Privacy Preserving Enforcement of Sensitive Policies in Distributed Environments

Muhammad Rizwan Asghar

Researcher
CREATE-NET
Italy

Saarbrücken, Germany
March 20, 2014
Growth of Smartphones

Number of Smartphone Buyers

Source: EMarketer, April 2013
What is an Opportunistic Network?

- A network where nodes connect *intermittently* and communicate even when no direct path exists
- It enables content exchange in a pub-sub fashion
  - Publishers publish content
  - Subscribers express interest
  - Brokers disseminate and match interest and content
- Typically *short-range* communication
- E.g., Haggle (an EU project from 2006 to 2010)
- DARPA - Content-Based Mobile Edge Networking (CBMEN)
Use case Scenario: Curiosity – A Military Mission

- **No Internet** connectivity in the battlefield
- Every Soldier is equipped with a **smartphone**
- A Scout collects and **shares** sensitive information
  - For instance, enemy positioning
- Only **short-range** communication is possible
- We can leverage opportunistic networks
  - such as **Haggle**
Privacy and Confidentiality Issues

- Brokers (or attackers) may easily learn
  - interest of subscribers
    - privacy issue
  - published content
    - confidentiality issue
Research Challenges

- **C1**: In the presence of unauthorised brokers, how to regulate access to disseminated content?

- **C2**: Considering curious brokers, how to exchange content without compromising privacy of subscribers?

- **C3**: How can subscribers subscribe without exposing interest to routing brokers?

- **C4**: For avoiding network flooding, how do we ensure that a subscriber receives content that she can decrypt?

- **C5**: Assuming the loosely-coupled pub-sub model, how to address C1-C4 without sharing keys?
Threat Model

- Honest but curious brokers
- Nodes may collude
  - Broker-broker collusion
  - Broker-subscriber collusion
  - Subscriber-subscriber collusion
- Trusted key management authority
  - distributes key material to nodes out of the band
  - can stay offline
- Passive adversaries
CP-ABE Policy: Building Blocks

- Ciphertext-Policy Attribute-Based Encryption (CP-ABE)
- Data encrypting entity exerts control over who can gain access
- E.g., a Major or a Soldier from the Infantry unit can get access

\[ P = \text{Major} \lor \text{Soldier} \land \text{Infantry} \]

\{Content\}_P
Scheme I: Regulate Access to Content

- Publishers encrypt content using **CP-ABE** policies
- Subscribers may decrypt if they satisfy policies
- It regulates access to content (C1)
- Issue: subscribers may receive content that they cannot decrypt – the network flooding issue (C4)

**Attributes** = {'Soldier’}

**Interest** = {'Curiosity’}

**Broker**

**Publisher**

**P** + **P** + **P**

**Tags** = {'Curiosity'}

**Major**

**Soldier**

**Infantry**

**No**
Scheme II: Authorisation Check

- Subscribers send **attributes** along with interest
- Brokers forward content if attributes satisfy policy, as well as interest matches with content
- It resolves **the network flooding issue (C4)**
- Issue: cleartext interest, attributes and policy leak privacy of subscribers (**C2 & C3**)

![Diagram showing Scheme II: Authorisation Check](image-url)
Scheme III: Hiding Private Information using a Hash

- Replace cleartext elements with hash
- Brokers matches hash values
- Issue: pre-computed dictionary attack

\[
\text{Interest} = \{H('Curiosity')\}
\]

\[
\text{Attributes} = \{H('Soldier'), H('Infantry')\}
\]

\[
P' = P + H('Major')
\]

\[
\text{Tags} = \{H('Curiosity')\}
\]
Scheme IV: Harden against a Pre-computed Dictionary Attack

- Publishers replace each leave node with a hash of concatenated pair of a tag and an attribute
- Subscribers subscribe using the hash of a concatenated pair of an interest item and an attribute
- It decreases number of comparisons at brokers
- Issue: still vulnerable to a pre-computed dictionary attack

$$\text{Subscription} = \{ \text{H('Curiosity' || 'Soldier')}, \text{H('Curiosity' || 'Infantry')} \}$$

$$P' = \text{H('Curiosity' || 'Major')} \Lambda \text{H('Curiosity' || 'Soldier')} \Lambda \text{H('Curiosity' || 'Infantry')}$$
PEKS: Building Blocks

- Public-key Encryption with Keyword Search (PEKS) contains four algorithms
  - **Keygen** generates public ($h_{\text{Soldier}}$) and private ($x_{\text{Soldier}}$) keys
  - **Etag** encrypts tag given a public key
  - **Trapdoor** transforms a keyword into trapdoor using a private key
  - **Test** checks whether an encrypted tag matches with the trapdoor

- It performs encrypted matching without revealing plaintext values
Proposed Scheme: PIDGIN

- **PIDGIN:** Privacy Preserving Interest and content sharing in opportunistic Networks [Asghar et al. ASIACCS’14]
- The main idea is to employ PEKS for protecting policies, tags and subscriptions (C2 & C3)
- Publishers encrypt leaf nodes in a policy using Etag
- Subscribers protect subscription using Trapdoor
- Brokers perform matching using Test

\[ \text{Subscription} = \{ \text{Trapdoor('Curiosity', } x_{\text{Soldier}}), \text{Trapdoor('Curiosity', } x_{\text{Infantry}}) \} \]

Publisher \( P \) + \( P' \)

Broker

Subscriber

\[ \Lambda \]

Etag('Curiosity', \( h_{\text{Major}} \))

Etag('Curiosity', \( h_{\text{Soldier}} \))

Etag('Curiosity', \( h_{\text{Infantry}} \))
Complex Policies

- Policy with multiple tags
- E.g., ‘Curiosity’ and ‘Urgent’
**PIDGIN – Implementation Details**

- We **developed** a prototype of PIDGIN in **C**
  - Using open source libraries: libfenc and pbc
- We **tested** PIDGIN on Samsung **Galaxy S III**
  - Cross-compiled gmp, pbc, libfenc and PIDGIN
  - Ported libraries and binaries on smartphone
- Content is encrypted with a symmetric key
- Symmetric key is encrypted under a policy
- Policy is encrypted using PEKS
PIDGIN – Overhead

- Publisher’s encryption incurs < 0.3 s
- Subscriber’s encryption incurs < 0.04 s
- Broker’s matching takes ~0.04 s
- Subscriber’s decryption takes < 0.05 s

- We considered
  - Content: 200 KB file
  - Policy: (Soldier ∧ Infantry) V Major
  - Attributes: {Soldier, Infantry}
  - Tags/Interest items: {Curiosity}

- We ran PIDGIN on Samsung Galaxy SIII
  - Operating system: Android 4.1.2
  - Processor: 1.4 GHz
  - RAM: 1 GB
Evaluation: Key Generation

- Key generation authority generates search and decryption keys

- Complexity
  - Linear
  - \( O(|\text{Attributes}|) \)
Evaluation: Content Encryption and Decryption

- Encryption and decryption of content using a symmetric key

- Complexity
  - Linear
  - $O(|\text{Content}|)$

![Graph showing AES encryption and decryption time vs. content size]
Performance Analysis: Publisher’s Encryption

- Encrypting symmetric key with policy and then extending policy with tags
  - Each Etag is of 256 bytes

- Complexity
  - Quadratic
  - $O(|\text{Tags}| \times |\text{Attributes-Pub}|)$
Performance Analysis: Subscriber’s Encryption

- Effect of number of interest items and attributes on subscriber’s encryption time
  - Each Trapdoor of interest item/attribute is of 128 Bytes

- Complexity
  - Quadratic
  - $O\left( |\text{Interest-Items}| \times |\text{Attributes-Sub}| \right)$
Performance Analysis: Broker’s Matching

- Effect of number of interest items and tags on broker’s matching time

- Complexity
  - $O \left( |\text{Tags}| \times |\text{Interest-Items}| \times |\text{Attributes-Pub}| \times |\text{Attributes-Sub}| \right)$
Related Work

- Search on encrypted data
  - Symmetric encryption schemes are not suitable in opportunistic environments
  - Public-key encryption schemes do not support expressive policies

- Attribute-Based Encryption (ABE) support expressive access control policies
  - CP-ABE and KP-ABE reveal policies and attributes, respectively

- Predicate encryption and hidden vector schemes assume end-to-end communication

- Shikfa et al. propose content dissemination in opportunistic networks
  - Only uni-directional communication from publishers
Discussion

- **Optimisation**
  - Short-circuit evaluation

- **Scalability**
  - Time to live
  - Content creation time
  - Content received time

- **Key management**
  - Deployment in practical scenarios
  - Distributed authorities
Summary

- We proposed **PIDGIN** that regulates access to content
- In PIDGIN, brokers **enforce sensitive policies** without compromising privacy of subscribers
- Publishers and subscribers **do not share** keys
- We **implemented** a prototype and measured performance by running on Samsung **Galaxy SIII**
- It can be applied to a number of other application scenarios, e.g.,
  - Reporting and controlling crimes
  - Offloading content delivery networks