A decentralized approach to publishing confidential data

Master Thesis Presentation 14.07.2009

Konstantin Welke

Professorship for Communication Systems
Department of Computer Science
Albert Ludwigs University of Freiburg

Outline

1. Motivation
2. Cryptography
3. The Protocol
4. Conclusion

Outline

1. Motivation
2. Cryptography
3. The Protocol
4. Conclusion

Motivation

- This is Bob
- Bob hosted a party at his house yesterday
- He made some pictures to share with his friends
- How can he do that?

Motivation

- This is Alice
- Alice maintains an electronic calendar
- Some appointments should be visible
  - Only to her co-workers
  - Only to her friends
  - Only to her family
  - Only to herself
- How can she do that?

Motivation

- How can Bob share his pictures with his friends?
- How can Alice share appointments with defined groups of peers?
- **But no one else!**
  1. Social Networks / Web Applications?
  2. Just set up a server?
  3. Upload an encrypted tar archive?

The Confidential Publishing Problem

- More general: How can people share content with peers over third-party infrastructure, with
  - Confidentiality
  - Integrity
  - Authenticity
  - Decentralized structure
  - Usability
- How to achieve that?

Outline

1. Motivation
2. Cryptography
3. The Protocol
4. Conclusion

How to solve the confidential publishing problem

- Confidentiality
  - Exchange symmetric key using public-key crypto
  - Encrypt using symmetric crypto
  - Problem: How to encrypt for n people?
- Integrity & Authenticity
  - Hash using one-way hash function
  - Sign using public-key crypto

How to solve the confidential publishing problem

- Decentralized structure
  - Just like "traditional internet services"
  - user@server
- Usability
  - Easy to use & to try out
    - Web application
  - User friendly crypto?
    - Problem: How to verify public-key fingerprints?

How to encrypt for n people?

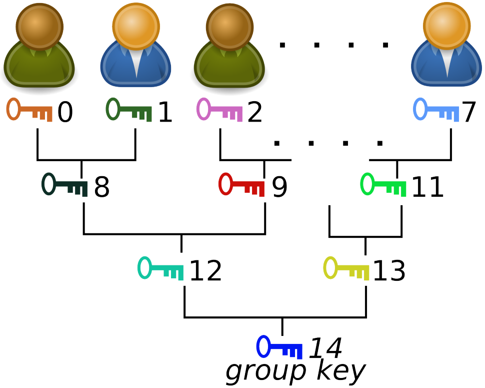
- Crypto aims at one-to-one encryption
- How to encrypt for n people?
- Naive approach
  - Generate symmetric key k
  - Share k with set of people U
  - If user u joins:
    - Give k to u
    - Constant cost
  - If user u is removed:
    - Generate new key k'
    - Share k' with $U \setminus u$
    - Linear cost

How to encrypt for n people?

- Can we do better than linear removal cost?
- Idea: Build a tree (Wallner et al. [1999], Wong et al. [2000])

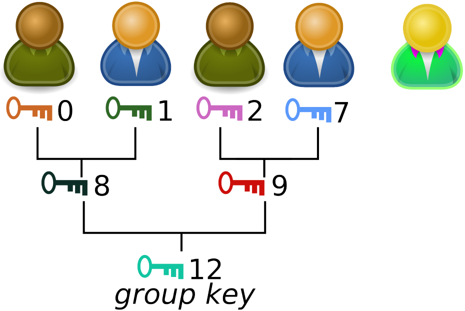
Key Trees

- User knows exactly the keys from their leaf key to the group key



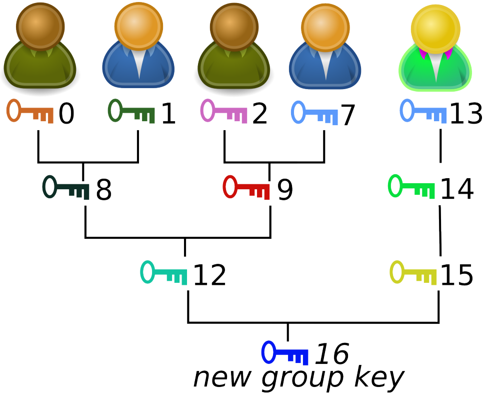
Adding a user

- Create new leaf for the user, send leaf to user
- If tree is full: create new group key



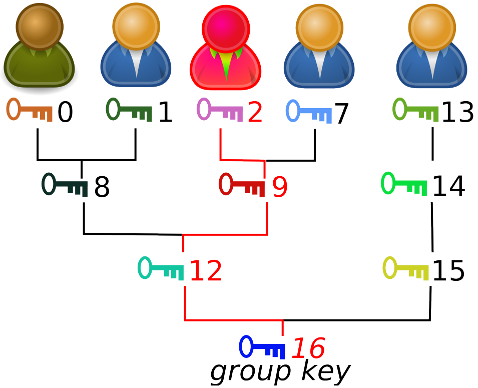
Adding a user

- Create new leaf for the user, send leaf to user
- If tree is full: create new group key



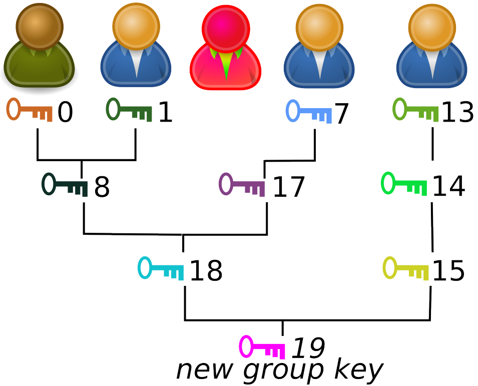
Removing a user

- All keys that the removed user knows need to be exchanged with new keys



Removing a user

- All keys that the removed user knows need to be exchanged with new keys



Efficiency

- Naive approach
  - Constant cost to add
  - Linear cost to remove
- Key Tree
  - Amortized constant cost to add
  - Logarithmic cost to remove

Public-key Fingerprints

- Users need to verify public-key fingerprints to detect man-in-the-middle-attacks
    - Real-life example: SSH
- Are these the same fingerprint?
  c1:b1:30:29:d7:b8:de:6c:97:77:10:d7:46:41:63:87

  c1:b1:30:29:d7:b8:de:6c:77:97:10:d7:46:41:63:87
- User friendly?

Human-Readable Fingerprints

- Introduced by Bååth and Kühn [2002], Kaminsky [2006] independently
- Generate human-readable sentences from the fingerprint
- Are these the same fingerprint?

Happy:
Lisa:chased:the cookie:
Savah:jumped over:the flower:
Chip:smiled at:the world:
Nacho:joked about:the pizza

Happy:
Lisa:chased:the cookie:
Savah:jumped over:the flower:
Chip:smiled at:the pizza:
Nacho:joked about:the world

Outline

1 Motivation

2 Cryptography

3 The Protocol

4 Conclusion

The Protocol

- User should be able to publish content to a set of peers with
  - Confidentiality
  - Integrity
  - Authenticity
  - Decentralized structure
  - Usability
- We covered the cryptography that is needed
- Let's describe the protocol!

Login

- Every user has a passphrase
- Generate symmetric key from passphrase
- Private content is encrypted with that key
    - E.g. the user's private key

Short-term passphrases

- User has *one main* passphrase
  - Encrypts his long-term public key
  - Valid for e.g. 5 years
- Problem: Should the user enter that passphrase at an Internet cafe?
- Solution: *Short-term* passphrases
  - Valid for a short period of time
  - Encrypt a *short-term public key*
  - Can only access a subset of data
  - Only that data can be compromised
  - Usable?

Public keys

- Folder “/public-keys”
  - Contains all public keys
  - Publicly accessible
- Public keys have validity period
- *Key messages*
  - Signed by preceeding key:
    - Introduce new keys
    - Declare old keys compromised
  - *Not* signed:
    - Declare old keys “lost”

Content Files

- Content is organized into folders and files
- *Content files* consists of
  - Actual (encrypted) content
  - Revision number
  - Signature
  - Key to access the content
    - Encrypted for every user/group that may access the content
- One content file can consist of multiple filesystem files
  - E.g. one containing content, another containing the signature
- Folders are just a "list of files"

File names

- Problem: Files names carry potentially confidential information
  - “Bob totally drunk”
  - “Alice’s doctor appointment”
- Solution: Generate *public filename(s)* from *private name*
  - Every folder defines a *salt*
  - Public name: Hash of salt + user id + namespace + private name
  - Namespace: e.g. content, revision, ...
    - If content file is split over multiple filesystem files
    - Avoids name clashes

File names

- Example:
  - Salt: “ssssssssssssssssssssssssssssssss”
  - Public name: “Alice’s doctor appointment”
  - Namespace: “content”
  - Private name:
    SHA-256(“ssssssssssssssssssssssssssssssss”
    + “alice@crypto.uni-freiburg.de:”
    + “content:/”)
    + “Alice’s doctor appointment”) =

`1afac6c112b4b5a1a9f43ff8445fdebc43e00c41078edfbad3b6d533e5efc15f`

- Hard to guess private name from public name

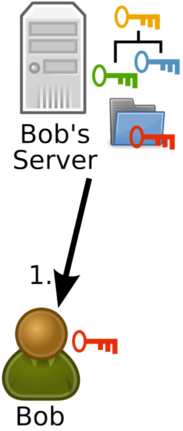
Publishing Content

- Users organize their contacts into groups
    - Every group = one key tree
- When publishing a content file, user can set which users/groups have read access

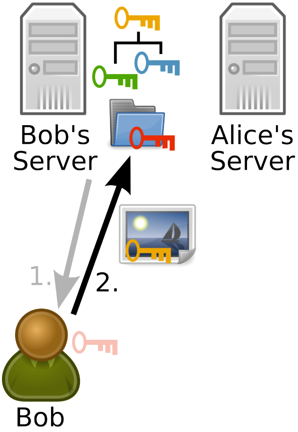
Social Network?

- *Privacy-respecting* social network becomes possible
- User maintain groups of contacts
  - Only visible to the group
- User publishes content
  - Select target audience
  - Not visible to service provider
    - Or any third party
- Also possible:
  - Messaging
  - “Wall”
  - Discussion groups
  - Tagging

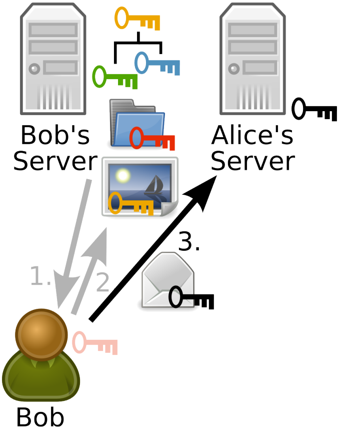
Example



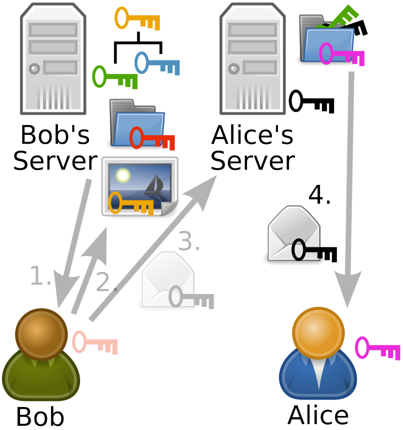
Example



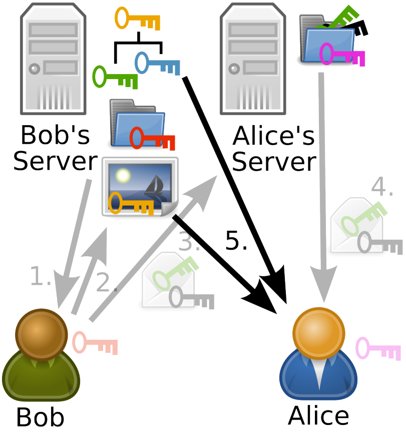
Example



Example



Example



Outline

1. Motivation
2. Cryptography
3. The Protocol
4. Conclusion

Conclusion

- Protocol solves confidential publishing problem
- Users can publish data over third-party infrastructure
  - Without revealing the data to the infrastructure
  - One-to-many encryption is cruicial
- Web application prototype
  - Incomplete (only single-user)

That's all folks!

Thank you for your attention!

Do you have any questions?

... & Demo