Access Control & Privacy Preservation in Online Social Networks

Feb. 22, 2013
CS6393 Lecture 6

Yuan Cheng
Institute for Cyber Security
University of Texas at San Antonio
ycheng@cs.utsa.edu
http://www.my.cs.utsa.edu/~ycheng

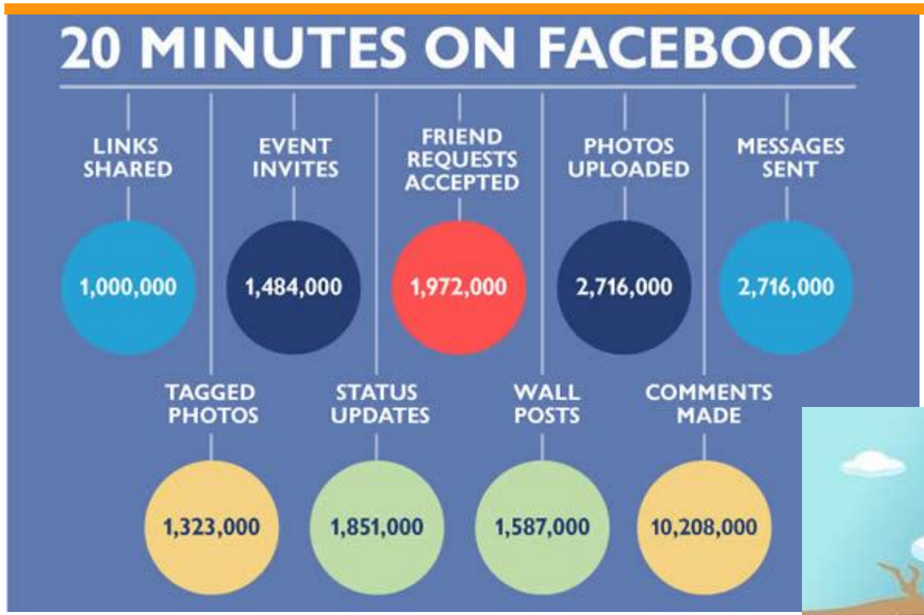
Outline

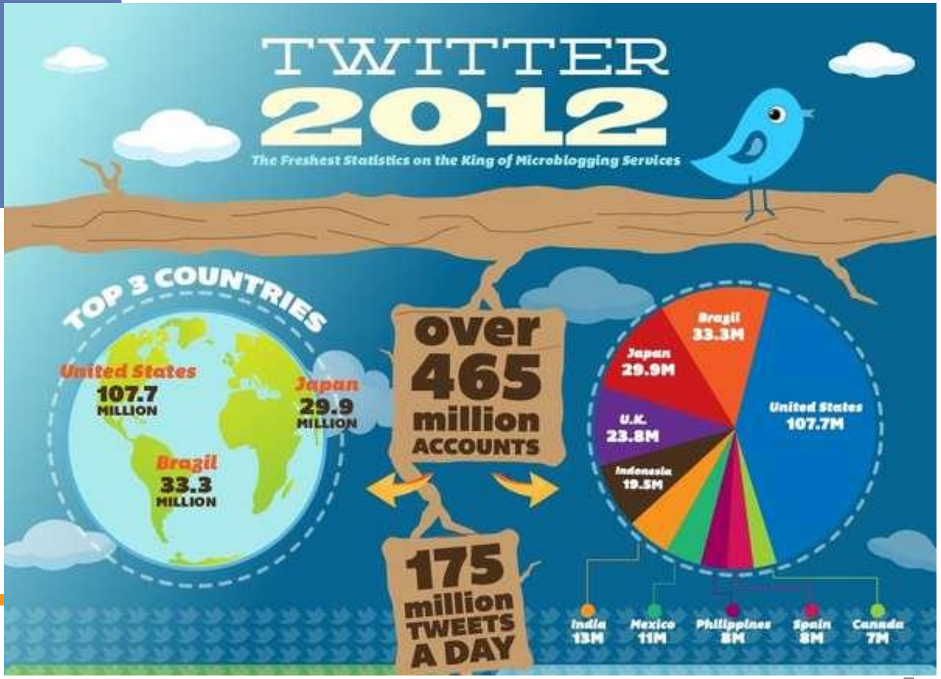
- Introduction
- Security & Privacy Issues in OSNs
- Access Control for OSNs
- Other Privacy Preservation Solutions

Online Social Networks

Statistics

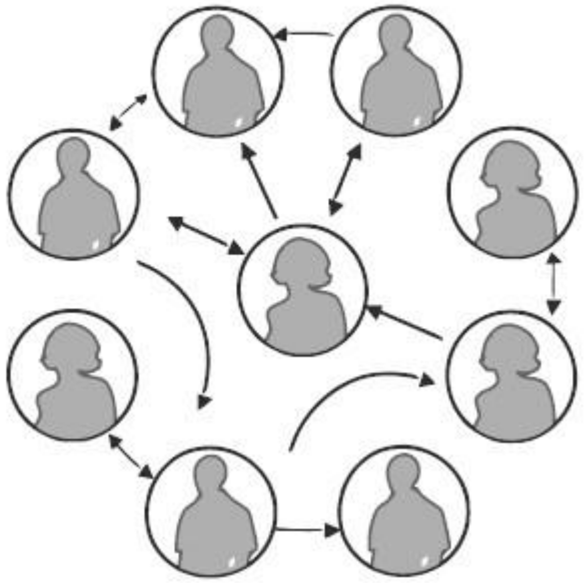
- Facebook, the largest OSN:
  - More than a billion monthly active users as of December 2012.
  - Approximately 82% of our monthly active users are outside the U.S. and Canada.
  - 618 million daily active users on average in December 2012.
  - 680 million monthly active users who used Facebook mobile products as of December 31, 2012.


20 MINUTES ON FACEBOOK
LINKS SHARED
1,000,000
EVENT INVITES
1,484,000
FRIEND REQUESTS ACCEPTED
1,972,000
PHOTOS UPLOADED
2,716,000
MESSAGES SENT
2,716,000
TAGGED PHOTOS
1,323,000
STATUS UPDATES
1,851,000
WALL POSTS
1,587,000
COMMENTS MADE
10,208,000


TWITTER 2012
The Freshest Statistics on the King of Microblogging Services
TOP 3 COUNTRIES
United States 107.7 MILLION
Japan 29.9 MILLION
Brazil 33.3 MILLION
over 465 million ACCOUNTS
175 million TWEETS A DAY
Brazil 33.3M
Japan 29.9M
United States 107.7M
U.K. 23.8M
Indonesia 19.5M
India 13M
Mexico 11M
Philippines 8M
Spain 8M
Canada 7M

Representation of an OSN

- An OSN is represented by means of a graph
  - Users are denoted as nodes
  - Relationships are represented as edges
    - Edges may be labeled to represent types
    - Edges may be directed



Outline

- Introduction
- Security & Privacy Issues in OSNs
- Access Control for OSNs
- Other Privacy Preservation Solutions

Security & Privacy Issues

- Security issues in OSNs can be organized into four categories
  - Privacy breaches
  - Spam and phishing attacks
  - Sybil attacks
  - Malware attacks
- Privacy breaches
  - Easy to happen from OSN providers, other users, and 3rd party applications
  - OSN providers store user data
  - 3rd party applications provide extra functionalities
  - Major threats are from *peer users*
    - Not aware of who they share with and how much
    - Have difficulty in managing privacy controls

Why Privacy is Hard to Protect

- Users tend to give out too much information
  - Unaware of privacy issues
  - Promote sharing vs. Protect privacy
- Users tend to be Reactive rather than Proactive
- Privacy policies
  - Changing over time
  - Confusing
  - Privacy thresholds vary by individuals

Outline

- Introduction
- Security & Privacy Issues in OSNs
- Access Control for OSNs
- Other Privacy Preservation Solutions

Control on Social Interactions

- A user wants to control other users' access to her own shared information
  - Only friends can read my post
- A user wants to control other users' activities who are related to the user
  - My children cannot be a friend of my co-workers
  - My activities should not be notified to my co-workers
- A user wants to control her outgoing/incoming activities
  - No accidental access to violent contents
  - Do not poke me
- A user's activity influences access control decisions
  - Once Alice sends a friend request to Bob, Bob can see Alice's profile

What existing OSNs offer

- Many OSNs allow users to choose from a pre-defined policy vocabulary
  - "public", "private", "friend", "friend of friend",…
- Some systems support customized relationships
  - circle, friend list
- Either too restrictive or too loose!

The Challenges of OSN Access Control

- Lack of a Central Administrator
  - Traditional access control mechanisms, such as RBAC, requires an administrator to manage access control
  - No such administrator exists in OSNs
- Dynamic Changing Environment
  - Frequent content updates and volatile nature of relationships
  - Identity and attribute-based access control are not scalable for OSNs

Relationship-based Access Control

- Users in OSNs are connected by social relationships (user-to-user relationships)
- Owner of the resource can control its release based on such relationships between the access requester and the owner

Related Works

- Fong et al. [*ESORICS 09*]
- Fong et al. [*CODASPY 11*]
- Carminati et al. [*ACM TISS 08*]
- Carminati et al. [*SACMAT 09*]

Fong et al. 11

- Relationship-Based Access Control: Protection Model and Policy Language
- Features:
  - Poly-relational, in the sense that it tracks not only whether a relationship exists, but also the type of that relationship
  - Authorization decision is solely based on the relationship between owner and accessor
  - A tree-shaped hierarchy of Access Contexts, which supports the scoping of the effectiveness of relationships

Fong 11: Policy Examples

- Grant access to the owner’s spouse
  - <spouse> a
- Grant access to the owner’s child
  - <-parent> a
- Grant access to grand parents
  - <parent><parent> a
- Grant access to parents, aunts and uncles
  - <parent> a V <parent><sibling> a V <parent><sibling><spouse> a

Fong 11: Policy Examples (cont.)

- Grant access unless the accessor is a parent of the owner
  - $\neg\langle parent\rangle\, a$
- Grant access to a sibling who is not married
  - $\langle sibling\rangle(a \wedge [spouse]\, \bot)$
- Grant access to a married sibling
  - $\langle sibling\rangle(a \wedge [spouse]\, \top)$
- Grant access if accessor is the only child of the owner
  - $\langle -parent\rangle\, a \wedge [-parent]\, a$

Carminati et al. 08

- Features
  - Discretionary
  - Rule-based
  - Semi-decentralized
- Policies are specified in terms of:
  - Relationship Types
  - Depth (Maximum length of the path)
  - Trust Levels (Minimum trust level)

C08: Approach

- Requestor must prove to the resource's owner that he/she satisfies the requirement stated in access control policy
  - Requestor sends access request to resource owner
  - Owner replies by sending access rules
  - Requestor provide the owner with a proof
  - Owner locally verifies the proof by a reasoner
  - Owner grants or rejects access.

C08: Trust Representation

- A trust relationship is usually modeled as a *directed* edge
- Trust relationship is *transitive*
  - We can use trust paths ABC to determine how much A considers C trustworthy

C08: Trust Computation

- Variant of the TidalTrust [Golbeck 2005]
  - 1: all the shortest paths are discovered
  - 2: set a trust threshold maxT, which is used to discard trust paths consisting of edges with a trust value less than maxT
  - 3: trust is computed by considering only the paths with a strength >= maxT

$$\mathrm{T}_{v,s} = \frac{\sum_{u \in N | \mathrm{T}_{v,u} \geq \max \mathrm{T}} \mathrm{T}_{v,u} \mathrm{T}_{u,s}}{\sum_{u \in N | \mathrm{T}_{v,u} \geq \max \mathrm{T}} \mathrm{T}_{v,u}}$$

C08: How Trust Works

- Trustworthiness of the proof
  - Relationship certificates
  - Certificate path -– a set of certificates
  - Certificate server –- a trusted third party
- Why is certificate server needed?
  - The requestor may maliciously omit one or more of the paths, providing only the paths with the highest level of trust
- The server stores into a central certificate directory all the relationship certificates specified by OSN nodes, and discovers certificates paths

C08: Trust-based Access Control

- Pros
  - We do it in reality
  - Requires little user input
- Cons
  - The concept of *trust* is complex and vague
  - Lacks of a standard measurement

Carminati et al. 09

- A Semantic Web Based Framework for Social Network Access Control
- Motivations:
  - Most of existing OSNs:
    - Implement very basic access control systems, by marking a given item as public, private, accessible by direct contacts, or some variants of this kind of setting.
    - Lack flexibility
    - Platform-specific

C09: The Idea

- Encode social network-related information by means of an ontology
  - User Profiles, Relationships among users, Resources, Relationships between users and resources, Actions
  - Construct the Social Network Knowledge Base (SNKB)
- Define security policies as rules
  - Encode authorizations to obtain the Security Authorization Knowledge Base (SAKB)
- Use a centralized reference monitor to enforce the policies

C09: Security Policies

- Access Control Policies
  - Regulate how resources can be accessed by SN participants
- Filtering Policies
  - Specify by a user to state which information she prefers not to access
  - Protect users from inappropriate or unwanted content
  - Do not equal to *negative* access control policies
- Admin Policies
  - State who is authorized to specify policies and for which users and objects

C09: The Values

- Relationships between users and resources
  - Access control of most existing models is solely based on the relationships between accessing user and resource owner
  - The only relationship between user and resource is ownership
  - Annotation based relationships need to be addressed
- Admin Policy Model
  - In SN, users should be recognized as the main authority over AC policies regarding the information related to them
- Filtering Policies
  - Protect users from inappropriate or unwanted data
- Hierarchical Structure for Policy Inference
  - Facilitate automatic policies propagation

Our Own Work

- Developed access control for OSNs based on relationships on the social graph
  - UURAC: User-to-User Relationship-based Access Control (DBSec 12)
  - URRAC: User-to-Resource Relationship-based Access Control (Winner of Best Paper Award at PASSAT 12)

Motivating Examples

- Related User's Control
  - There exist several different types of relationships in addition to *ownership*
  - e.g., Alice and Carol want to control the release of Bob's photo which contains Alice and Carol's image.
- Administrational Control
  - A change of relationship may result in a change of authorization
  - Treat *administrative activities* different from normal activities
    - Policy specifying, relationship invitation and relationship recommendation
  - e.g., Bob's mother Carol may not want Bob to become a friend with her colleagues, to access any violent content or to share personal information with others.

Problems

- Traditional access control mechanisms are not suitable for OSNs
  - OSNs keep massive resources and change dynamically
- Existing relationship-based access control approaches are coarse-grained and limited
  - Commercial systems support either limited types or limited depth of U2U relationships
  - Academic works are also not flexible and expressive enough in relationship composition
- Policy administration and conflict resolution are missing
  - Multiple users can specify policies for the same resource

Scope and Assumptions

- Assumptions
  - The threat model does *not* include OSN providers
  - Users' computers are *not* compromised by malicious intruders or malwares
  - *Do not* consider the case when a hacker gains unauthorized access to a site's code and logic
- Scope
  - Aim to improve the *access control* mechanism

Comparison

	Fong [14]	Fong [12,13]	Carminati [8]	Carminati [5,6]	UURAC [10]	URRAC [9]
Relationship Category						
Multiple Relationship Types		✓	✓	✓	✓	✓
Directional Relationship		✓	✓		✓	✓
U2U Relationship	✓	✓	✓	✓	✓	✓
U2R Relationship				✓		✓
Model Characteristics						
Policy Individualization	✓	✓	✓	✓	✓	✓
User & Resource as a Target				(partial)	✓	✓
Outgoing/Incoming Action Policy				(partial)	✓	✓
Relationship Composition						
Relationship Depth	0 to 2	0 to n	1 to n	1 to n	0 to n	0 to n
Relationship Composition	f, f of f	exact type sequence	path of same type	exact type sequence	path pattern of different types	path pattern of different types

- The advantages of our approach:
  - Passive form of action allows outgoing and incoming action policy
  - Path pattern of different relationship types and hopcount skipping make policy specification more expressive
  - System-level conflict resolution policy

Social Networks

- Social graph is modeled as a directed labeled simple graph $G=<U, E, \Sigma>$
  - Nodes *U* as users
  - Edges *E* as relationships
  - $\Sigma=\{\sigma_1, \sigma_2, \ldots,\sigma_n, \sigma_1^{-1}, \sigma_2^{-1},\ldots, \sigma_n^{-1}\}$

  as relationship types supported

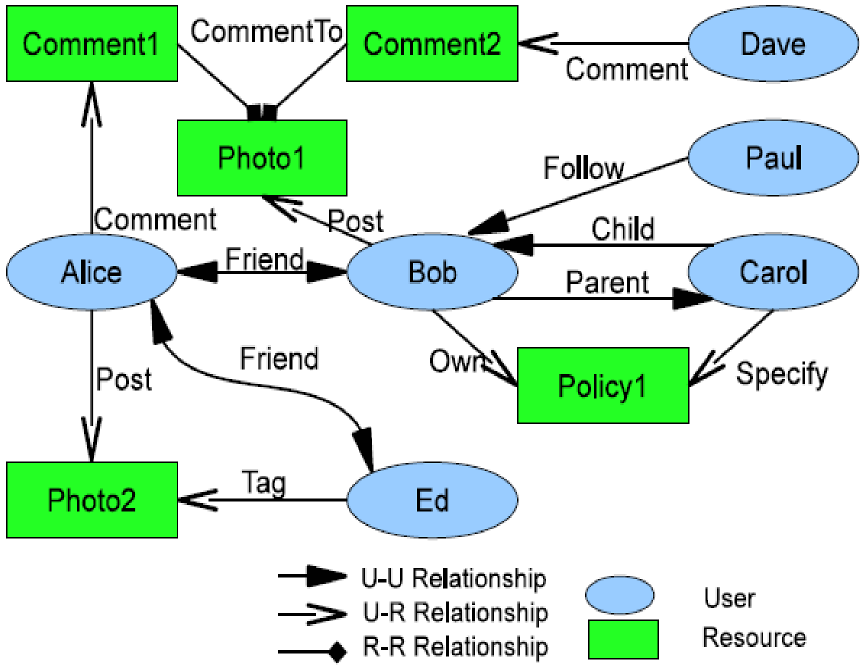
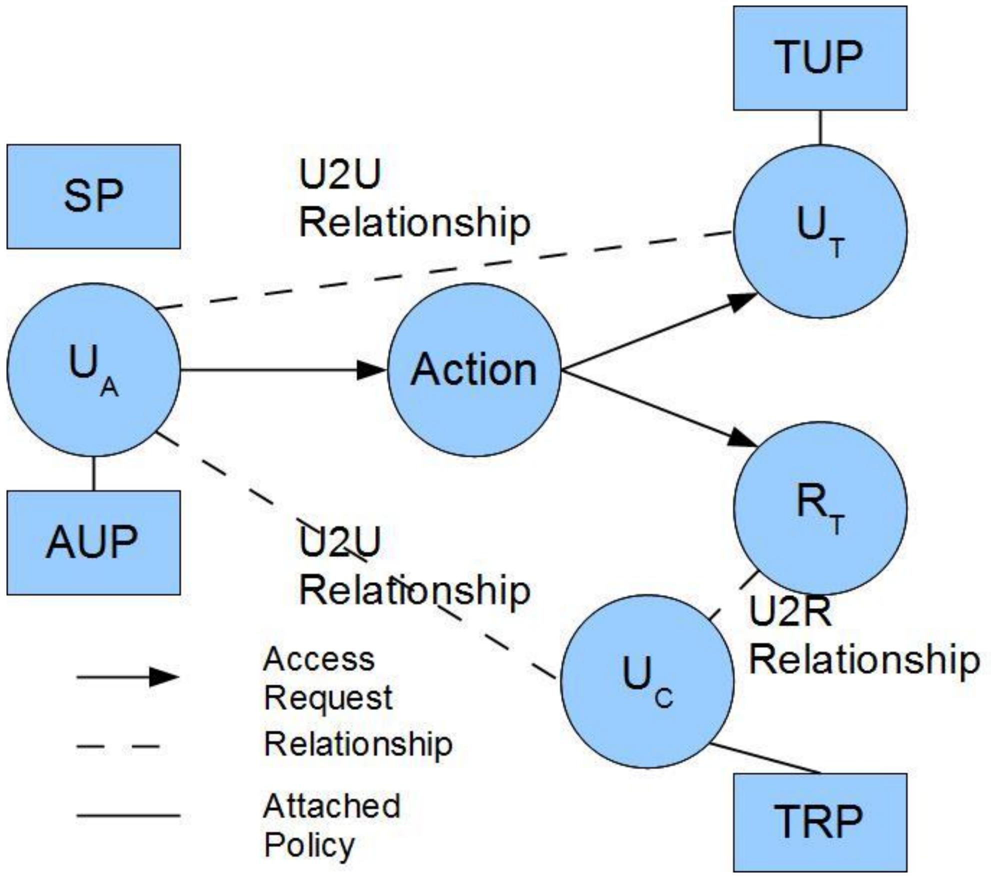


Fig. 3. A Sample Social Graph

Characteristics of Access Control in OSNs

- Policy Individualization
  - Users define their own privacy and activity preferences
  - Related users can configure policies too
  - Collectively used by the system for control decision
- User and Resource as a Target
  - e.g., poke, messaging, friendship invitation, etc.
- User Policies for Outgoing and Incoming Actions
  - User can be either requester or target of activity
  - Allows control on 1) activities w/o knowing a particular resource and 2) activities against the user w/o knowing a particular access requestor
  - e.g., block notification of friend's activities; restrict from viewing violent contents

U2U Relationship-based Access Control (UURAC) Model



U_A: Accessing User
U_T: Target User
U_C: Controlling User
R_T: Target Resource
AUP: Accessing User Policy
TUP: Target User Policy
TRP: Target Resource Policy
SP: System Policy

- Policy Individualization
- User and Resource as a Target
- Separation of user policies for incoming and outgoing actions
- Regular Expression based path pattern w/ max hopcounts (e.g., <u_a, (f*c,3)>)

Access Request and Evaluation

- Access Request <u_a, *action*, *target*>
  - u_a tries to perform *action* on *target*
  - Target can be either user u_t or resource r_t

- Policies and Relationships used for Access Evaluation
  - When u_a requests to access a user u_t
    - u_a's AUP, u_t's TUP, SP
    - U2U relationships between u_a and u_t
  - When u_a requests to access a resource r_t
    - u_a's AUP, r_t's TRP (associated with u_c), SP
    - U2U relationships between u_a and u_c

Policy Representations

Accessing User Policy	$< action, (start, path\ rule)>$
Target User Policy	$< action^{-1}, (start, path\ rule)>$
Target Resource Policy	$< action^{-1}, r_t, (start, path\ rule)>$
System Policy for User	$< action, (start, path\ rule)>$
System Policy for Resource	$< action, r.type, (start, path\ rule)>$

- *action*$^{-1}$ in TUP and TRP is the passive form since it applies to the recipient of action
- TRP has an extra parameter r_t to distinguish the actual target resource it applies to
  - owner(r_t)→ a list of u_c→U2U relationships between u_a and u_c
- SP does not differentiate the active and passive forms
- SP for resource needs *r.type* to refine the scope of the resource

Graph Rule Grammar

$GraphRule ::= \text{“(”} < StartingNode > \text{“,”} < PathRule > \text{“)”}$

$PathRule ::= < PathSpecExp > | < PathSpecExp >< Connective >< PathRule >$

$Connective ::= \vee | \wedge$

$PathSpecExp ::= < PathSpec > | \neg < PathSpec >$

$PathSpec ::= \text{“(”} < Path > \text{“,”} < HopCount > \text{“)”} | \text{“(”} < EmptySet > \text{“,”} < Hopcount > \text{“)”}$

$HopCount ::= < Number >$

$Path ::= < TypeExp > | < TypeExp >< Path >$

$EmptySet ::= \emptyset$

$TypeExp ::= < TypeSpecifier > | < TypeSpecifier >< Wildcard >$

$StartingNode ::= u_a | u_t | u_c$

$TypeSpecifier ::= \sigma_1 | \sigma_2 | .. | \sigma_n | \sigma_1^{-1} | \sigma_2^{-1} | .. | \sigma_n^{-1} | \Sigma \text{ where } \Sigma = \{\sigma_1, \sigma_2, .., \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, .., \sigma_n^{-1}\}$

$Wildcard ::= \text{“} * \text{”} | \text{“?”} | \text{“} + \text{”}$

$Number ::= [0 - 9]+$

Example

- Alice's policy P$_{Alice}$:
  - $< poke, \left(u_a, (f*, 3)\right) >, < poke^{-1}, \left(u_t, (f, 1)\right) >,$
  - $< read, \left(u_a, (\Sigma*, 5)\right) >, < read^{-1}, file1, \left(u_c, (cf*, 4)\right) >$
- Harry's policy P$_{Harry}$:
  - $< poke, \left(u_a, (cf*, 5) \vee (f*, 5)\right) >, < poke^{-1}, \left(u_t, (f*, 2)\right) >$
  - $< read^{-1}, file2, (uc, \neg(p+, 2) >$
- System's policy P$_{Sys}$:
  - $< poke, \left(u_a, (\Sigma*, 5)\right) >$
  - $< read, photo, \left(u_a, (\Sigma*, 5)\right) >$

- "Only Me"
  - $< poke, \left(u_a, (\emptyset, 0)\right) >$ says that ua can only poke herself
  - $< poke^{-1}, \left(u_t, (\emptyset, 0)\right) >$ specifies that ut can only be poked by herself
- The Use of Negation Notation
  - $(fffc \wedge \neg fc)$ allows the coworkers of the user's distant friends to see, while keeping away the coworkers of the user's direct friends

Policy Extraction

- Policy: <*act*, *graph rule*>

It determines the starting node, where the evaluation starts

The other user involved in access becomes the evaluating node

- Graph Rule: *start*, *path rule*

↓

- Path Rule: *path spec* ∧|∨ *path spec*

↓

- Path Spec: *path*, *hopcount*

Path-check each path spec using Algorithm 2 (introduced in detail later)

Policy Evaluation

- Evaluate a combined result based on conjunctive or disjunctive connectives between path specs
- Make a collective result for multiple policies in each policy set.
  - Policy conflicts may arise. We assume system level conflict resolution strategy is available (e.g., disjunctive, conjunctive, prioritized).
- Compose the final result from the result of each policy set (AUP, TUP/TRP, SP)

Path Checking Algorithm

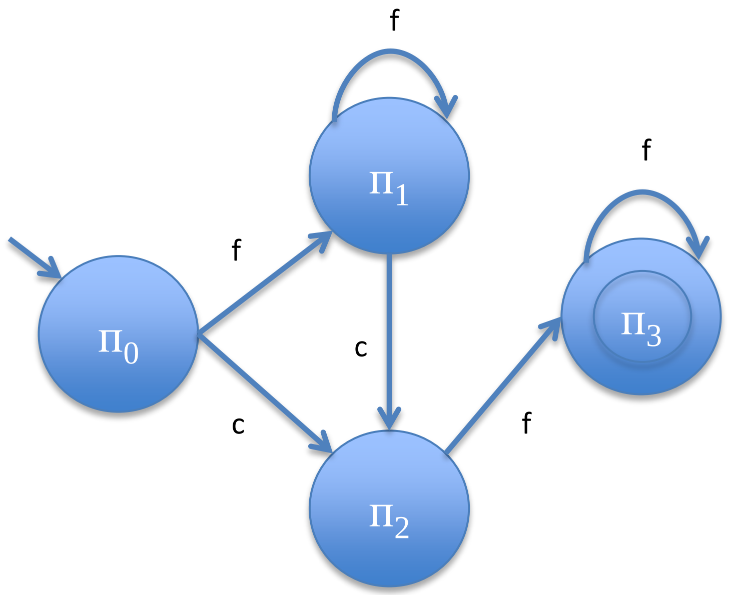
- Parameters: G, path, hopcount, s, t
- Traversal Order: Depth-First Search
  - Activities in OSN typically occur among people with close distance
  - DFS needs only one pair of variables to keep the current status and history of exploration
  - Hopcount limit prevents DFS from lengthy useless search

Initiation

Access Request: (Alice, read, r_t)

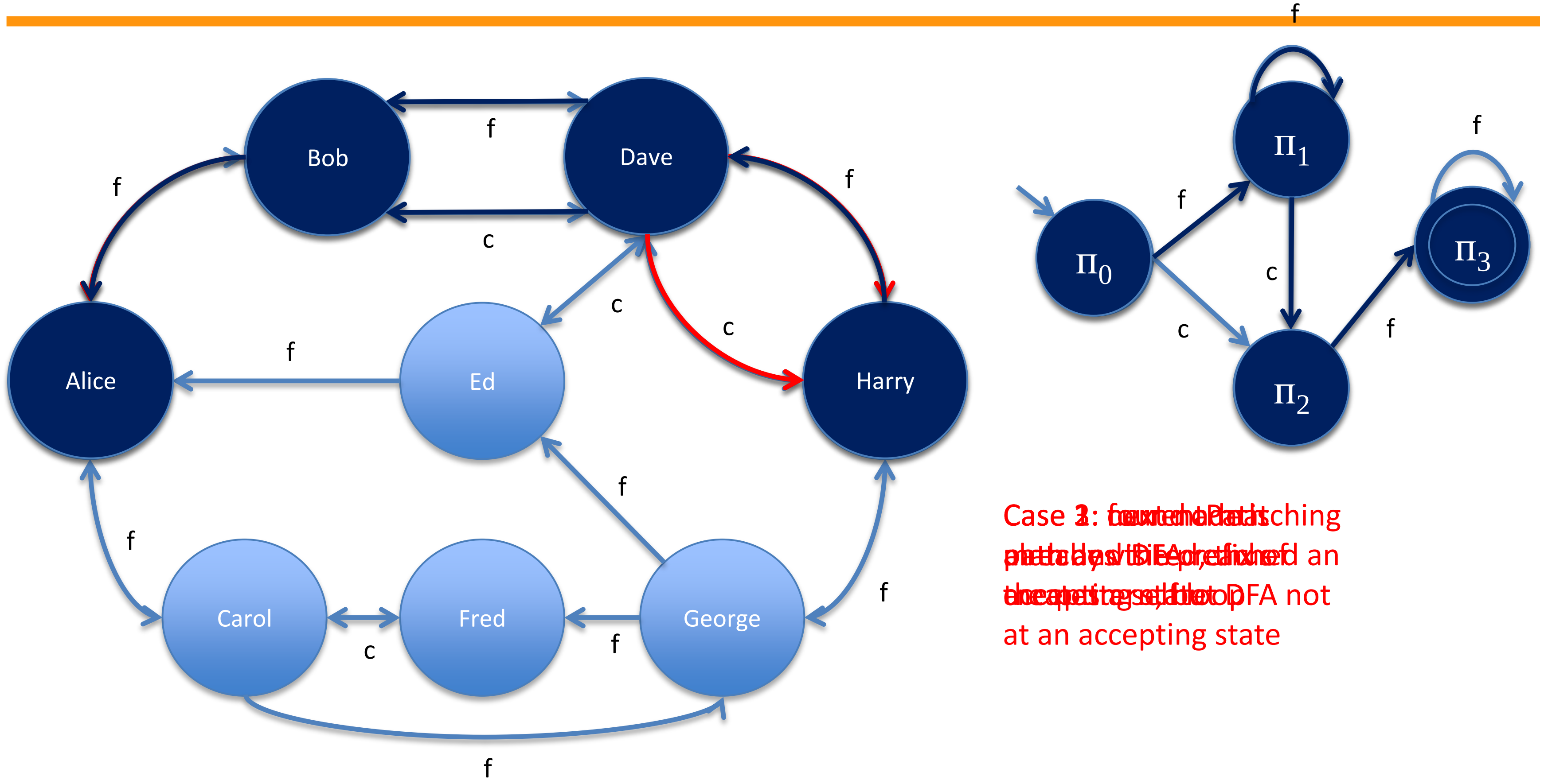
Policy: ($read^{-1}$, r_t, (f*cf*, 3))

Path pattern: f*cf*
Hopcount: 3



DFA for f*cf*

Path pattern: f*cf*
Hopcount: 3



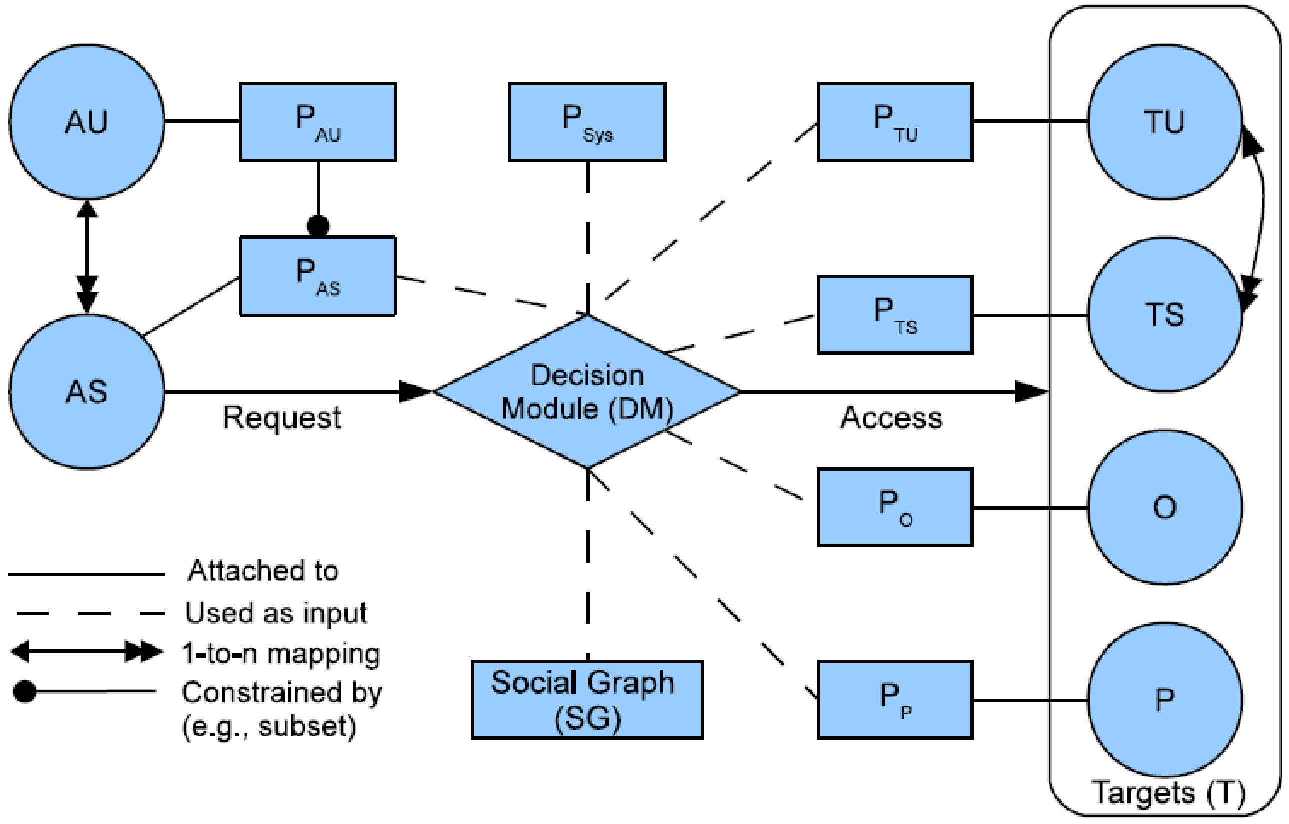
Case [illegible]: [illegible]ching [illegible]d an [illegible] DFA not at an accepting state

d: [illegible]
currentPath: (H,D,f)(D,B,c)(B,A,f)
stateHistory: 0123

Beyond U2U Relationships

- There are various types of relationships between users and resources in addition to U2U relationships and ownership
  - e.g., share, like, comment, tag, etc
- U2U, U2R and R2R
- U2R further enables relationship and policy administration

URRAC Model Components



AU: Accessing User
AS: Accessing Session
TU: Target User
TS: Target Session
O: Object
P: Policy
P_{AU}: Accessing User Policy
P_{AS}: Accessing Session Policy
P_{TU}: Target User Policy
P_{TS}: Target Session Policy
P_{O}: Object Policy
P_{P}: Policy for Policy
P_{Sys}: System Policy

Differences with UURAC

- Access Request
  - (s, act, T) where T may contain multiple objects
- Hopcount Skipping
  - Option to omit the hops created by resources
  - Hopcount stated inside [[]] will not be counted in the global hopcount
  - e.g., ([f*,3][[c*,2]],3)
- Policy Administration
- User-session Distinction

Hopcount Skipping

- U2R and R2R relationships may form a long sequence
  - Omit the distance created by resources
  - Local hopcount stated inside "[[]]" will not be counted in global hopcount.
  - E.g., "([*f**,3][[*c**, 2]],3)", the local hopcount 2 for *c** does not apply to the global hopcount 3, thus allowing *f** to have up to 3 hops.
- Six degrees of separation
  - Any pair of persons are distanced by about 6 people on average. (4.74 shown by recent study)
  - Hopcount for U2U relationships is practically small

Policy Conflict Resolution

- System-defined conflict resolution for potential conflicts among user-specified policies
- Disjunctive, conjunctive and prioritized order between relationship types
  - $\wedge, \vee, >$ represent disjunction, conjunction and precedence
  - @ is a special relationship "null" that denotes "self"

Policy Conflict Resolution (cont.)

- $< read^{-1}, (own \wedge tag) >$
  - The more rigid one between the owner's and the tagged users' "read-1" policies over the photo is honored.
- $< friend\_request, (parent > @) >$
  - When child attempts friendship request to someone, parents' policies get precedence over child's own will.
- $< share^{-1}, (own \vee tag \vee share) >$
  - A weblink is sharable if either the original owner, or any of the tagged users or shared users allows.

Example

- **View a photo where a friend is tagged.** *Bob and Ed are friends of Alice, but not friends of each other. Alice posted a photo and tagged Ed on it. Later, Bob sees the activity from his news feed and decides to view the photo: (Bob, read, Photo2)*
  - *Bob's* $P_{AS}(read)$: $<read,(u_a,([\Sigma_{u\_u}*,2][[\Sigma_{u\_r},1]],2))>$
  - *Photo2's* $P_O(read^{-1})$ by *Alice*: $<read^{-1},(t,([post^{-1},1][friend*,3],4))>$
  - *Photo2's* $P_O(read^{-1})$ by *Ed*: $<read^{-1},(u_c,([friend],1))>$ In conflicts
  - $AP_{Sys}(read)$: $<read,(ua,([\Sigma_{u\_u}*,5][[\Sigma_{u\_r},1]],5))>$
  - $CRP_{Sys}(read)$: $<read^{-1},(own \wedge tag)>$

Example (cont.)

- **Parental control of policies.** *The system features parental control such as allowing parents to configure their children's policies. The policies are used to control the incoming or outgoing activities of children, but are subject to the parents' will. For instance, Bob's mother Carol requests to set some policy, say Policy1 for Bob: (Carol, specify policy, Policy1)*
  - *Carol*'s P_{AS}*(specify_policy)*: *<specify_policy,*$(u_\alpha,([own],1) \vee ([child \cdot own],2))$*>*
  - *Policy1'*s P_P*(specify_policy*$^{-1}$*)* by *Bob*: *<specify_policy*$^{-1}$*,*$(t,([own^{-1}],1))$*>*
  - P_{Sys}*(specify_policy)*: *<specify_policy,*$(u_\alpha,([own],1) \vee ([child \cdot own],2))$*>*
  - CRP_{Sys}*(specify_policy)*: *<specify_policy, (parent* $\wedge$ *@)>*

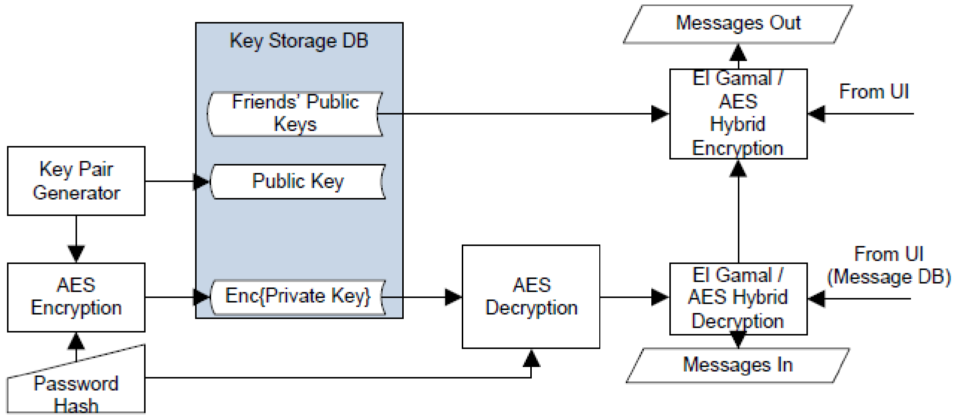
Outline

- Introduction
- Security & Privacy Issues in OSNs
- Access Control for OSNs
- Other Privacy Preservation Solutions

flyByNight: Mitigating the Privacy Risks of Social Networking

- A Facebook application designed to encrypt and decrypt data with an objective to mitigate privacy risks in OSNs.
- Primary goal:
  - Hide information transferred through the OSN from the provider and the application server.
- Key ideas:
  - Encrypt sensitive data on the client side and send the cipher text to intended parties.
  - Uses
    - El-Gamal encryption
    - Proxy Cryptography

How It Works



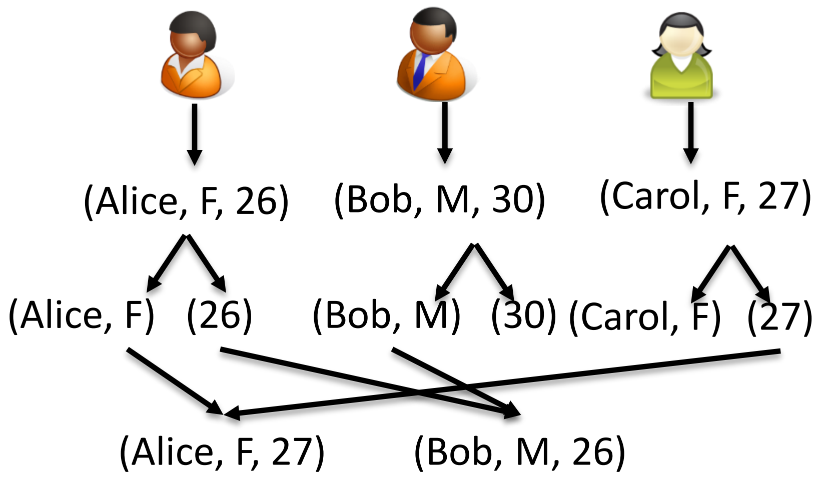
- Initialization
  - Client generates Public/Private key pair, password
  - Client transfers encrypted private key to flyByNight server, and saves in key Database
- Send Data:
  - Client encrypts private data M with friends' PK, and tags the encrypted data with friends' ID, saves encrypted data in message Database on flyByNight server
- Receive Data:
  - Client decrypts private key with password, decrypts M with the private key

NOYB: Privacy in Online Social Networks

- An architecture that scatters user data to protect privacy while preserving the functionality of OSN service
- Key Ideas:
  - Encrypt user data such that the cipher text shares the same semantic and statistical properties with legitimate data
  - Allow the OSN provider to work on cipher text

Architecture

- Uses out of band channel for key management
- User data is divided into atoms
- Atoms of similar type constitute a dictionary
- Atoms are replaced with other atoms from the dictionary



Conclusion

- The emergence of OSNs pose severe privacy risks to users
- Lots of work have been done to protect privacy and security of user data
  - Access control models
  - Cryptographic solutions
  - Social networking platforms for third party applications

Questions?