easy the required authentication can be spoofed
- Really, estimate based on the above factors

Example: Passport Required

- DN has name on passport, number and issuer of passport
- What are points of trust?
  - Passport not forged and name on it not altered
  - Passport issued to person named in passport
  - Person presenting passport is person to whom it was issued
  - CA has checked passport and individual using passport

PGP Certificates

- Level of trust in signature field
- Four levels
  - Generic (no trust assertions made)
  - Persona (no verification)
  - Casual (some verification)
  - Positive (substantial verification)
- What do these mean?
  - Meaning not given by OpenPGP standard
  - Signer determines what level to use
  - Casual to one signer may be positive to another

Back to Policies . . .

- Policy composition

Same Policies

- If we can change policies that components must meet, composition is trivial (as above)
- If we *cannot*, we must show composition meets the same policy as that of components; this can be very hard

Different Policies

- What does "secure" now mean?
- Which policy (components) dominates?
- Possible principles:
  - Any access allowed by policy of a component must be allowed by composition of components (*autonomy*)
  - Any access forbidden by policy of a component must be forbidden by composition of components (*security*)

Implications

- Composite system satisfies security policy of components as components' policies take precedence
- If something neither allowed nor forbidden by principles, then:
  - Allow it (Gong & Qian)
  - Disallow it (Fail-Safe Defaults)

Example

- System X: Bob can't access Alice's files
- System Y: Eve, Lilith can access each other's files
- Composition policy:
  - Bob can access Eve's files
  - Lilith can access Alice's files
- Question: can Bob access Lilith's files?

Solution (Gong & Qian)

- Notation:
  - (a, b): a can read b's files
  - AS(x): access set of system x
- Set-up:
  - AS(X) = ∅
  - AS(Y) = { (Eve, Lilith), (Lilith, Eve) }
  - AS(X∪Y) = { (Bob, Eve), (Lilith, Alice),
    (Eve, Lilith), (Lilith, Eve) }

Solution (Gong & Qian)

- Compute transitive closure of AS(X∪Y):
  - $AS(X \cup Y)^+$ = { (Bob, Eve), (Bob, Lilith), (Bob, Alice),
    (Eve, Lilith), (Eve, Alice),
    (Lilith, Eve), (Lilith, Alice) }
- Delete accesses conflicting with policies of components:
  - Delete (Bob, Alice)
- (Bob, Lilith) in set, so Bob can access Lilith's files

Idea

- Composition of policies allows accesses not mentioned by original policies
- Generate all possible allowed accesses
  - Computation of transitive closure
- Eliminate forbidden accesses
  - Removal of accesses disallowed by individual access policies
- Everything else is allowed
- Note: determining if access allowed is of polynomial complexity

Interference

- Think of it as something used in communication
  - Holly/Lara example: Holly interferes with the CPU utilization, and Lara detects it—communication
- Plays role of writing (interfering) and reading (detecting the interference)

Model

- System as state machine
  - Subjects $S = \{ s_i \}$
  - States $\Sigma = \{ \sigma_i \}$
  - Outputs $O = \{ o_i \}$
  - Commands $Z = \{ z_i \}$
  - State transition commands $C = S \times Z$
- Note: no inputs
  - Encode either as selection of commands or in state transition commands

Functions

- State transition function T: $C \times \Sigma \rightarrow \Sigma$
  - Describes effect of executing command c in state σ
- Output function P: $C \times \Sigma \rightarrow O$
  - Output of machine when executing command c in state σ
- Initial state is σ_0

Example

- Users Heidi (high), Lucy (low)
- 2 bits of state, *H* (high) and *L* (low)
  - System state is (H, L) where H, L are 0, 1
- 2 commands: xor_0, xor_1 do xor with 0, 1
  - Operations affect *both* state bits regardless of whether Heidi or Lucy issues it

Example: 2-bit Machine

- S = { Heidi, Lucy }
- Σ = { (0,0), (0,1), (1,0), (1,1) }
- C = { xor_0, xor_1 }

	Input States (H, L)			
	(0,0)	(0,1)	(1,0)	(1,1)
xor0	(0,0)	(0,1)	(1,0)	(1,1)
xor1	(1,1)	(1,0)	(0,1)	(0,0)

Outputs and States

- T is inductive in first argument, as

  $T(c_0, \sigma_0) = \sigma_1$; $T(c_{i+1}, \sigma_{i+1}) = T(c_{i+1}, \mathrm{T}(c_i, \sigma_i))$
- Let C^* be set of possible sequences of commands in C
- T^*: $C^* \times \Sigma \rightarrow \Sigma$ and

  $c_s = c_0 \ldots c_n \Rightarrow T^*(c_s, \sigma_i) = T(c_n, \ldots, T(c_0, \sigma_i) \ldots)$
- P similar; define P^* similarly

Projection

- $T^*(c_s,\sigma_i)$ sequence of state transitions
- $P^*(c_s,\sigma_i)$ corresponding outputs
- *proj*(s, c_s, σ_i) set of outputs in $P^*(c_s,\sigma_i)$ that subject *s* authorized to see
  - In same order as they occur in $P^*(c_s,\sigma_i)$
  - Projection of outputs for *s*
- Intuition: list of outputs after removing outputs that *s* cannot see

Purge

- $G \subseteq S$, G a group of subjects
- $A \subseteq Z$, A a set of commands
- $\pi_G(c_s)$ subsequence of c_s with all elements (s,z), $s \in G$ deleted
- $\pi_A(c_s)$ subsequence of c_s with all elements (s,z), $z \in A$ deleted
- $\pi_{G,A}(c_s)$ subsequence of c_s with all elements (s,z), $s \in G$ and z $\in A$ deleted

Example: 2-bit Machine

- Let $\sigma_0 = (0,1)$
- 3 commands applied:
  - Heidi applies xor_0
  - Lucy applies xor_1
  - Heidi applies xor_1
- $c_s = ((\text{Heidi}, xor_0), (\text{Lucy}, xor_1), (\text{Heidi}, xor_0))$
- Output is 011001
  - Shorthand for sequence (0,1)(1,0)(0,1)

Example

- $proj(\text{Heidi}, c_s, \sigma_0) = 011001$
- $proj(\text{Lucy}, c_s, \sigma_0) = 101$
- $\pi_{\text{Lucy}}(c_s) = ((\text{Heidi}, xor_0), (\text{Heidi}, xor_1))$
- $\pi_{\text{Lucy},xor1}(c_s) = ((\text{Heidi}, xor_0), (\text{Heidi}, xor_1))$
- $\pi_{\text{Heidi}}(c_s) = ((\text{Lucy}, xor_1))$

Example

- $\pi_{\text{Lucy},xor0}(c_s) = ((\text{Heidi}, xor_0), (\text{Lucy}, xor_1), (\text{Heidi}, xor_1))$
- $\pi_{\text{Heidi},xor0}(c_s) = \pi_{xor0}(c_s) = ((\text{Lucy}, xor_1), (\text{Heidi}, xor_1))$
- $\pi_{\text{Heidi}, xor1}(c_s) = ((\text{Heidi}, xor_0), (\text{Lucy}, xor_1))$
- $\pi_{xor1}(c_s) = ((\text{Heidi}, xor_0))$

Noninterference

- Intuition: Set of outputs Lucy can see corresponds to set of inputs she can see, there is no interference
- Formally: $G, G' \subseteq \mathrm{S}, G \neq G'$; $A \subseteq Z$; Users in G executing commands in A are *noninterfering* with users in G' iff for all $c_s \in C^*$, and for all $s \in G'$,

$$proj(s, c_s, \sigma_i) = proj(s, \pi_{G,A}(c_s), \sigma_i)$$

  - Written A,G :| G'