The Personal Grid: Eliminating Large-scale Health Record Breaches

William A. Yasnoff, MD, PhD, FACMI
Managing Partner, NHII Advisors
Adjunct Professor, Health Sciences Informatics, Johns Hopkins University
A secure and efficiently searchable health information architecture

William A. Yasnoff *

NHl Advisors, 1854 Clarendon Blvd., Arlington, VA 22201-2914, United States
Health Sciences Informatics, Johns Hopkins University, 2024 Monument St., Suite 1-200, Baltimore, MD 21205, United States

http://dx.doi.org/10.1016/j.jbi.2016.04.004
The Health IT Problem

- Comprehensive Electronic Patient Records When and Where Needed
  - Immediate access to comprehensive records for individuals (for care)
  - Ability to search and aggregate across population (for population health, quality improvement, medical research, and policy)
    - Does not require immediate response time
Health IT Requirements

- All patient records must be
  - Digital
  - Encoded with common standards
- Need mechanism for aggregation
  - Individual records created & stored at every site of care
  - Must be able to immediately access all records for a given individual
Architectural Approaches

- Federated (Distributed) Model
  - Leave records where they are created
  - Retrieve and aggregate records in real time when needed

- Centralized Model
  - Deposit records in a centralized repository
  - Each patient’s records stored together in one “account”
  - Comprehensive records immediately available when needed
Federated (Distributed) Model

Diagram © Health Record Banking Alliance, 2013. Used by permission.
Problems with Distributed Model*

- Complex
  - Real time record reconciliation
  - Requires unique patient identifier
- Prone to Error – missing data
- Insecure – data transmitted twice each time
- Searching Not Feasible
  - Sequential
  - Massive processing load on source EHRs
- Not Financially Sustainable
  - High cost
  - Cannot capture value of searching

Centralized Model

Diagram © Health Record Banking Alliance, 2013. Used by permission.
Problems with Centralized Model

- **Governance**
  - Who is data steward?
  - Who controls access?
    - Individual records
    - Searching across population
- **Stakeholder Cooperation**
  - Reluctance to deposit “their” data
- **Single Point of Failure**
- **Security**
  - Risk of loss of entire dataset
Repository Security Challenge

- Centralized data best way to ensure security*
  - Distributed data less secure: multiple transmission for each use
- Inherent vulnerability of central database
  - Potential loss of all data in one incident
- Multiple security breaches → widespread belief that nothing is secure
  - Perception is now reality
- Challenge: Efficient search without large-scale breach vulnerability

---

Patient Data Repository Requirements

1. Individual Record Access - Immediate

2. Efficient Searching Across Population of Records
   - Immediate access not needed

3. Security
   - Prevent loss of entire dataset from a single intrusion
Specific Security Requirements

- No unauthorized access to entire dataset via:
  1. Any single decryption key
  2. Any single user’s credentials
  3. Any single file
  4. Searching of the dataset
Personal Grid Architecture

- Each patient’s record stored in separate file with separate encryption
- Efficient searching using highly parallel virtual processors in cloud and/or network (which may include mobile phones)
- No access point for all patients’ data – even for operator of service
  - Eliminates risk of loss of entire dataset
- May be practical and useful alternative architecture for centralized patient-centric health record systems
Technical Description of the Personal Grid Architecture
Typical Repository: Relational Database

**Person**

<table>
<thead>
<tr>
<th>RecNo</th>
<th>LastName</th>
<th>FirstName</th>
<th>BirthYear</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>O’Carroll</td>
<td>Patrick</td>
<td>1957</td>
<td>M</td>
</tr>
<tr>
<td>002</td>
<td>Shakespeare</td>
<td>William</td>
<td>1564</td>
<td>M</td>
</tr>
<tr>
<td>003</td>
<td>Seymour</td>
<td>Jane</td>
<td>1509</td>
<td>F</td>
</tr>
<tr>
<td>004</td>
<td>Dickens</td>
<td>Charles</td>
<td>1812</td>
<td>M</td>
</tr>
<tr>
<td>005</td>
<td>Seymour</td>
<td>Jane</td>
<td>1947</td>
<td>F</td>
</tr>
</tbody>
</table>

**Event**

<table>
<thead>
<tr>
<th>VisNo</th>
<th>RecNo</th>
<th>VacNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>X001</td>
<td>001</td>
<td>VN104</td>
</tr>
<tr>
<td>X002</td>
<td>004</td>
<td>VN101</td>
</tr>
<tr>
<td>X003</td>
<td>003</td>
<td>VN132</td>
</tr>
<tr>
<td>X004</td>
<td>004</td>
<td>VN105</td>
</tr>
<tr>
<td>X005</td>
<td>005</td>
<td>VN105</td>
</tr>
</tbody>
</table>

**Vaccine**

<table>
<thead>
<tr>
<th>VacNo</th>
<th>VacName</th>
<th>LotNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>VN101</td>
<td>HIB</td>
<td>C120-345</td>
</tr>
<tr>
<td>VN102</td>
<td>DTP</td>
<td>BCQ-12</td>
</tr>
<tr>
<td>VN103</td>
<td>Influenza</td>
<td>34-C-7645</td>
</tr>
<tr>
<td>VN104</td>
<td>Hep A</td>
<td>Hep456-9</td>
</tr>
<tr>
<td>VN105</td>
<td>Varicella</td>
<td>Var9-001</td>
</tr>
</tbody>
</table>
Relational Database Index

- Last Name Index

<table>
<thead>
<tr>
<th>Name (sorted)</th>
<th>Rec No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dickens</td>
<td>004</td>
</tr>
<tr>
<td>OÕCarroll</td>
<td>001</td>
</tr>
<tr>
<td>Seymour</td>
<td>003</td>
</tr>
<tr>
<td>Seymour</td>
<td>005</td>
</tr>
<tr>
<td>Shakespeare</td>
<td>002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RecNo</th>
<th>LastName</th>
<th>FirstName</th>
<th>BirthYear</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>OÕCarroll</td>
<td>Patrick</td>
<td>1957</td>
<td>M</td>
</tr>
<tr>
<td>002</td>
<td>Shakespeare</td>
<td>William</td>
<td>1564</td>
<td>M</td>
</tr>
<tr>
<td>003</td>
<td>Seymour</td>
<td>Jane</td>
<td>1509</td>
<td>F</td>
</tr>
<tr>
<td>004</td>
<td>Dickens</td>
<td>Charles</td>
<td>1812</td>
<td>M</td>
</tr>
<tr>
<td>005</td>
<td>Seymour</td>
<td>Jane</td>
<td>1947</td>
<td>F</td>
</tr>
</tbody>
</table>
Relational Database Properties

1. Efficient Access
   ● Rapid retrieval of individual patient data

2. Efficient Searching
   ● Rapid retrieval across population

3. High Potential for Total Data Loss
   ● All tables and indices in a single file
   ● Access to any  → access to all
   ● Leads to large scale breaches
Personal Grid: Separate Records

- Each record is separately stored and encrypted with two keys - one held by the owner and the other by the system.
- Both keys are required to decrypt a record.

User key $\alpha_p$

System key $\beta_p$

File “safe deposit boxes”
Personal Grid Overview

- Each patient’s data stored in central location
  - Separate file for each patient
  - Separate encryption
- Pro: no single point of access to all data
- Con: Sequential searching

*each record stored and encrypted separately
Searching Separate Records

Pt record 1*
Pt record 2
. .
Pt record N

*each record stored and encrypted separately

START → i ← 1

Retrieve record i

Decrypt record i

Search record i

i ← i + 1

i ≥ N?

no

yes

END

Requires N iterations
Searching: 2 Processors

Pt record 1*
Pt record 2
Pt record N

*each record stored and encrypted separately

START → i ← 1

Retrieve record i
Decrypt record i
Search record i

i ← i+2

i>(N-1)?

no → i>(N-1)?
yes → END

Requires N/2 iterations

Requires N/2 iterations
Searching: K Processors

START \( \rightarrow \) \( i \leftarrow 1 \)

- Retrieve rec \( i \)
- Decrypt rec \( i \)
- Search rec \( i \)

i \( \leftarrow i + K \)

- \( i \geq (N - K)? \)
  - no \( \rightarrow \) \( \text{no} \)
  - yes \( \rightarrow \) END

- Retrieve rec \( i + 1 \)
- Decrypt rec \( i + 1 \)
- Search rec \( i + 1 \)

\( \cdots \)

- Retrieve rec \( i + K - 1 \)
- Decrypt rec \( i + K - 1 \)
- Search rec \( i + K - 1 \)

In cloud environment, \( K \) can be 1,000 or even 10,000 \( \rightarrow \) fast searching

Requires \( N/K \) iterations
Operational Details
Operational Details - Topics

- Accessing Individual Records
- Granting Record Access Permission
- Searching Process
  - Central Control
  - Cloud Search Server Process
Personal Grid User Files (1 of 2)

- User partial decryption key (U-KEY) [row for each accessible patient record]
  - \( \{ p, E_{\text{key}(u)} \alpha_p \} \) where
    - \( p \) = unique patient record identifier
    - \( E \) = symmetric encryption function
    - \( \text{key}(u) \) = encryption key supplied by user \( u \)
    - \( \alpha_p \) = user partial encryption key for patient record \( p \)
Personal Grid System Files (1 of 2)

- System Partial Decryption Key Table (S-KEY) [row for each patient record]
  - \( \{ p, \text{date}, E_{\text{key(date)}}\beta_p \} \) where
    - \( p \) = unique patient record identifier
    - \( \text{date} \) = calendar date when \( \beta_p \) encrypted
    - \( E \) = symmetric encryption function
    - \( \text{key(date)} \) = encryption key for (date) (from D-KEY)
    - \( \beta_p \) = system partial encryption key \( \beta \) for patient record \( p \)
Personal Grid System Files (2 of 2)

- Date Specific Decryption Key Table (D-KEY) [row for each day]
  - \( \{ \text{date, } E_{\text{key(master)}} \text{key(date)} \} \) where
    - date = calendar date
    - \( E \) = symmetric encryption function
    - key(master) = master system encryption key
    - key(date) = encryption key for (date)

- Master decryption key (M-KEY)
  - \( \{ \text{master system encryption key} \} \)
Individual Patient Record Access

User Process

- **Input**
  - User u provides key(u)
  - User requests access to record p

- **Process**
  - U-KEY table row with p found (if not found, access denied)
  - key(u) used to decrypt α_p

- **Output**
  - Record Access Request (user u, record p, α_p)

- **Result**
  - System returns decrypted record p to user u workspace

System Process

- **Input**
  - Record Access Request (user u, record p, α_p)

- **Process**
  - S-KEY row p found
  - key(date) retrieved from D-KEY
  - key(date) used to decrypt β_p
  - α_p and β_p used to decrypt record p

- **Output**
  - Decrypted record p returned to user u workspace
Personal Grid User Files (2 of 2)

- User Permission Access Pending Table (PEND-TAB) [row for each record with pending access permission]
  - \{ p, date, E_{key(date)}\alpha_p \} where
    - p = unique patient record identifier
    - date = calendar date when \alpha_p encrypted
    - E = symmetric encryption function
    - key(date) = encryption key for (date) (from D-KEY)
    - \alpha_p = user partial encryption key \alpha for patient record p
Granting Access Permission

User Processes

- **Input** *(User u login)*
  - User u owns record p, wants to grant access to user u*

- **Process**
  - Access Permission Request *(p, u*, α_p)*

- **Input** *(User u* login)*
  - PEND-TAB

- **Process**
  - Decrypt \( α_p \) (system call)
  - Encrypt \( α_p \) with key(u*)

- **Output**
  - \( \{ p, E_{key(u*)}α_p \} \) added to U-KEY

- **Result**
  - User u* has access to record p

System Process

- **Input**
  - Access Permission Request *(p, u*, α_p)*

- **Process**
  - (date) and system master key used to retrieve key(date) from D-KEY
  - \( α_p \) encrypted with key(date)

- **Output**
  - date, \( E_{key(date)}α_p \)

- **Result**
  - System appends row to PEND-TAB in user u* workspace
  - \( \{ p, date, E_{key(date)}α_p \} \)
Search Process: Files

1. Search Server Login Credentials Table (Q-LOGIN) [row for each search server]
   - \(\{\text{AE}_{\text{key}(\text{MS-PUB})} \text{(key } Q_n\text{)}\}\) where
     - \(\text{AE} = \) asymmetric encryption function
     - \(\text{key}(\text{MS-PUB}) = \) master search public key
     - \(\text{key } Q_n = \) login password for search server \((n)\) [randomly generated]

2. Master Search Public Key (MS-PUB)

3. Master Search Private Key (2 parts)
   - MS-PVT1 – on secure external USB 1
   - MS-PVT2 – on secure external USB 2
Search Process: Controller

- **Input**
  - Search parameters specified
  - External USB drives for MS-PVT1 and MS-PVT2 connected (two separate search administrators)
  - Search initiation command issued by both search administrators

- **Process (for each row of Q-LOGIN)**
  - Use MS-PVT1 and MS-PVT2 to decrypt search server password
  - Allocate search server processor
  - Initiate “Record Access Request” system process on allocated search server
  - Send username, password Q_n, and search parameters to search server

- **Output**
  - Receive results from search servers; accumulate

- **Result**
  - Completed search
Search Process: Search Server

- **Input**
  - Login credentials (username and password $Q_n$)
  - Search parameters

- **Process (for each row of U-KEY)**
  [accessible records for this search server user]
  - Record Access Request (processed by local system-level process)
  - Perform search of decrypted record
  - Accumulate search results
  - Erase decrypted record

- **Output**
  - Send accumulated results to search controller

- **Result**
  - Completed search for records assigned to this search server
Personal Grid Characteristics
## Personal Grid Vulnerabilities

<table>
<thead>
<tr>
<th>ENTRY PT</th>
<th>COMPROMISE</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Keys</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User key (\alpha_p)</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td>System key (\beta_p)</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td>System master encryption key</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td>Master search private key</td>
<td></td>
<td>Need both for access</td>
</tr>
<tr>
<td><strong>2. Credentials</strong></td>
<td>User</td>
<td>Access to user’s records</td>
</tr>
<tr>
<td>Search server</td>
<td></td>
<td>Small # of records</td>
</tr>
<tr>
<td>System administrator</td>
<td></td>
<td>No direct access</td>
</tr>
<tr>
<td><strong>3. Files</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User key table</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td>User permission pending table</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td>System key table</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td>Date-specific decryption table</td>
<td></td>
<td>No access</td>
</tr>
<tr>
<td><strong>4. Search</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search admin credentials</td>
<td></td>
<td>Need 2 sets and USBs</td>
</tr>
<tr>
<td>Search server penetrated</td>
<td></td>
<td>Small # of records</td>
</tr>
</tbody>
</table>
Population Search Times

Search Time vs. # of Servers for Various Populations
[Search speed = 25 records/second]

- Minutes
- Number of Parallel Search Servers
- Population
  - 30,000,000
  - 20,000,000
  - 10,000,000
  - 5,000,000
  - 1,000,000
Search Times vs. Complexity

Search Time vs. # of Servers for Various Search Speeds
[Population = 10,000,000]

Minutes

Records searched/second

Number of Parallel Search Servers

© 2016
## Relational DB vs. Personal Grid

<table>
<thead>
<tr>
<th>CHARACTERISTIC</th>
<th>RELATIONAL DB</th>
<th>PERSONAL GRID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>Twice the size of data or more (data &amp; indices)</td>
<td>Data only (plus optional indices of each record)</td>
</tr>
<tr>
<td>Time for adding/updating</td>
<td>Update data &amp; indices</td>
<td>Update data only (and optional indices if used)</td>
</tr>
<tr>
<td>Search efficiency</td>
<td>Excellent</td>
<td>Acceptable with large # of virtual servers</td>
</tr>
<tr>
<td>Search flexibility</td>
<td>Excellent</td>
<td>Acceptable for patient data searches</td>
</tr>
<tr>
<td>Capacity</td>
<td>Limited only by storage</td>
<td>Searching may be too slow for populations &gt; 30 million</td>
</tr>
<tr>
<td>Security</td>
<td>Susceptible to total data loss from one incident</td>
<td>Risk of loss limited to few records</td>
</tr>
</tbody>
</table>
Personal Grid Architecture

- Each patient’s record stored in separate file with separate encryption
- Efficient searching using highly parallel virtual processors in cloud and/or network (which may include mobile phones)
- No access point for all patients’ data – even for operator of service
  - Eliminates risk of loss of entire dataset
- May be practical and useful alternative architecture for centralized patient-centric health record systems
Questions?

Blog with link to full text of paper: [http://www.yasnoff.com](http://www.yasnoff.com)

William A. Yasnoff, MD, PhD, FACMI
william.yasnoff@nhiiadvisors.com
703/527-5678