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.
  - Not everyone in P2P networks is trustworthy.
  - Not everyone in P2P networks is trustworthy.
- 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.

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 to construct P2P networks creates more trust between users.
- Considering a hybrid-structured-unnstructed overlay:
  - The distributed hash table (DHT) is used as a base storage layer

Security Requirements

- Attribute-based policies: e.g., friends AND family OR colleagues.
- Cryptographic Protection:
  - Hybrid encryption scheme: public traditional 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:
  1) Decrypting update objects, which are ABEencrypted, 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.
- 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
  - Selecting a contact
  - DHT lookup
  - Pulling information
  - Caching information
  - Updating presence table
  - Performing DHT lookups for offline social contacts with no mutual social contacts

Social Caching

- Caching information
- Pulling information
- Updating presence table
- Performing DHT lookups for offline social contacts with no mutual social contacts

Algorithm 1: User P joins the network

```
1 //User P joins the network
2 generatePresenceTable(table); 1
3 socialCachingAlg(GenerationId, table, socialCachingAlg(Id, table, K, n)); 2
4 socialCachingAlg(Id, table, K, n); 3
5 if (set of shared presence objects) 
6 setPresenceTable(SharedPresenceTable.table); 4
7 getPresenceTable(table); 5
8 update = Decrypt(table); 6
9 if (set of mutual social contacts) 
10 socialCachingAlg(table, K, n); 7
11 }socialCachingAlg(table, K, n); 8
12 }
```

Algorithm 2: Social caching algorithm

```
1 void socialCachingAlg([presenceTable table], 
2 Cacher=cacher, 
3 DHT lookups=(Q, Q. updateObj)); 
4 if (set of shared presence objects) 
5 setPresenceTable(SharedPresenceTable.table); 
6 getPresenceTable(table); 
7 update = Decrypt(table); 
8 if (set of mutual social contacts) 
9 socialCachingAlg(table, K, n); 
10 SocialContact R = selectSocialContact(Inputtable); 
11 socialCachingAlg(R, table); 
12 }
```

### References

