Enforcing Information Flow Policies Through Chain- And Tree-based Enforcement Schemes

Mark Jones

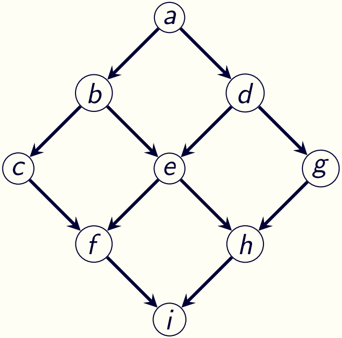
Royal Holloway, 7 Jan 2017

Joint research with...

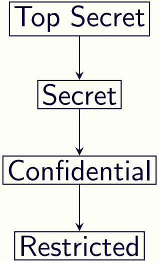
- Gregory Gutin 
- Jason Crampton (Information Security Group, RHUL) 
- Naomi Farley (ISG, RHUL) 
- Bertram Poettering (Ruhr University Bochum) 

Information Flow Policies

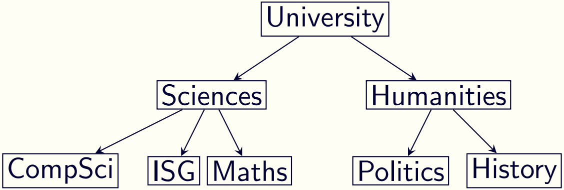
- **Information Flow Policy:** A partially ordered set $(X, \leqslant)$ of *security labels*, set U of *users*, set R of *resources*, labelling $\lambda : U \cup R \to X$;
- User u is authorized to access resource r iff $\lambda(u) \geqslant \lambda(r)$.
- We represent $(X, \leqslant)$ with the Hasse diagram (minimal acyclic digraph s.t. there is a path from x to y iff $x \geqslant y$).



Example: Security Clearance



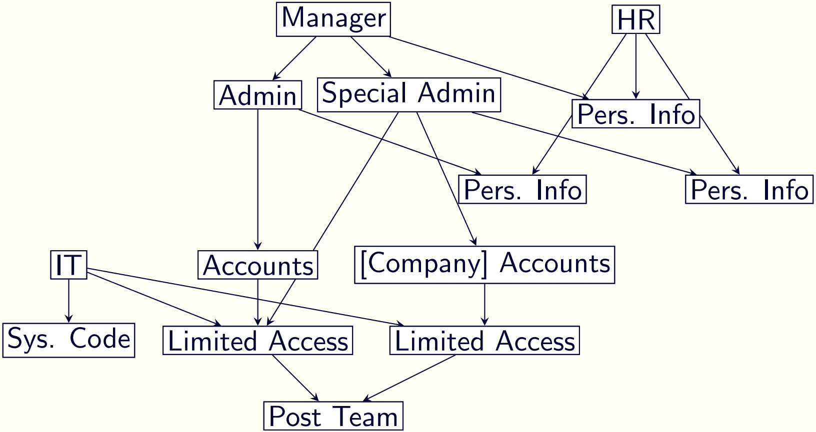
Example: University Departments



Example: The Real World

"There's the post team, who have the least access to anything, then there's the admin teams to have access to all the client accounts except the ones for [company name]. The accountss for [company name] are handled by a special admin, who has limited access to the other accounts. There's the managers, who have access to everything the admin teams have access to. There's IT, who have limited access to all the accounts and also have access to the system code. And then there's HR, who have access to everyone else's info, but don't have access to the accounts."

Example: The Real World



Cryptographic Enforcement schemes

- A *Cryptographic Enforcement Scheme* is a way of enforcing an information flow policy.
- Each node in $x \in X$ is assigned a key $\kappa(x)$.
- Each user $u \in U$ is given secret information $\sigma(x)$.
- We may also publish a set *Pub* of *public information*.
- κ and σ are constructed such that u can derive $\kappa(x)$ iff $u \geqslant x$.
- (i.e. there exists an algorithm to construct $\kappa(x)$ given $\sigma(u)$ and *Pub*)

Enforcement Scheme: give everybody all the keys

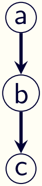
- Simplest solution:
- Assign keys randomly.
- $\forall u \in U$ set $\sigma(u) = \{\kappa(x) : \lambda(u) \geq x\}$.
- Problem: each user recieves a large amount of secret information.

Enforcement Scheme: derivation through publically labelled edges

- We can use public information to reduce amount of secret information:
- Assign keys randomly.
- $\forall u \in U$ set $\sigma(u) = \{\kappa(\lambda(u))\}$.
- For each edge xy in Hasse diagram, encrypt $\kappa(y)$ using $\kappa(x)$ and add it to Pub
- Then for any path $\lambda(u) = x_1 x_2 \ldots x_t$, u derives $\kappa(x_2)$ using $\kappa(x_1)$, then $\kappa(x_3)$ using $\kappa(x_2)$...
- Problem: large amount of public information.

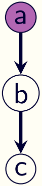
Simple enforcement scheme: single chain

- We are interested in minimizing the amount of secret information without relying on public information.
- Suppose $(X \leqslant)$ is a chain (i.e. linear order) $x_1 \geqslant x_2 \geqslant \cdots \geqslant x_n$.
- There exists an information flow policy with no public information and minimal secret information, as follows.
- Assign $\kappa(x_1)$ randomly.
- Using a **PRF (pseudorandom function)** $\mathcal{F}$, assign $\kappa(x_{i+1}) = \mathcal{F}(\kappa(x_i))$.
- $\forall u \in U$ set $\sigma(u) = \kappa(x_i)$ where $x_i = \lambda(x_i)$.



$\kappa(a) = \text{randomly generated}$
$\kappa(b) = \mathcal{F}(\kappa(a))$
$\kappa(c) = \mathcal{F}(\kappa(b))$

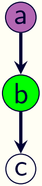
$\kappa(a)$

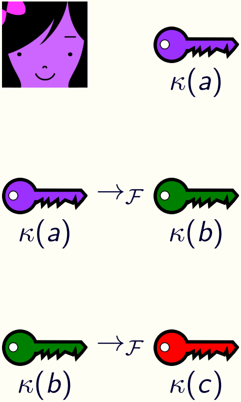


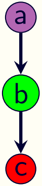
$\kappa(a)$

 $\rightarrow_{\mathcal{F}}$ 

$\kappa(a)$ $\kappa(b)$



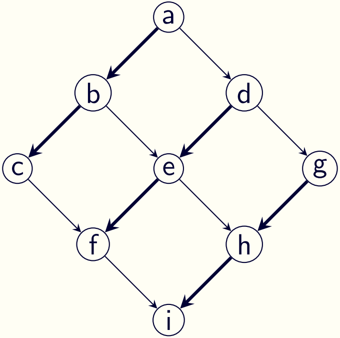

$\kappa(a)$
$\kappa(a) \rightarrow_{\mathcal{F}} \kappa(b)$
$\kappa(b) \rightarrow_{\mathcal{F}} \kappa(c)$

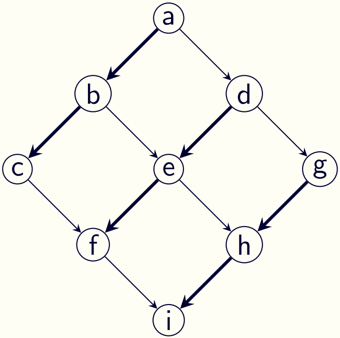

a
b
c

- In general $(X, \leqslant)$ is not a chain.
- However, we can extend this idea by partitioning X into chains.

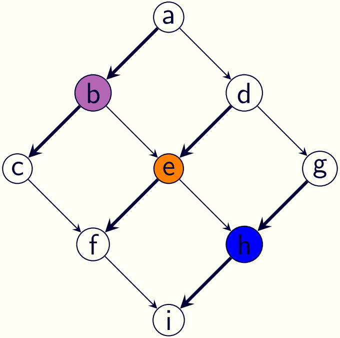
Chain-Based Enforcement Schemes

- Partition X into a set of chains (i.e. directed paths)
- For each maximal element x_1 of a chain; assign $\kappa(x_1)$ randomly.
- For a chain $x_1 \geqslant x_2 \geqslant \cdots \geqslant x_n$, assign $\kappa(x_{i+1}) = \mathcal{F}(\kappa(x_i))$.
- $\forall u \in U$ set $\sigma(u) = \{\kappa(y) : y$ is the maximal element in its chain s.t. $x \geqslant y\}$.





$\kappa(a) =$ randomly generated, $\kappa(b) = \mathcal{F}(\kappa(a)), \kappa(c) = \mathcal{F}(\kappa(b))$
$\kappa(d) =$ randomly generated, $\kappa(e) = \mathcal{F}(\kappa(d)), \kappa(f) = \mathcal{F}(\kappa(e))$
$\kappa(g) =$ randomly generated, $\kappa(h) = \mathcal{F}(\kappa(g)), \kappa(i) = \mathcal{F}(\kappa(h))$



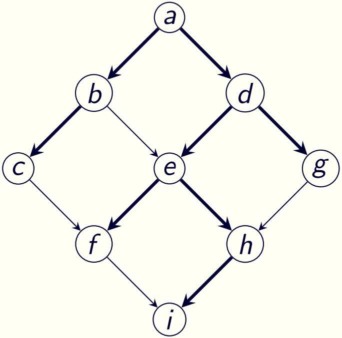
$\kappa(a)$ $\kappa(e)$ $\kappa(h)$

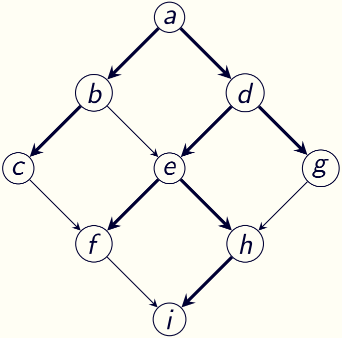
Generalizing Chain-Based Enforcement Schemes

- Key idea: key for a lower element is determined by its parent.
- We can generalize this idea to forests instead of chains.

Forest-Based Enforcement Schemes

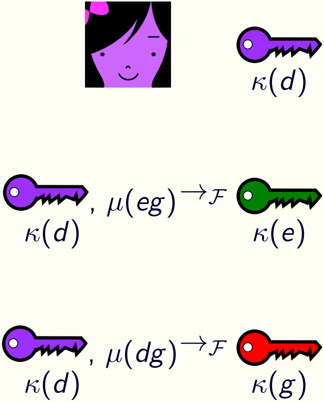
- Partition X into a set of out-trees.
- For each root r of a tree, assign $\kappa(r)$ randomly.
- Let each arc e in the forest have a (fixed) label $\mu(e)$.
- For each arc xy in the forest, assign $\kappa(y) = \mathcal{F}(\kappa(x), \mu(e))$
- $\forall u \in U$ set $\sigma(u) = \{\kappa(y) : y$ is a maximal element in its tree s.t. $x \geqslant y\}$

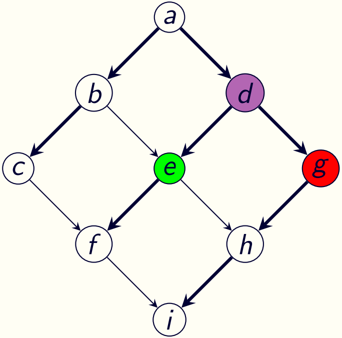


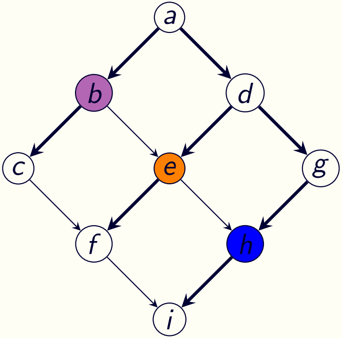


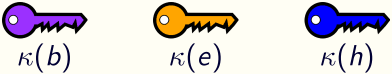
$\kappa(a) =$ randomly generated

$\kappa(b) = \mathcal{F}(a, \mu(ab)), \kappa(c) = \mathcal{F}(b, \mu(bc)), \kappa(d) = \mathcal{F}(a, \mu(ad)), \ldots$


$\kappa(d)$
$\kappa(d)$, $\mu(eg)^{\to\mathcal{F}}$ $\kappa(e)$
$\kappa(d)$, $\mu(dg)^{\to\mathcal{F}}$ $\kappa(g)$


a
b
d
c
e
g
f
h
i


a
b
d
c
e
g
f
h
i


κ(b)
κ(e)
κ(h)

Question

- Give an enforcement scheme, $\sum_{u \in U} |\sigma(u)|$ denotes the total number of keys assigned to users.
- Given an information flow policy $(X, \leqslant), U, R, \lambda$, what choice of forest/chain partition minimizes $\sum_{u \in U} |\sigma(u)|$?

Assigning weights to edges

- **Assumption:** In what follows, we assume $(X, \leqslant)$ has a maximal element *root* and any forest-based scheme uses a single tree rooted at *root*.
- For any arc yz in the Hasse diagram of $(X, \leqslant)$, set $\gamma(yz) = \{u \in U : x \geqslant z, x \not\geqslant y$, where $x = \lambda(u)\}$.
- Let $w(yz) = |\gamma(yz)|$
- **Claim:** For any out-tree F,

$$\sum_{yz \in A(F)} w(yz) + |\sigma^{-1}(root) \cap U| = \sum_{u \in U} |\sigma(u)|$$

Assigning weights to edges

- $w(yz) = |\{u \in U : \lambda(u) \geqslant z, \lambda(u) \ngeqslant y|$
- **Claim:** For any out-tree F,

$$\sum_{yz \in A(F)} w(yz) + |\sigma^{-1}(root) \cap U| = \sum_{u \in U} |\sigma(u)|$$

- Proof: $\forall x \in U, z \in X \setminus \{root\}$, set $\chi(u, z) = 1$ if $\lambda(u) \geqslant z$, $\lambda(u) \ngeqslant y$, where $y =$ parent of z in F.
  Then
  $\sum_{yz \in A(F)} w(yz) + |\sigma^{-1}(root) \cap U|$
  $= \sum_{z \in X \setminus \{root\}} \sum_{u \in U} \chi(u, z) + |\sigma^{-1}(root) \cap U|$
  $= \sum_{u \in U} |\sigma(u)|$

Assigning weights to edges

- $w(yz) = |\{u \in U : \lambda(u) \geqslant z, \lambda(u) \ngeqslant y|$
- **Claim:** For any out-tree F,

$$\sum_{yz \in A(F)} w(yz) + |\sigma^{-1}(root) \cap U| = \sum_{u \in U} |\sigma(u)|$$

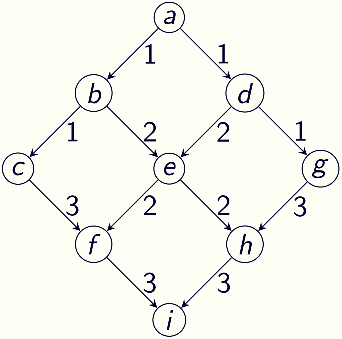
- Proof: $\forall x \in U, z \in X \setminus \{root\}$, set
  $\chi(u, z) = 1$ if $\lambda(u) \geqslant z$, $\lambda(u) \ngeqslant y$,
  where $y =$ parent of z in F.
  Then
  $\sum_{yz \in A(F)} w(yz) + |\sigma^{-1}(root) \cap U|$
  $= \sum_{z \in X \setminus \{root\}} \sum_{u \in U} \chi(u, z) + |\sigma^{-1}(root) \cap U|$
  $= \sum_{u \in U} |\sigma(u)|$

Finding a solution

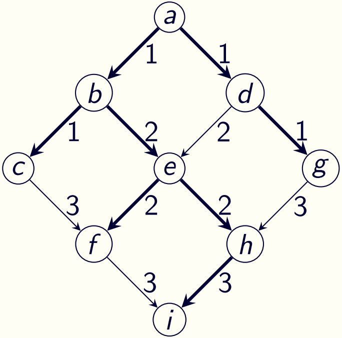
- $\sum_{yz \in A(F)} w(yz) + |\sigma^{-1}(root) \cap U| = \sum_{u \in U} |\sigma(u)|$
- Therefore to minimize $\sum_{u \in U} |\sigma(u)|$ it is enough to find a spanning out-forest with minimum weights on edges
- This can be done by choosing the minimum-weight in-arc for each vertex.

Theorem

Given an information flow policy$((X, \leqslant), U, R, \lambda)$, there exists a polynomial-time algorithm to find the forest-based assignment scheme that distributes a minimum number of keys.



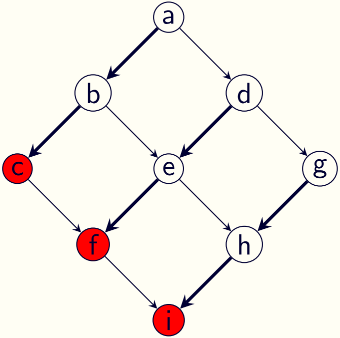
a
1
1
b
d
1
2
2
1
c
e
g
3
2
2
3
f
h
3
3
i

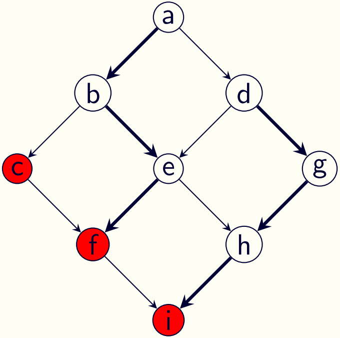

a
b
d
c
e
g
f
h
i
1
1
1
2
2
1
3
2
2
3
3
3

Chain-based enforcement schemes

- For some situations, we still prefer a chain-based enforcement scheme.
- The same trick of assigning weights to edges will work, however we cannot choose in-arcs independently.
- We can solve the problem using a flow network, provided we know the desired flow (i.e. number of chains)

Value of solution depends only on minimal elements





Lemma

Let $b_1, \ldots b_r$ be the bottom elements of each chain in a given partition. Then $\sum_{u \in U} |\sigma(u)| = \sum_i |\{u \in U : \lambda(u) \geq b_i\}|$

- Proof idea: Every element in $\{u \in U : \lambda(u) \geq b_i\}$ counts towards $w(yz)$ for exactly one edge yz in the chain containing b_i.

Value of solution depends only on minimal elements

Lemma

Let $b_1, \ldots b_r$ be the bottom elements of each chain in a given partition. Then $\sum_{u \in U} |\sigma(u)| = \sum_i |\{u \in U : \lambda(u) \geq b_i\}|$

- Let w be (width) of $(X, \leqslant)$, i.e. $\max\{|Y| :$ every pair of elements in $Y \subseteq X$ are incomparable$\}$ = min number of chains in a chain partition of $(X, \leqslant)$.
- For any chain partition of $\mathcal{C}$ of $(X, \leqslant)$, there exists a chain partition $\mathcal{C}'$ of $(X, \leqslant)$ with w chains s.t. bottom elements of $\mathcal{C}' \subseteq$ bottom elements of $\mathcal{C}$.

Theorem

Given an information flow policy$((X, \leqslant), U, R, \lambda)$ with width w, there exists chain-based assignment scheme that distributes a minimum number of keys and has exactly w chains, and such a scheme can be found in polynomial time.

Conclusion

- Out-forest based Cryptographic Enforcement Schemes: An efficient way to enforce information flow policies.
- Enforcement schemes are determined entirely by the forest.
- Polynomial-time algorithms for optimal (in terms of number of keys) **forest**-based schemes and **chain**-based schemes.
- Cost of a chain-based scheme determined by minimal elements; polynomial time algorithm for optimal chain-based scheme with minium number of chains.
- Open question: minimizing the maximu number of secrets per user?

The End

- Thank you for listening!
- Happy Birthday Gregory!