Leveraging Actors for Privacy Compliance

Jeffery von Ronne[1]

Department of Computer Science
The University of Texas at San Antonio

October 21, 2012

[1]This work was supported by the National Science Foundation under grant CNS-0964710.

Outline

Outline

Societal Expectations of Privacy

> *Whatever, in connection with my professional service, or not in connection with it, I see or hear, in the life of men, which ought not to be spoken of abroad, I will not divulge, as reckoning that all such should be kept secret.*
>
> — *Hippocratic Oath*

Professional Expectations:

- priest-penitent
- doctor-patient
- attorney-client

Abundance of Private Information

Personal information that we expect to be kept secret:

- disease diagnoses
- credit card purchases
- dates of birth
- social security numbers
- income
- bank account balances

Legal and Contractual Protections

Legislated Protections:

- Family Education Rights and Privacy Act (FERPA)
- Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule
- Gramm-Leach-Biley (GLB) Act Financial Privacy Rule
- European Union Data Protection Directive/Regulation

Contractual protections:

- non-disclosure agreements
- privacy policies / notices

Examples from HIPAA Regulations

A covered entity may not use or disclose protected health information, except as permitted or required . . . A covered entity is permitted to use or disclose protected health information as follows: To the individual . . . For treatment, payment, or health care operations . . . Pursuant to and in compliance with a valid authorization . . .

CFR Title 45, Section 164.502

Except as otherwise provided . . . , an individual has a right of access to inspect and obtain a copy of protected health information about the individual in a designated record set, for as long as the protected health information is maintained in the designated record set, except for . . .

CFR Title 45, Section 164.524

Organizational Privacy Policies

Under HIPAA, each *covered entity* must:

- have a privacy policy
- provide it to patients
- comply with it where it is stronger than other laws
  - violations of the privacy policy are a violation of HIPAA

Similarly, financial institutions are required to provide Privacy Notices.

Information Systems

Organizations use Computer Systems to Process and Communicate Protected Information

- patient billing
- electronic medical record systems
- online banking systems

Do these systems avoid making unlawful disclosures?

Do these systems perform necessary privacy-related functions?

- Those functions an organization relies on to fulfill its obligations related to privacy.

Ensuring Compliance of Information Systems

One approach:

- Formalize policies in temporal logic.
- Design and implement the software system as hierarchical collections of communicating actors.
  - Each actor/collection has its own specification of what it can/must communicate.
  - Verify actor code against policy.
  - Verify composition of local specifications against policy.

Outline

Formalizing Privacy Policy with First-Order Temporal Logic

Formalize Privacy Properties in First-Order Temporal Logic (FOTL)

- based on Barth et al. (2006)'s work on formalizing contextual integrity
- focus is transmissions of messages (containing protected information)
- sequence of such transmissions is or is not compliant
- FOTL formula defines what is compliant

LTL Formulas Over Sequences of Messages

Assume:

- Alice is a doctor.
- Bob is her patient.
- Charlie is a third party individual.
- Bob's height is protected health information.

Policy:

$$\Box \forall m.\text{send}(\text{alice}, \text{charlie}, m) \wedge \text{contains}(m, \text{bob}, \text{height})$$

$$\rightarrow \diamondminus \text{send}(\text{bob}, \text{alice}, \text{authorize-release}(\text{bob}, \text{height}, \text{charlie}))$$

"henceforth": $\Box\phi$ means ϕ holds in all subsequent 'states'
"once": $\diamondminus\phi$ means ϕ holds in some previous 'state'

LTL Formulas Over Sequences of Messages

Policy:

$$\Box \forall m.\text{send}(\text{alice}, \text{charlie}, m) \wedge \text{contains}(m, \text{bob}, \text{height})$$

$$\rightarrow \diamondsuit \text{send}(\text{bob}, \text{alice}, \text{authorize-release}(\text{bob}, \text{height}, \text{charlie}))$$

Not compliant:

```
send(charlie, alice, request-information(bob, height))
send(alice, charlie, {(bob, height, 2 meters)})
```

Compliant:

```
send(bob, alice, authorize-release(bob, height, charlie))
send(charlie, alice, request-information(bob, height))
send(alice, charlie, {(bob, height, 2 meters)})
```

More General Policy Example

$$\Box \forall p_1, p_2, q, m_1, t.\text{send}(p_1, p_2, m_1) \wedge \text{contains}(m_1, q, t)$$
$$\rightarrow$$
$$\text{inrole}(p_1, \text{doctor}) \wedge \text{inrole}(p_2, \text{covered-entity}) \wedge \texttt{medical-record}(m_1, q)$$
$$\vee$$
$$\diamondminus \exists m_2.\text{send}(q, p_2, m_2) \wedge \text{authorizes}(m_2, q, t)$$
$$\vee$$
$$\text{inrole}(p_2, \text{doctor})$$
$$\vee$$
$$p_2 = q$$

$$\Box \forall p_1, p_2, m_1.\text{send}(p_1, p_2, m_1) \wedge \text{request-records}(m_1, p_1)$$
$$\rightarrow$$
$$\forall p_3, m_2.\diamondminus \text{send}(p_3, p_2, m_2) \wedge \text{medical-record}(m_2, p_1) \rightarrow \Diamond \text{send}(p_2, p_1, m_2)$$

"eventually": $\Diamond \phi$ means that at some future 'state' ϕ holds.

Outline

Information System as Actors

Characteristics of Information Systems:

- distributed rather than monolithic
- evolving structure

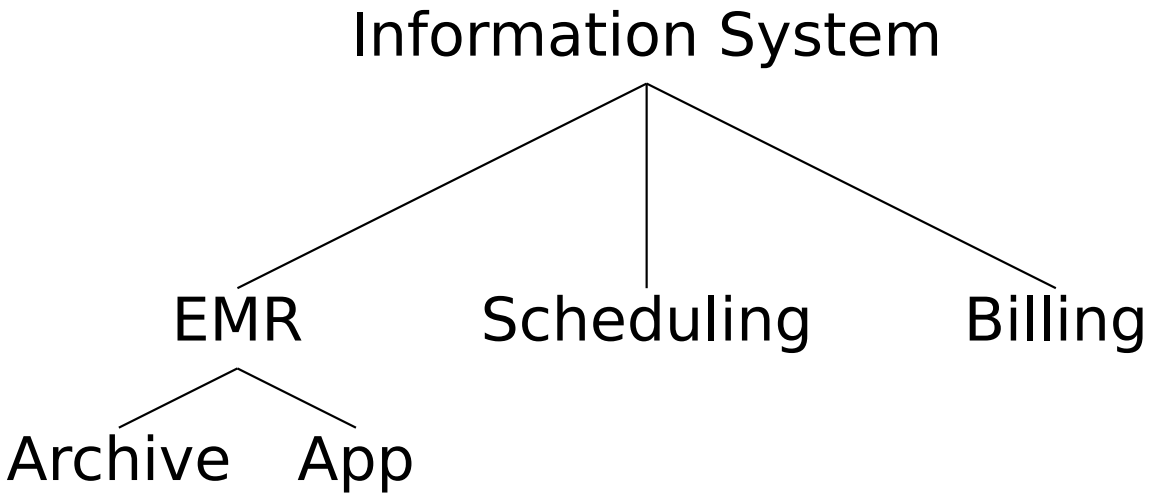
Characteristics of Policies:

- views history as sequence of messages
- describes permissible communication

Why Actors?

- focus is on communication; matches policies
- hierarchical composition modeled after actor configurations in Agha et al. (1997) and Talcott (1998)
  - can handle evolving distributed systems

Example: Medical Clinic Information System



Example HAPL Component Definition

```
component System (clinic:P) {
  channel emr = records.app

  records = new component EMR()
  scheduling = new component Scheduling()
  billing = new component Billing()


  component Billing() { ... }

  component Scheduling() { ... }

  component EMR() {
    channel app = new actor EMRApp(clinic,archive)
    archive = new actor EMRArchive(clinic)
  }
}
```

Example HAPL Actor Definition

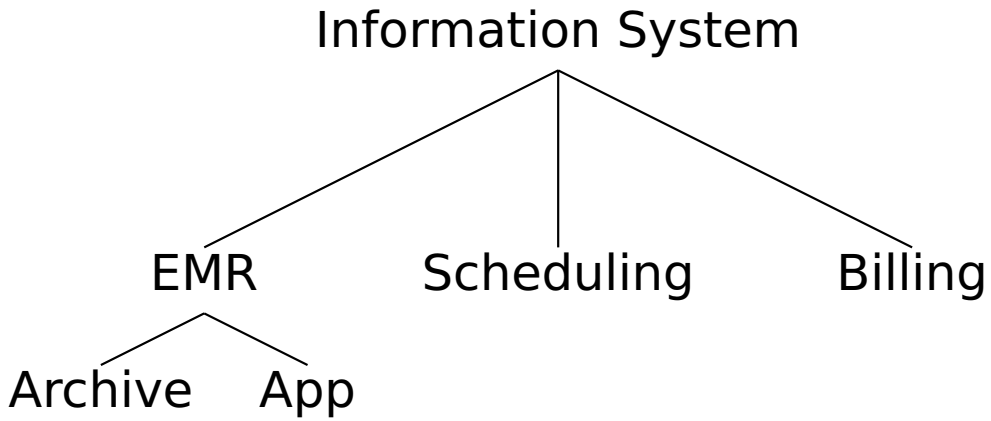
```
message request_rec(P,Actor,P,Date);
message store_record(PatientRecord);
message authorize(P,P,Date);

actor EMRArchive(var clinic:P) {

  var store = new List[PatientRecord]();

  action request_rec(requester:P,dest:Actor,patient:P,date:Date)
  {
    ...
  }

  action store_record(record:PatientRecord) {
    ...
  }

  action authorize(clinic:P,authorizee:P,date:Date) {
    ...
  }
}
```

Verification Strategy

Verification Strategy

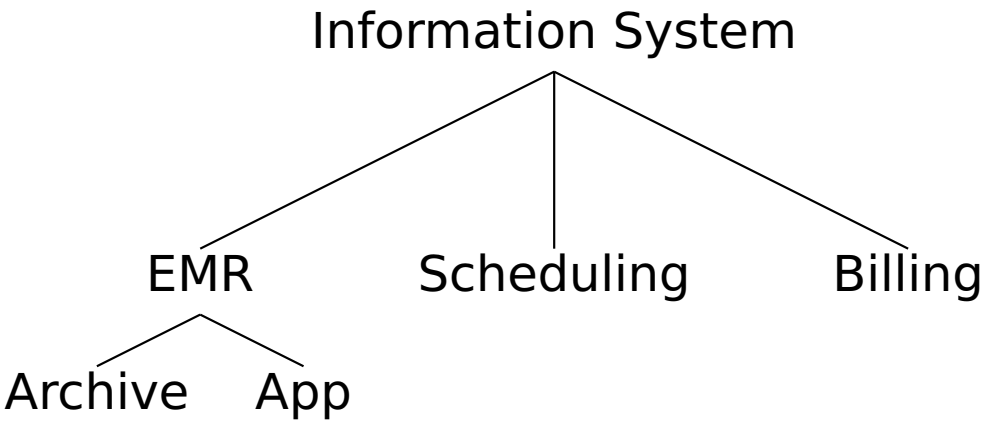
- develop privacy specifications
  - for each Actor class
  - for each HAPL component
- use static analysis
  - verify HAPL actor classes against specifications
- use model checking and other techniques
  - actor class specifications entail actor component specifications
  - actor component specifications entail privacy policy

Example: Permissible Transmissions (Informal)



Policy Only transmissions: to subject; to doctor; submission of medicall records by doctor; or with authorization

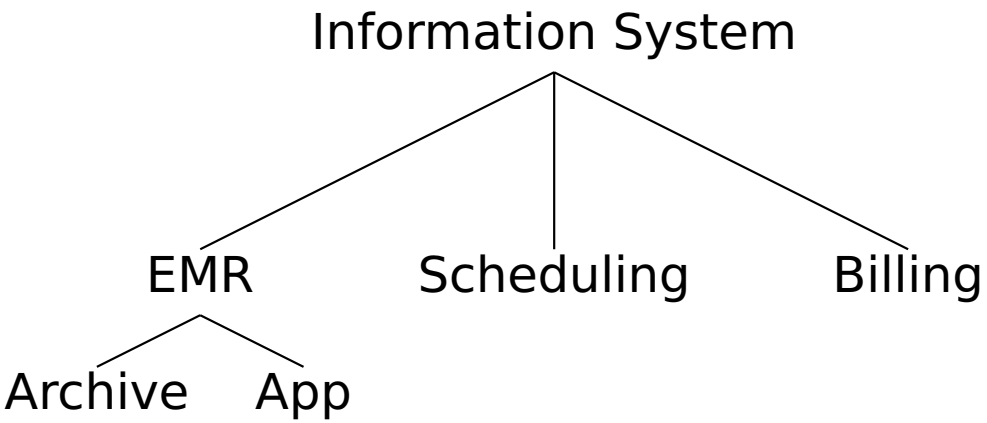
Example: Permissible Transmissions (Informal)



Policy Only transmissions: to subject; to doctor; submission of medicall records by doctor; or with authorization

Information System actors in system only transmit to subject; doctors; when an authorization has previously been received by the `emr` channel; and internally

Example: Permissible Transmissions (Informal)

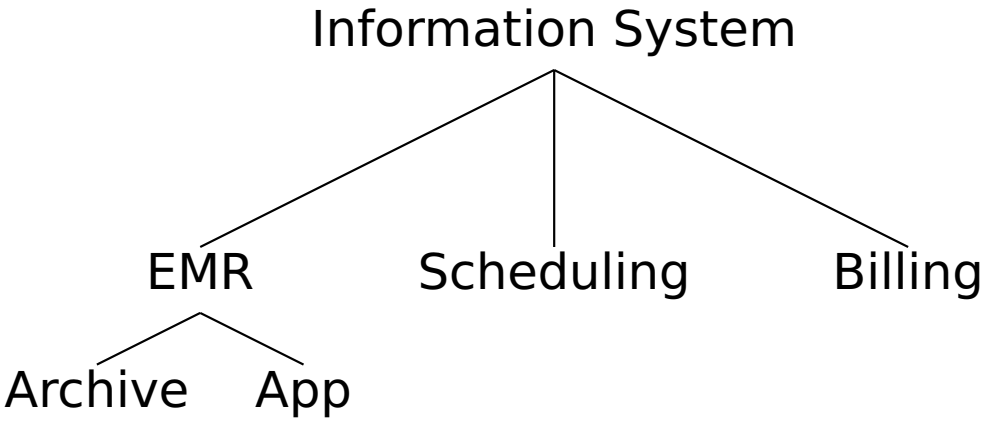


Information System actors in system only transmit to subject; doctors; when an authorization has previously been received by the `emr` channel; and internally

EMR actors in EMR only transmit to subject; doctors; when an authorization has previously been received by the `emr` channel; and internally

Scheduling/Billing actors in EMR only transmit to individuals; doctors; and internally

Example: Permissible Transmissions (Informal)

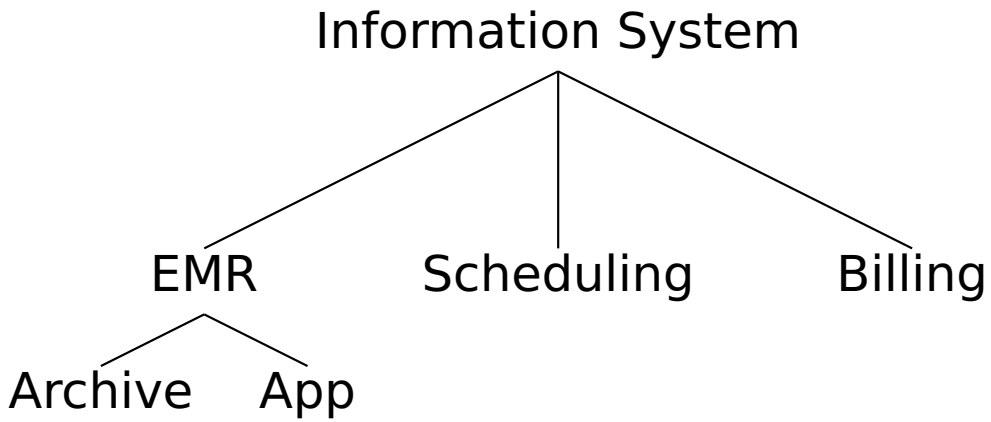


EMR actors in EMR only transmit to subject; doctors; when an authorization has previously been received by the `emr` channel; and internally

Archive an EMR Archive Actor only transmits to a patient; doctor; or when an EMR App certifies there was prior authorization

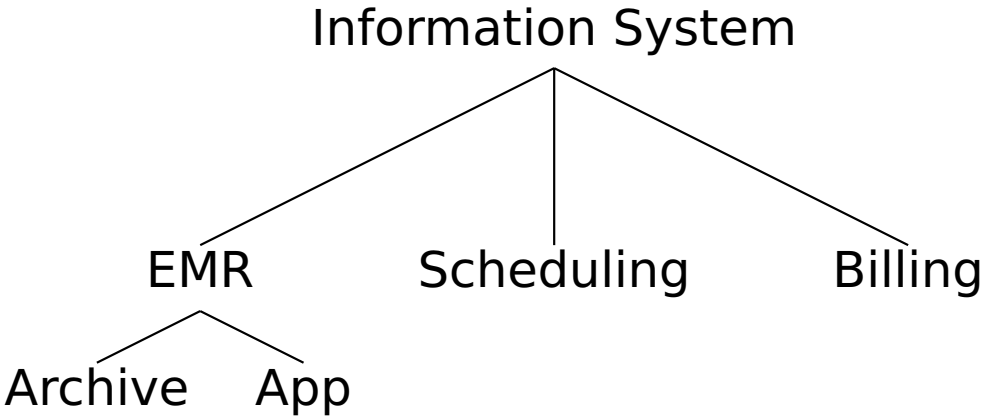
App an EMR App only transmits a certification of prior authorization if it has been previously received from the subject; no transmissions of protected information

Example: Required Transmissions (Informal)



Policy Must send information subjects their own records on request. Requests must be submitted through EMR web interface.

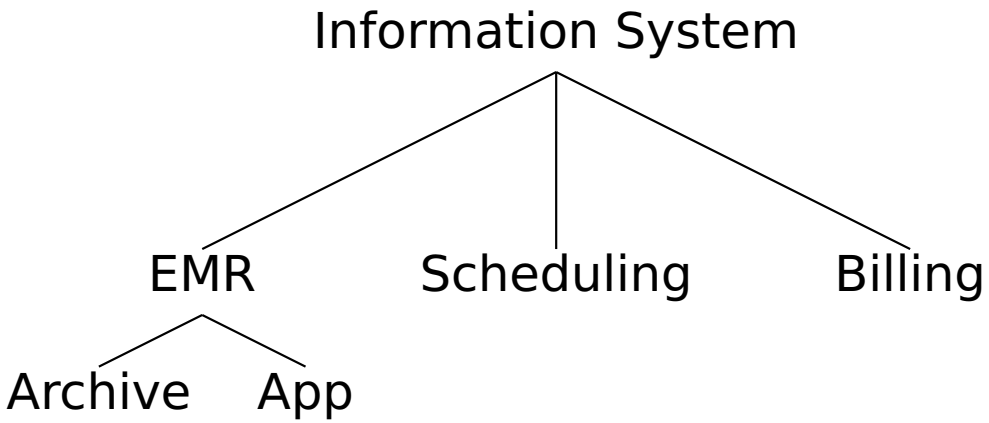
Example: Required Transmissions (Informal)



Policy Must send information subjects their own records on request. Requests must be submitted through EMR web interface.

Information System If request is received on emr channel, must send requester all of their own medical records.

Example: Required Transmissions (Informal)

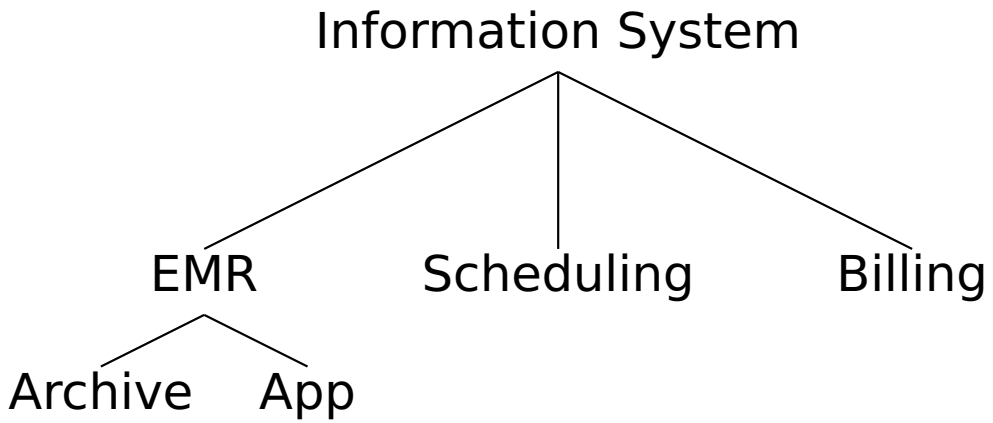


Information System If request is received on emr channel, must send requester all of their own medical records.

EMR If request received by emr.app, must send requester all medical records.

Scheduling/Billing no requirements

Example: Required Transmissions (Informal)



EMR If request received by emr.app, must send requester all medical records.

Archive If request received, must send all medical records to requester.

App If request received, must forward to archive.

Outline

Status

Ongoing project at UTSA

- Collaborators: William Winsborough, Jianwei Niu, Omar Chowdhuri, Shamim Ashik
- HAPL language
  - source-to-source (HAPL to Scala) translator (for Actor class sub-language)
  - design of static semantics (type system and static analysis)
  - in progress:
    - prototype EMR system
    - implementation of verifier (static semantics)
- specification
  - policy language and formalization of HIPAA in language
  - in progress:
    - small model theorem
    - slicing techniques
    - analysis techniques

Summary

Problem

- How do we know information systems handle private data correctly?

Approach

- formalize privacy policy using temporal logic
- design system as hierarchical collections of actors
- create formal privacy specification for each class of actor and for each collection of actors (component)
- implement Actors using a domain specific language
  - provides abstractions for actors and messages
  - provides abstractions for principals and protected information
- verify code against actor class privacy specifications
- verify actor specifications entail specifications for collections and privacy policy