A Security API for Distributed Social Networks

Kim Pecina

joint work with
Michael Backes and Matteo Maffei

Network and Distributed Systems Security Symposium 2011
San Diego, California
Social Networks

Vast number of users:
- Facebook: 500 million
- twitter: 200 million
- myspace: 60 million
- ...

Huge amounts of data in the hands of a few social networks
- Copyright issues
- Privacy issues

Reports claim that Facebook silently gave profile access to Italian police
Distributed Social Networks help ...

- User data not entrusted to third parties
  - Not a single point of failure
  - User data remains under user control
... but help only partially!

We also need other security properties, such as anonymity, privacy of social relations, and coercion-resistance:

**WIRED MAGAZINE: 16.11**

**Cairo Activists Use Facebook to Rattle Regime**

The regime strikes back and tortures leading activist to get Facebook password
Our Contribution

- Cryptographic API providing
  - Fine-grained access control
  - Anonymity
  - Privacy of social relations
  - Flavor of coercion resistance
- API also applicable in centralized settings
- Formal verification of all API methods
- Experimental Evaluation
Facebook
Facebook

Friend Requests

Bob
17 mutual friends

Confirm  Not Now

Monday, February 7, 2011
Request is checked against ACL
Resource released if check against ACL succeeds
Our Approach: Decentralized Setting

Alice

Friend Requests
Bob
17 mutual friends
Confirm Not Now

Notifications
Alice accepted your friend request.
Our Approach: Decentralized Setting

Friend Requests
Bob
17 mutual friends
Confirm Not Now

Notifications
Alice accepted your friend request.
Our Approach: Decentralized Setting

I am Bob.
Please befriend me.

Bob
17 mutual friends

Alice accepted your friend request.

Monday, February 7, 2011
Our Approach: Decentralized Setting

I am Bob.
Please befriend me.

Friend Requests
Bob
17 mutual friends
Confirm Not Now

Notifications
Alice accepted your friend request.
Our Approach: Decentralized Setting

I am Bob.
Please befriend me.

Hi Bob, you are my friend.

Notifications
Alice accepted your friend request.
Our Approach: Decentralized Setting

We deploy certificates to establish authenticity in decentralized setting

I am Bob.
Please befriend me.

Hi Bob, you are my friend.

- We deploy certificates to establish authenticity in decentralized setting
Certificates realized via digital signatures
[Camenisch and Lysyanskaya, SCN’02]

- Can be publicly verified
- Cannot be forged

\(\text{cert}_A(m)\) denotes A’s certificate on \(m\)
Certificates

- Certificates realized via digital signatures
  [Camenisch and Lysyanskaya, SCN’02]
  - Can be publicly verified
  - Cannot be forged

- \( \text{cert}_{\text{Bob}}(\text{Please befriend me}) \)

- \( \text{cert}_A(m) \) denotes A’s certificate on m
Certificates realized via digital signatures
[Camenisch and Lysyanskaya, SCN’02]
  • Can be publicly verified
  • Cannot be forged

\[\text{cert}_A(m) \text{ denotes } A\text{’s certificate on } m\]
**Pseudonyms**

- cert$_{Bob}$(Please befriend me)

- cert$_{Alice}$(“friend”)
  - cert$_{Alice}$(“Bob”)

- Plain names inhibit anonymity
Plain names inhibit anonymity
- ACLs reveal social graph
Pseudonyms

Plain names inhibit anonymity
  • ACLs reveal social graph

We use pseudonyms (cf. [Pseudo-Trust, Lu et al., IPDPS’07])
Pseudonyms

- Desired properties (similar to real names):
  - One pseudonym belongs to one user
    - Impersonation / identity theft impossible
  - Pseudonyms should be trackable
    - If desired
Pseudonyms

Desired properties (similar to real names):

- One pseudonym belongs to one user
  - Impersonation / identity theft impossible
- Pseudonyms should be trackable
  - If desired
- One user may own several pseudonyms
  - Increases anonymity
  - Prevents complete tracking
Pseudonyms

Desired properties (similar to real names):

- One pseudonym belongs to one user
  - Impersonation / identity theft impossible
- Pseudonyms should be trackable
  - If desired
- One user may own several pseudonyms
  - Increases anonymity
  - Prevents complete tracking
- Do not reveal the identity of the owner
Pseudonyms

Desired properties (similar to real names):
• One pseudonym belongs to one user
  • Impersonation / identity theft impossible
• Pseudonyms should be trackable
  • If desired
• One user may own several pseudonyms
  • Increases anonymity
  • Prevents complete tracking
• Do not reveal the identity of the owner

Implemented as discrete exponentiation $g^x$ in finite groups
• DLog($g^x$) hard to compute
• Prevents impersonation
Zero-Knowledge Proofs

- Prevent impersonation using a proof of pseudonym ownership

\[ \text{cert}_{Bob} \text{(Please befriend 4711)} \]

\[ \text{cert}_{Alice} \text{("friend")} \]

\[ \text{cert}_{Alice} \text{(4711)} \]
Zero-Knowledge Proofs

\[ \text{cert}_{Bob}(\text{Please befriend 4711}) \]
\[ \text{ZK}(\exists x. g^x = 4711) \]

\[ \text{cert}_{Alice}(\text{“friend”}) \]
\[ \text{cert}_{Alice}(4711) \]

- Prevent impersonation using a proof of pseudonym ownership
- Zero-knowledge proofs [Camenisch and Lysyanskaya, SCN’02]
Zero-Knowledge Proofs

Prevent impersonation using a proof of pseudonym ownership

Zero-knowledge proofs [Camenisch and Lysyanskaya, SCN’02]
- Convince verifier (Alice)
- Cannot be forged by prover (Bob)
- Hide quantified values (zero-knowledge property)
Zero-Knowledge Proofs

cert_{Bob}(\text{Please befriend 4711})
ZK(\exists x. g^x = 4711)

cert_{Alice}(\text{“friend”})
cert_{Alice}(4711)

- Prover (Bob) must “know” all quantified values
- Verification requires only non-quantified values
Secure Storage Devices

\text{cert}_{\text{Bob}}(\text{Please befriend 4711})

\text{ZK}(\exists x. g^x = 4711)

\text{cert}_{\text{Alice}}(\text{“friend”})

\text{cert}_{\text{Alice}}(4711)

Secret values exclusively stored on secure storage device
Secure Storage Devices

Secret values exclusively stored on secure storage device

\[ \text{cert}_{\text{Bob}}(\text{Please befriend 4711}) \]
\[ \text{ZK}(\exists x. g^x = 4711) \]

chose \( x, g^x = 4711 \)

\[ \text{cert}_{\text{Alice}}(\text{“friend”}) \]
\[ \text{cert}_{\text{Alice}}(4711) \]
Secret values exclusively stored on secure storage device
Secure Storage Devices

cert_{Bob}(\text{Please befriend 4711})

ZK(\exists x. g^x = 4711)

cert_{Alice}(\text{“friend”})

cert_{Alice}(4711)

Secret values exclusively stored on secure storage device
Secure Storage Devices

\[ \text{cert}_{\text{Bob}}(\text{Please befriend 4711}) \]

\[ ZK(\exists x. g^x = 4711) \]

Bob - 4711

\[ \text{cert}_{\text{Alice}}(\text{“friend”}) \]

\[ \text{cert}_{\text{Alice}}(4711) \]

Secret values exclusively stored on secure storage device

Monday, February 7, 2011
Secure Storage Devices

\[ \text{cert}_{\text{Bob}}(\text{Please befriend 4711}) \]
\[ \text{ZK}(\exists x. g^x = 4711) \]

Bob - 4711

\[ \text{cert}_{\text{Alice}}(\text{“friend”}) \]
\[ \text{cert}_{\text{Alice}}(4711) \]

Secret values exclusively stored on secure storage device

Monday, February 7, 2011
Secure Storage Devices

\[ \text{cert}_\text{Bob}(\text{Please befriend 4711}) \]
\[ \text{ZK}(\exists x. g^x = 4711) \]

Secret values exclusively stored on secure storage device

Bob - 4711

\[ \text{cert}_\text{Alice}(\text{"friend"}) \]
\[ \text{cert}_\text{Alice}(4711) \]
Secret values exclusively stored on secure storage device

\[ \text{cert}_{\text{Bob}}(\text{Please befriend 4711}) \]
\[ \text{ZK}(\exists x. g^x = 4711) \]

\[ \text{cert}_{\text{Alice}}(\text{“friend”}) \]
\[ \text{cert}_{\text{Alice}}(4711) \]
- Knowledge of a valid certificate must be proven
- Pseudonym ownership must be proven
Knowledge of a valid certificate must be proven
Pseudonym ownership must be proven
Knowledge of a valid certificate must be proven

Pseudonym ownership must be proven
Knowledge of a valid certificate must be proven

Pseudonym ownership must be proven
Knowledge of a valid certificate must be proven

Pseudonym ownership must be proven

ZK(∃c,x. certifies(c, 4711, Alice) ∧ g^x=4711
I want to access)

cert_Alice(4711)
Retrieving Resources

ZK(∃c,x. certifies(c, 4711, Alice) \land g^x=4711
I want to access )

- Knowledge of a valid certificate must be proven
- Pseudonym ownership must be proven

Monday, February 7, 2011
Proof does not require secret input on Alice’s side
- Pseudonym-user binding

Zero-knowledge proof reveals
- Pseudonym
- Requested picture

ZK(∃c,x. certifies(c, 4711, Alice) ∧ g^x=4711

I want to access

Proof does not require secret input on Alice’s side
- Pseudonym-user binding

Zero-knowledge proof reveals
- Pseudonym
- Requested picture
Proof does not require secret input on Alice’s side

- Pseudonym-user binding

Zero-knowledge proof reveals

- Pseudonym
- Requested picture
ZK(∃c. certifies(c, “friend”, Alice)
        I want to access

Zero-knowledge proof hides the identity of the prover and only reveals the social relation between verifier and prover
I want to access

\[ ZK(\exists c. \text{certifies}(c, \text{“friend”}, \text{Alice}) ) \]

Zero-knowledge proof hides the identity of the prover and only reveals the social relation between verifier and prover.
ZK(∃c. certifies(c, “friend”, Alice))

I want to access

Zero-knowledge proof hides the identity of the prover and only reveals the social relation between verifier and prover
Zero-knowledge proof hides the identity of the prover and only reveals the social relation between verifier and prover.
Retrieving Resources: Full Protocol

Choose random key $k$

4711 read, write
Friends read
Retrieving Resources: Full Protocol

4711 read, write
Friends read

k
Retrieving Resources: Full Protocol

\[ \text{ZK}(\exists c. \text{certifies}(c, \text{"friend"}, \text{Alice}) \quad \text{I want to access}\quad \text{, k}) \]
\begin{align*}
\{ ZK(\exists c. \text{certifies}(c, \text{“friend”}, Alice) \} \\
\text{I want to access , k)} \}
\end{align*}
ZK(∃c. certifies(c, "friend", Alice)
I want to access \textbf{Friends}, k)
Retrieving Resources: Full Protocol

Language-based Security
Full protocol incorporates encryption

- Asymmetric encryption ensures data privacy
- Symmetric encryption facilitates anonymity of requester (Bob)
Resistance to Outside Attackers

\[ \left\{ \exists c. \text{certifies}(c, \text{"friend"}, \text{Alice}) \right\}_A \]

I want to access, k

\[ \left\{ \text{Network traffic looks random} \right\}_k \]
Resistance to Compromise

- Certificates on pseudonyms/social relations on secure device
- Pseudonym-user bindings stored on secure device
- Resources will be leaked
Resistance to Compromise

- Certificates on pseudonyms/social relations on secure device
- Pseudonym-user bindings stored on secure device
- Resources will be leaked
- ACL
Resistance to Compromise

- Certificates on pseudonyms/social relations on secure device
- Pseudonym-user bindings stored on secure device
- Resources will be leaked
- ACL
  - Social relations hide social graph
  - Pseudonyms can be faked and ACLs can be padded
  - De-anonymization attacks exploiting graph structure not applicable (e.g., [Narayanan and Shmatikov, S&P’09])
Resistance to Compromise

\[ \exists c. \text{certifies}(c, \text{“friend”}, \text{Alice}) \]

I want to access \[ k \]

- Zero-knowledge proofs and symmetric encryption key protect identity of requester

- A flavor of coercion resistance
  - If coerced, Alice can return fake pseudonym-user bindings and hide certain signatures while revealing the others

Monday, February 7, 2011
API Methods

- **register**
  - Acquire friends

- **getHandles**
  - Returns previews of resources (e.g., thumbnails)

- **getResources/putResources**

- **getFriends**
  - Returns friends that agreed on revealing parts of the social graph

- **indirectRegister**
  - Acquire friends of friends
Hand-made proofs error-prone

Formalized all API methods in a process calculus
  • Idealized cryptographic operations
  • Focus on protocol logic

Automated verification using ProVerif
  • Proofs for unbounded number of parallel sessions
  • Ensures absence of unintended protocol interleavings
Attacker model:
- Attacker controls network topology
  - Number of principals
  - Social relations
- Attacker dictates which protocols to run
  - Corrupted principals allowed

Trace-based verification
- Proven access control for all protocols
Attacker model:
- Two systems, two distinguished principals
- Attacker controls network topology
- Attacker dictates which protocols to run

Distinguished principals must register the same principals

Anonymity for all protocols except for friend requests
Experimental Evaluation

- Implemented all cryptographic primitives
- Performed on a standard notebook
  - 2.5 GHz Dual Core Processor
  - 4 GB main memory
- Signature scheme fast even for large numbers
- Run-time dominated by zero-knowledge proofs
  - Not surprising ...
  - Very practical (≈ 1 second)
∃x. g^x = 4711

Time in ms

Pseudonym size in bits

Size in kB

Pseudonym size in bits
∃c, x. certifies(c, 4711, Alice) ∧ g^x = 4711
∃c. certifies(c,"friend",Alice)

Proof generation in ms

Proof verification in ms

- Pseudonymous Authentication
- Relation Authentication

Key size and Pseudonym size in bits

Key size and Pseudonym size in bits
∃c,p. certifies(c, 4711, Alice) \land \text{owns}(p, 4711)

∃c. certifies(c,"friend",Alice)
Prototype integrated into Facebook

- Realized as Facebook app
  - Facebook most popular social network
  - Facebook has well-documented API
  - No interference with regular Facebook functionality

- Anonymous group-based access to pictures and wall posts
Conclusion

- Presented a cryptographic API that
  - Enforces fine-grained access control
  - Provides anonymity
  - Keeps the social relations private
  - Is usable in centralized and decentralized settings

- Secure even if system is compromised
  - Signatures can be stored in a secure location
  - ACLs do not identify friends and reveal no network structure
  - Zero-knowledge proofs protect requesters

- Formally verified protocols

- Efficient implementation