Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching

Shirin Nilizadeh[*], Sonia Jahid[+], Prateek Mittal[+], Nikita Borisov[+], Apu Kapadia[*]
[*]Indiana University Bloomington, [+]University of Illinois at Urbana-Champaign

Online Social Networks (OSNs)

❖ Online social networks have emerged as significant social and technical phenomena over the last several years.

➢ Facebook has grown beyond 900 million monthly active users,

➢ Google+ reached the mark of 10 million users in only 2 weeks after going public.

❖ The lack of user privacy:
- ➢ Users are not in control of their private data,
- ➢ OSN operators gather an extensive amount of information about users,
- ➢ As single points of failure, any vulnerability in these systems (or even accidental leaks) can be exploited by a malicious adversary to obtain user data,
- ➢ Not fine-grained access controls can be defined on users' data.

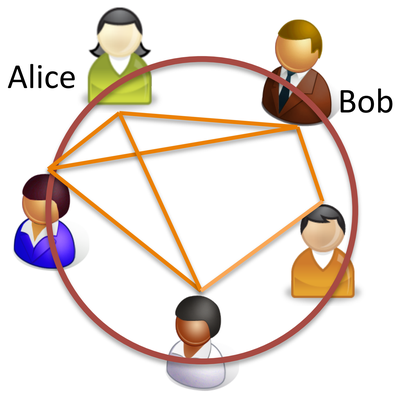
Distributed P2P Networks

❖ Providing the same functionalities as OSNs in P2P networks is challenging and raises entirely new privacy concerns:
- ➢ Not everyone in P2P networks is trustworthy,
- ➢ Network traffic is sometimes interpreted as hostile.

❖ On the other hand, one approach to mediate security and privacy concerns in P2P networks is to leverage trusted social links between users.

❖ P2P paradigm and social networks mutually can improve one another's efficiency, security, and privacy.
- ➢Using P2P architecture for social networks increases privacy and anonymity and
- ➢ Using social networks concepts to construct P2P networks creates more trust between users.

❖ Considering a hybrid structured-unstructured overlay:
- ➢ The distributed hash table (DHT) is used as a base storage layer
- ➢ A gossip-based social caching algorithm dramatically increases performance.



Security Requirements

❖ Confidentiality and integrity of user data,
❖ Users have complete control over the permissions to content they create
❖ No user accesses content unless explicitly authorized by the owner
❖ User relationships should remain hidden from third parties, such as the storage nodes.

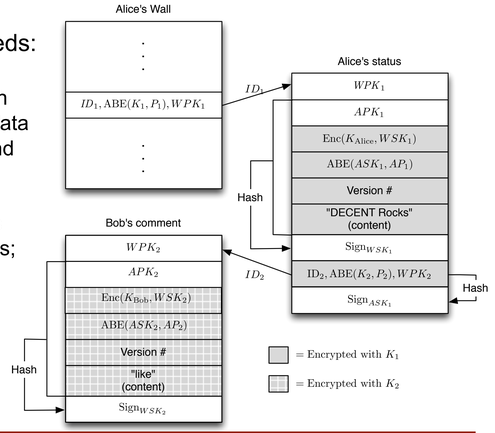
Base Architecture

❖ **Attribute-based policies:** e.g., *friends AND family OR colleagues*.

❖ **Cryptographic Protection**:
- ➢ Hybrid encryption scheme:
  - ✓ traditional public key and attribute-based encryption (ABE).
- ➢ Attribute-based encryption
  - ✓ Each of the social contacts is issued a different secret attribute-key defining what attributes that person contains.
  - ✓ An object is encrypted with an attribute-based policy.
  - ✓ A person can decrypt an object if and only if her secret key satisfies the policy used to encrypt it.

❖ Downloading the newsfeed needs:

1) Decrypting update objects, which are ABEncrypted, to yield metadata such as an update's DHT key and symmetric decryption key;
2) Accessing multiple small objects located in different storage nodes;
3) decrypting the retrieved update objects with their corresponding symmetric keys.



Social Caching

❖ **Goal:** progressively retrieve cached, unencrypted versions of objects to greatly speed up the process of loading the newsfeed or wall.

❖ **The basic idea:** use of social links between users who act as caches to store unencrypted objects recently seen in the social network.

❖ **Presence Protocol**
- ➢ Uses social caching for finding online contacts.
- ➢ Online social contacts provide cached, decrypted objects to other contacts who also satisfy the policy for presence objects related to offline contacts.
- ➢ Minimize the number of decryptions by dynamically learning which peers yield the most cached objects.

❖ **Gossip-based Social Caching**
1) Creating the presence table
2) Selecting a contact
3) DHT lookup
4) Pulling information
5) Caching information
6) Updating presence table
7) Performing DHT lookups for offline social contacts with no mutual social contacts

Algorithm 1: User P joins the network

```
1
2  //User P joins the network
3  generatePresenceTable(table);
4  socialCachingAlg(table, cache);
5  for(social contact Q : table.keySet()){
6    if(!cache.contains(Q.update)){
7      getDHTKeyFor(Q.update);
8      encUpdate = dhtLookUp(Q, Q.updateObj);
9      update = dycrypt(encUpdate);
10     cache.put(Q, update);
11   }
12 }
```

Algorithm 2: Social caching algorithm

```
1
2  void socialCachingAlg(presenceTable table,
3   Cache cache){
4    for(SocialContact Q : table.keySet()){
5      Q.visited = TRUE;
6      dhtLookUp(Q, Q.presenceObj);
7      if(Q.presence.status){
8          sendTo(Q, Q.presenceObj);
9          receiveMessageFrom(Q, bufr);
10         if(bufr.contains(presenceObj))
11             updateTable(table, bufr);
12         if(bufr.contains(UpdateObj))
13             selectUpdatesToKeep(cache, bufr);
14     }
15     SocialContact R = selectSocialContact(&table);
16     socialCachingAlg(R, table);
17 }
```

➢ Cached content is stored unencrypted
➢ An explicit list of authorized users is included in each container that can be used to mediate sharing.

Implementation and simulation

❖ A simulator for Cachet based on the FreePastry simulator
❖ To simulate the social graph, we used the Facebook friendship graph[1]:
- ➢ 63,732 nodes,
- ➢ 1.54 million edges.

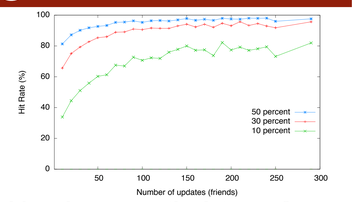
❖ Considering different percentages of users amongst **P** 's social contacts that remain online — 10%, 30% and 50%.

❖ **Performance metrics:**
- ➢ Hit Rate: the percentage of objects that has been provided by social contacts.
- ➢ Progressive Hit Rate: the percentage of objects that have been obtained after d DHT lookups and pulling social contacts' cached objects.

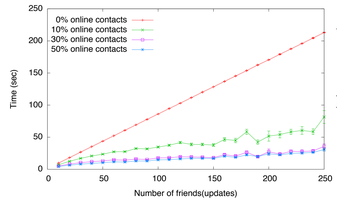
Results

❖ **Social caching provides most of the update objects for viewing the newsfeed**

➢ It depicts the average Newsfeed Hit Rate as a function of number of updates and the fraction of online social contacts.



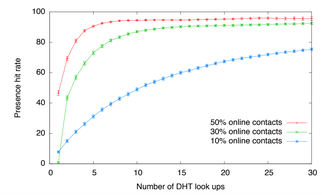
❖ **Social caching decreases the latency for retrieving the newsfeed**



➢ The simulation time includes the time for:
1) ABDecrypting the references to both presence and update objects that are not provided by social contacts,
2) Performing DHT lookups for retrieving objects,
3) Decrypting the objects.

❖ **Most of the presence objects would be available after a few DHT lookups and decryptions**

➢ It plots the Average Progressive Hit Rate after d DHT lookups and ABDecryptions for users with $100 \leq m \leq 200$ where m is one's number of social contacts.



Future Work

❖ Searching social contacts
❖ Privacy issues:
1) Users not be aware that they are being excluded from accessing the object.
2) Avoid leakage of information about the identities of users who satisfy a particular policy to all of those identities.
3) Avoid revealing information about when a user comes online or offline.

Acknowledgments

I would like to thank to my supervisor Professor Apu Kapadia for the valuable guidance and advice. I give special thanks to all the co-authors of the Cachet paper [2, 3]: Professor Nikita Borisov, Prateek Mittal, Sonia Jahid.

References

[1] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In Proceedings WOSN'09, August 2009.
[2] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia. DECENT: A decentralized architecture for enforcing privacy in online social networks. In Proceedings SESOC, 2012.
[3] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia. Cachet: A decentralized architecture for privacy preserving social networking with caching. To appear in CoNEXT, 2012.

INDIANA UNIVERSITY