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

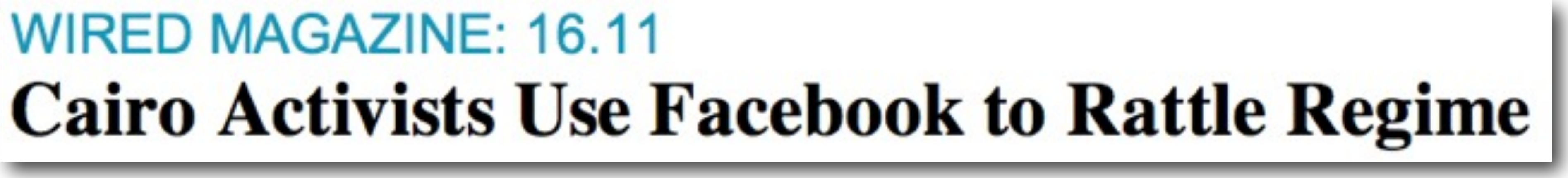
Reports claim that Facebook silently gave profile access to Italian police

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

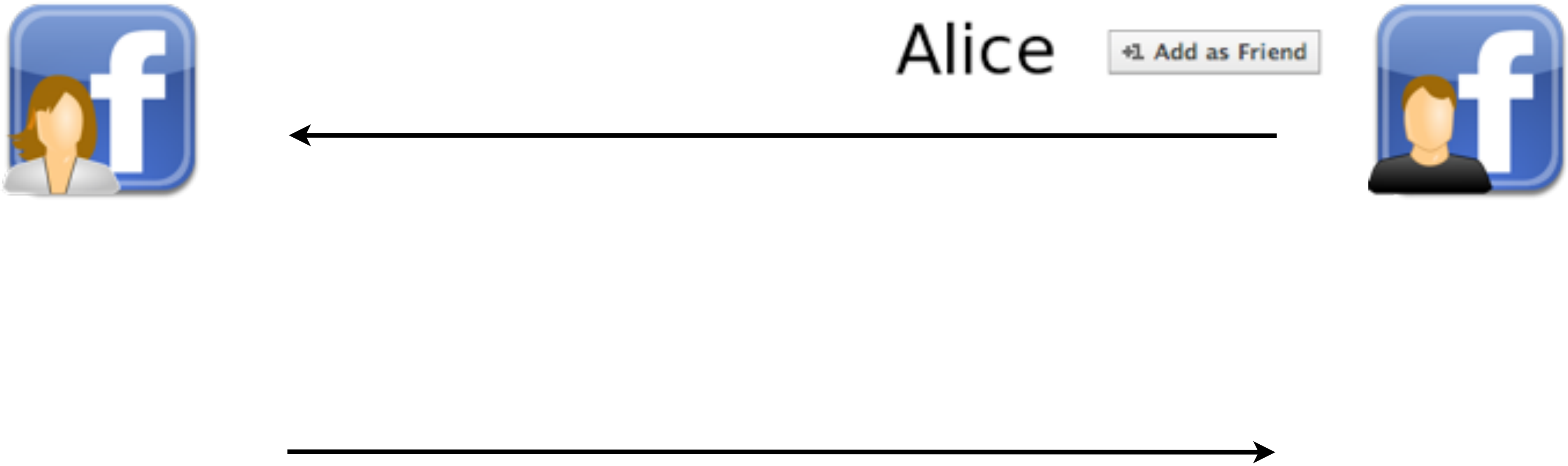
Cairo Activists Use Facebook to Rattle Regime

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

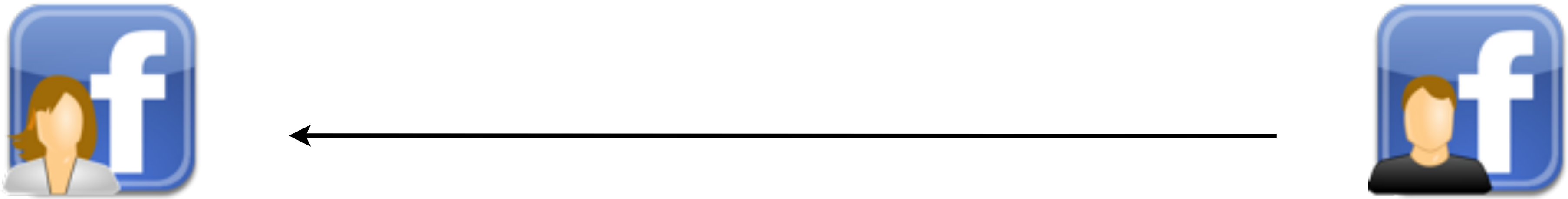
Our Contribution

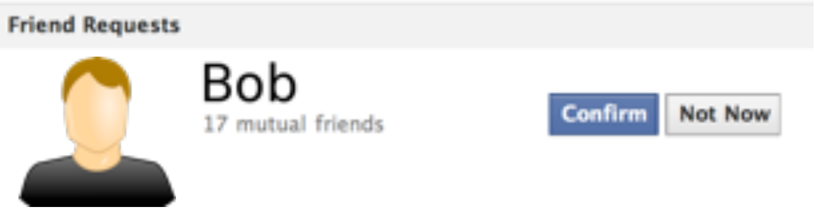
- Cryptographic API providing
  - Fine-grained access control
  - Anonymity
  - Privacy of social relations
  - Flavor of coercion resistance
- API also applicable in centralized settings
- Formal verification of all API methods
- Experimental Evaluation

Facebook



Facebook





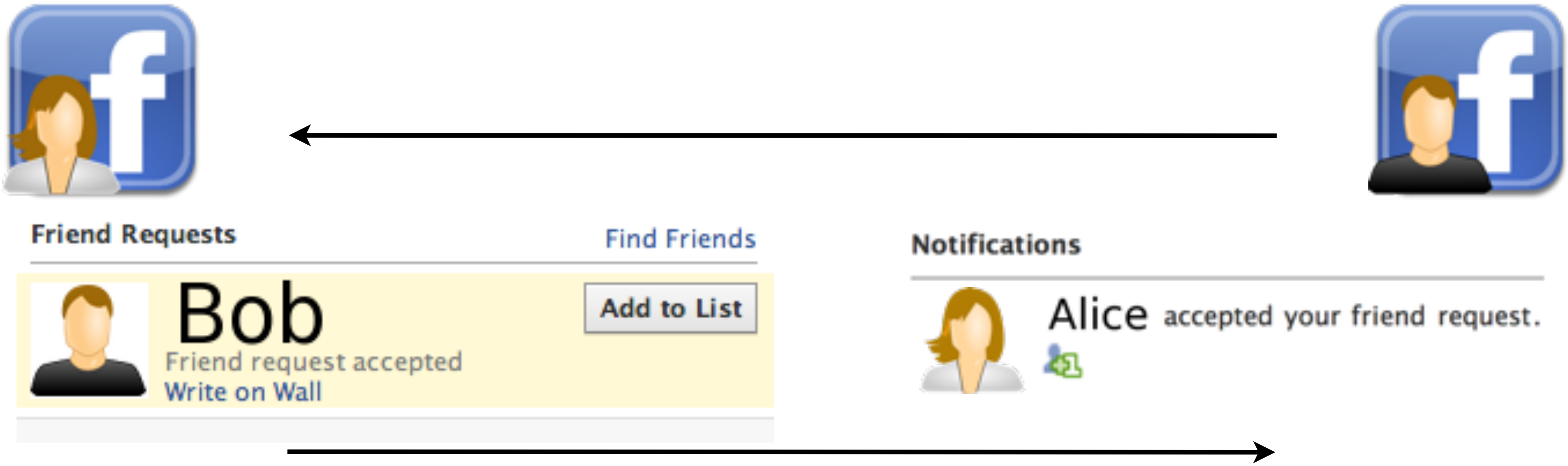
Friend Requests

Bob

17 mutual friends

Confirm Not Now

Facebook



Friend Requests

Find Friends

Bob

Add to List

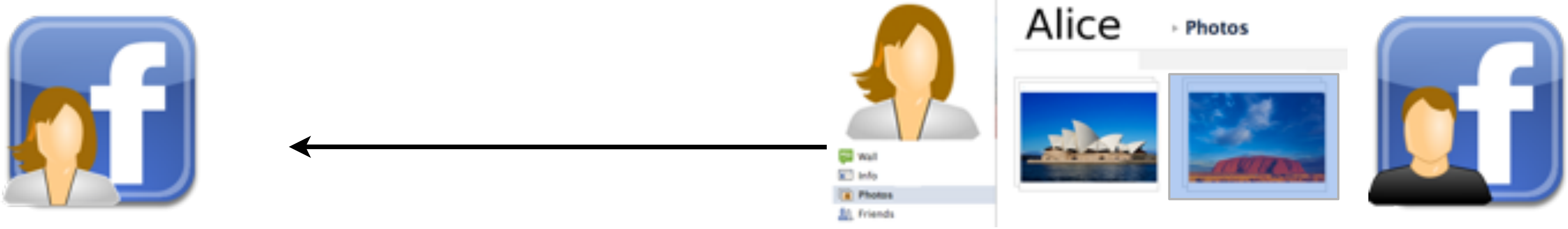
Friend request accepted

Write on Wall

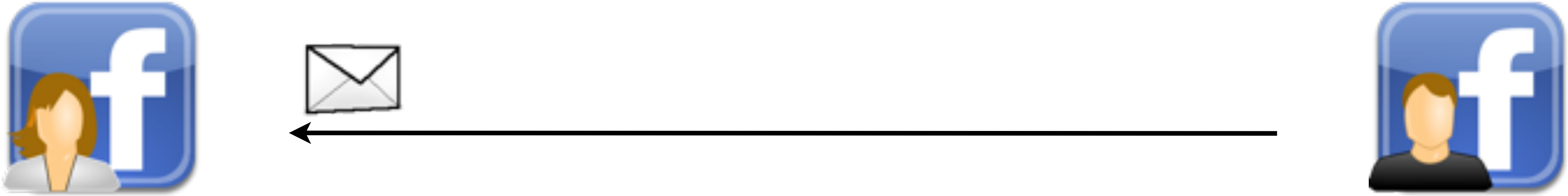
Notifications

Alice accepted your friend request.

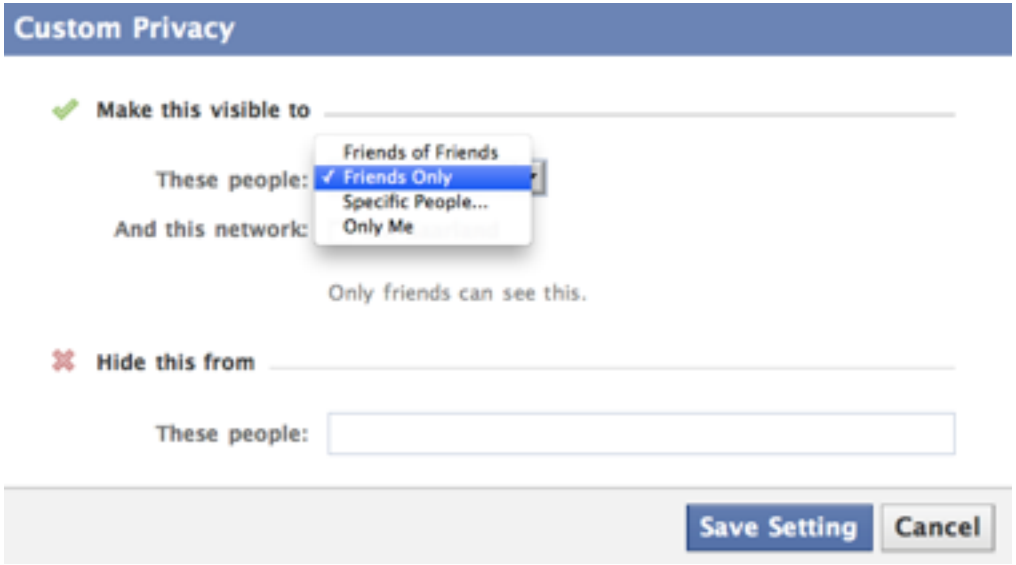
Facebook



Facebook



Bob
Friends



Custom Privacy

✓ **Make this visible to**

These people: Friends Only

And this network:

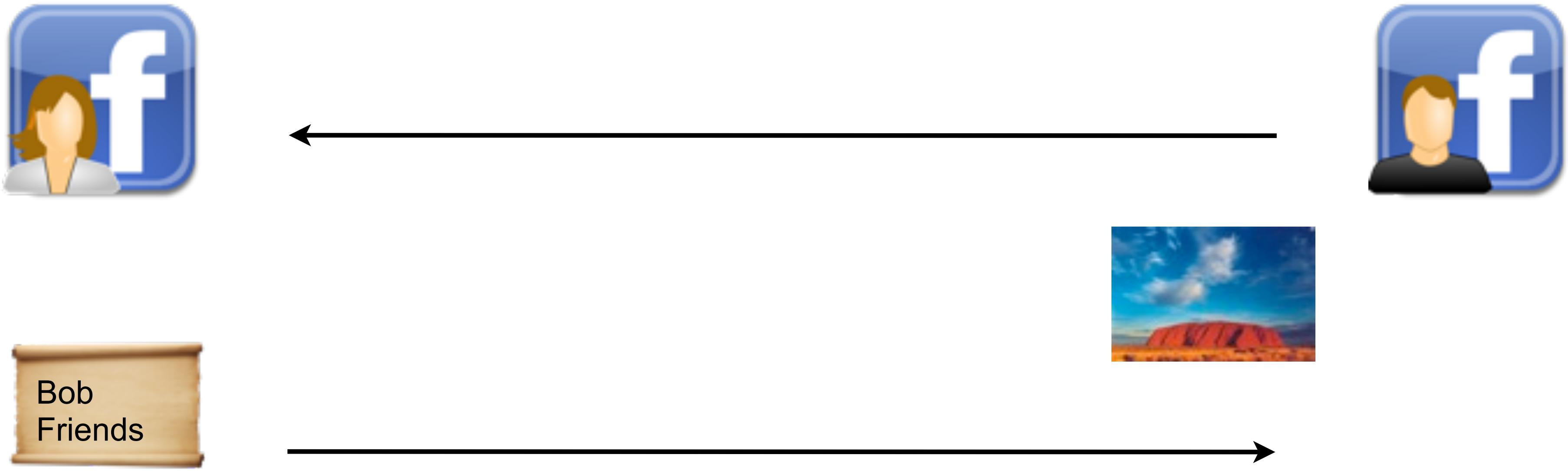
Only friends can see this.

✗ **Hide this from**

These people:

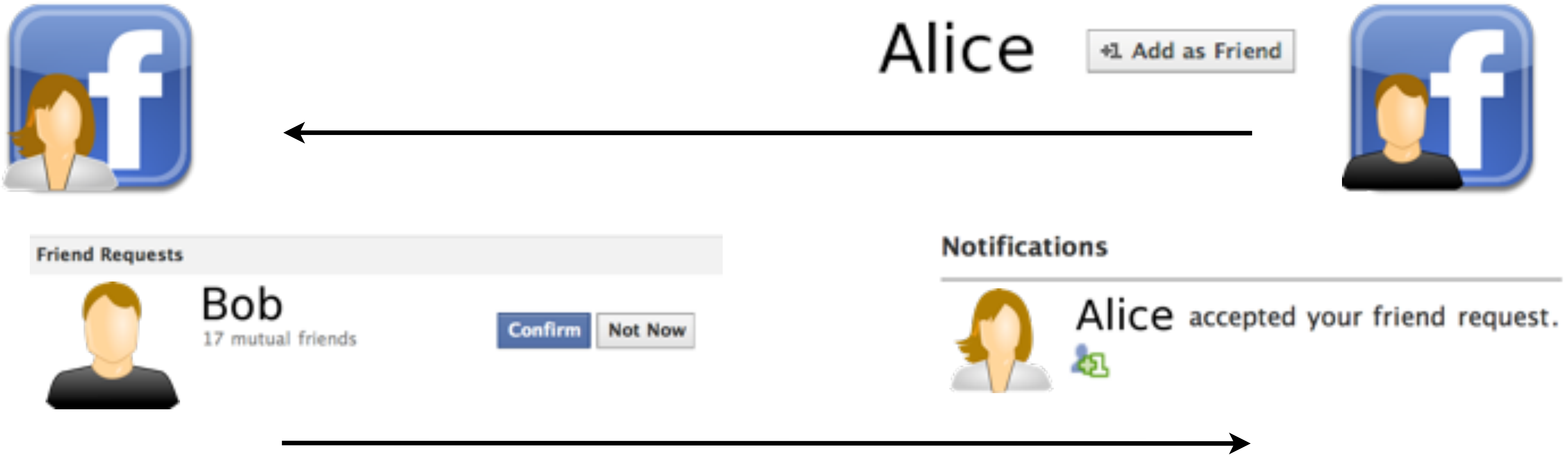
- Request is checked against ACL

Facebook



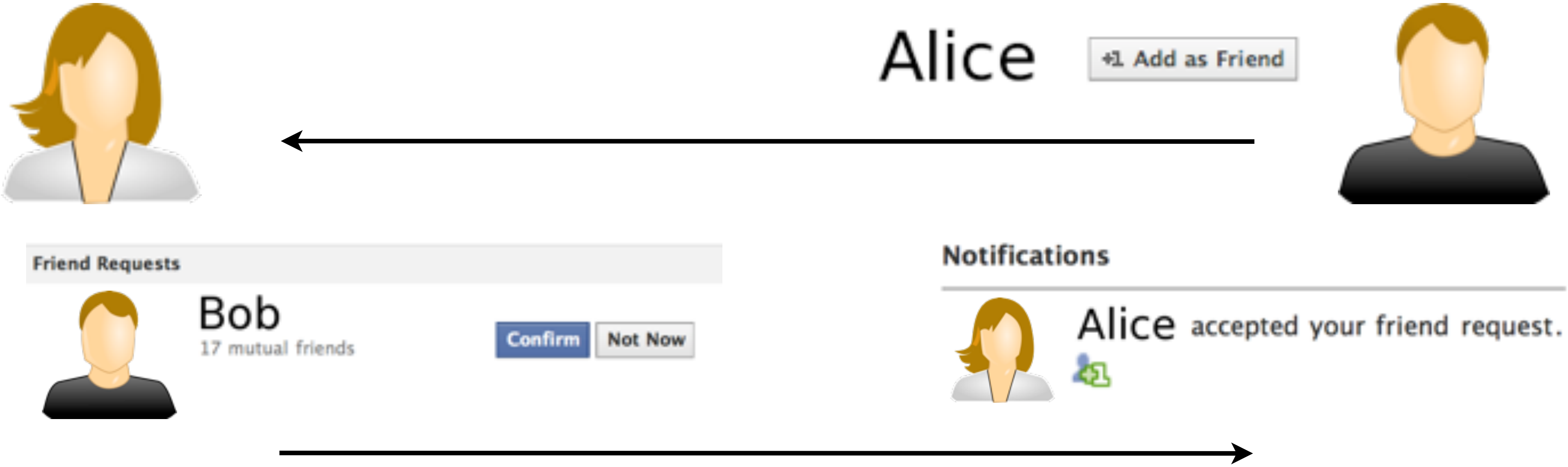
- Resource released if check against ACL succeeds

Our Approach: Decentralized Setting



Alice accepted your friend request.

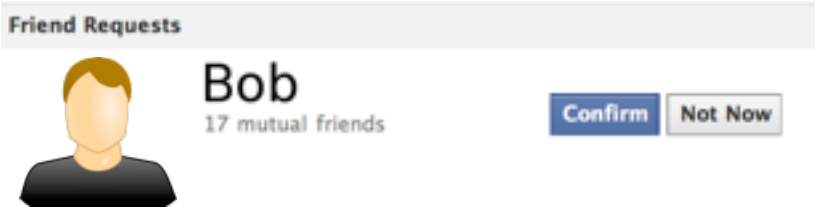
Our Approach: Decentralized Setting



Alice accepted your friend request.

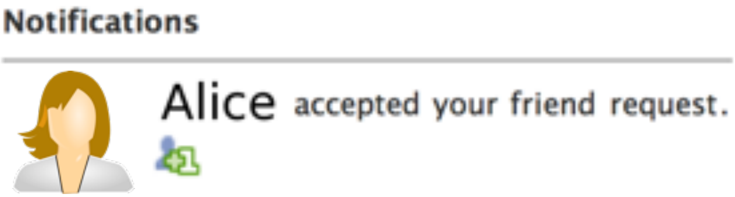
Our Approach: Decentralized Setting

I am Bob.
Please befriend me.



Friend Requests

Bob
17 mutual friends

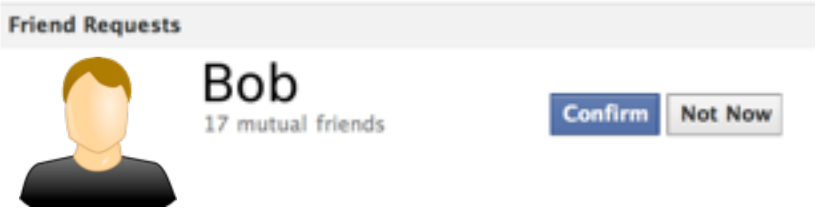


Notifications

Alice accepted your friend request.

Our Approach: Decentralized Setting

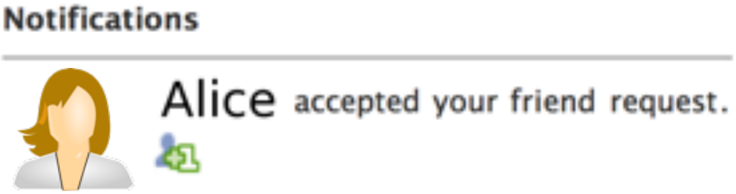
I am Bob.
Please befriend me.



Friend Requests

Bob

17 mutual friends

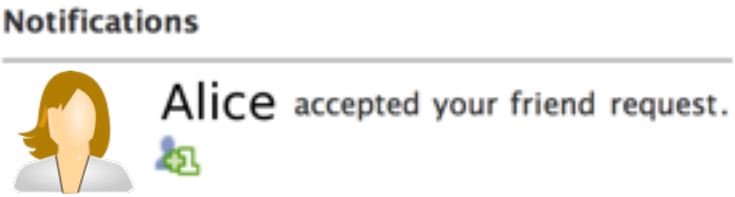


Notifications

Alice accepted your friend request.

Our Approach: Decentralized Setting

I am Bob.
Please befriend me.

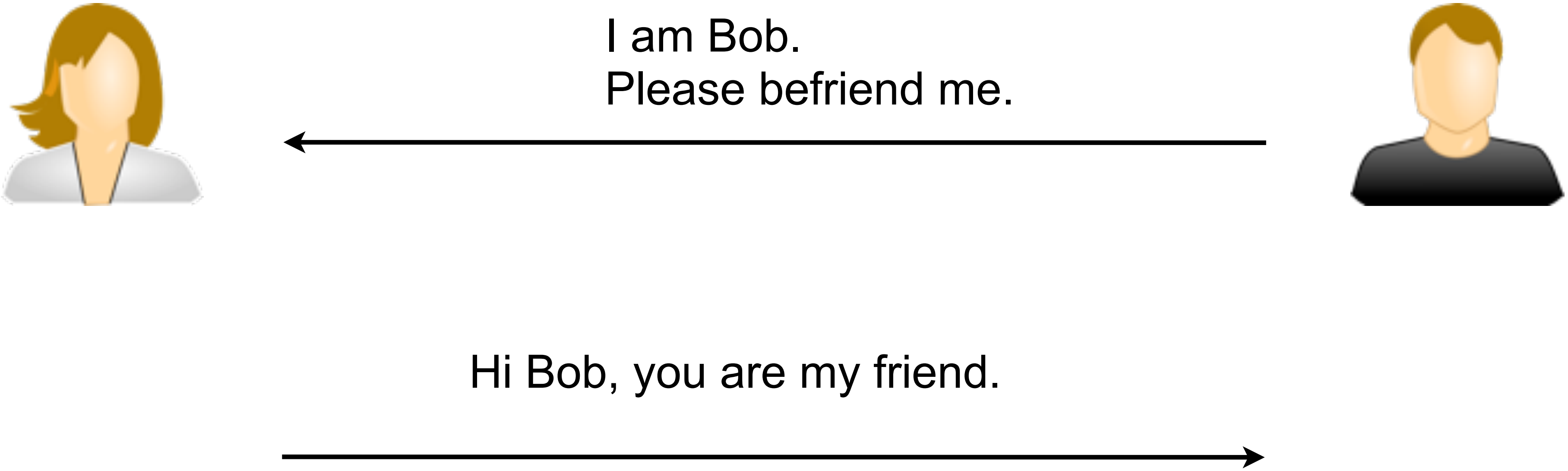


Notifications

Alice accepted your friend request.

Hi Bob, you are my friend.

Our Approach: Decentralized Setting

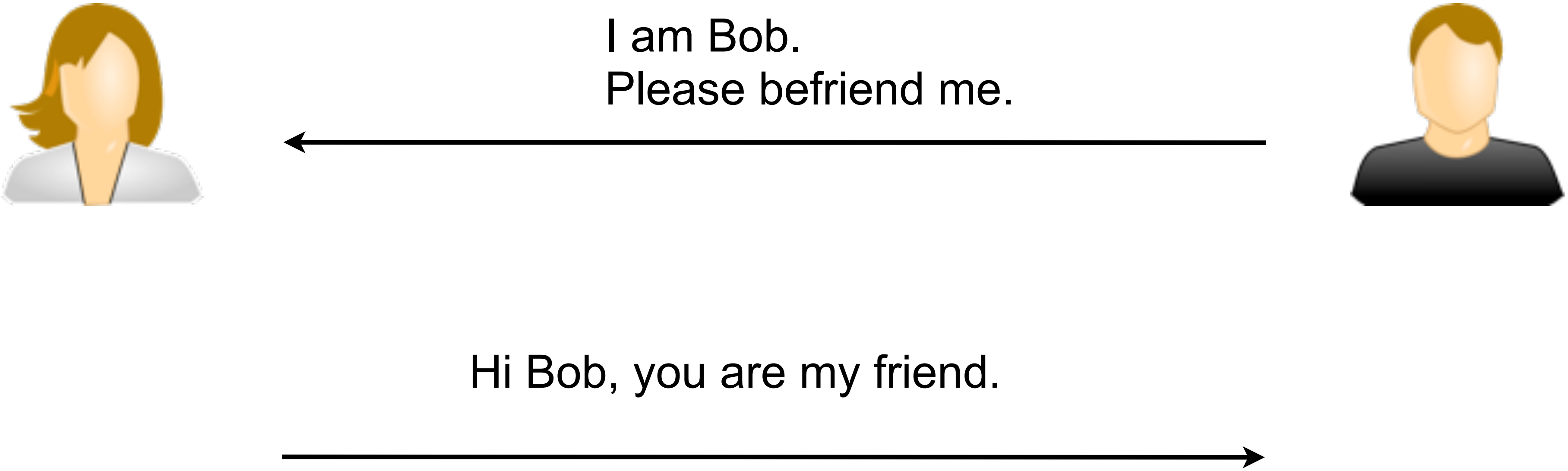


I am Bob.
Please befriend me.

Hi Bob, you are my friend.

- We deploy certificates to establish authenticity in decentralized setting

Certificates

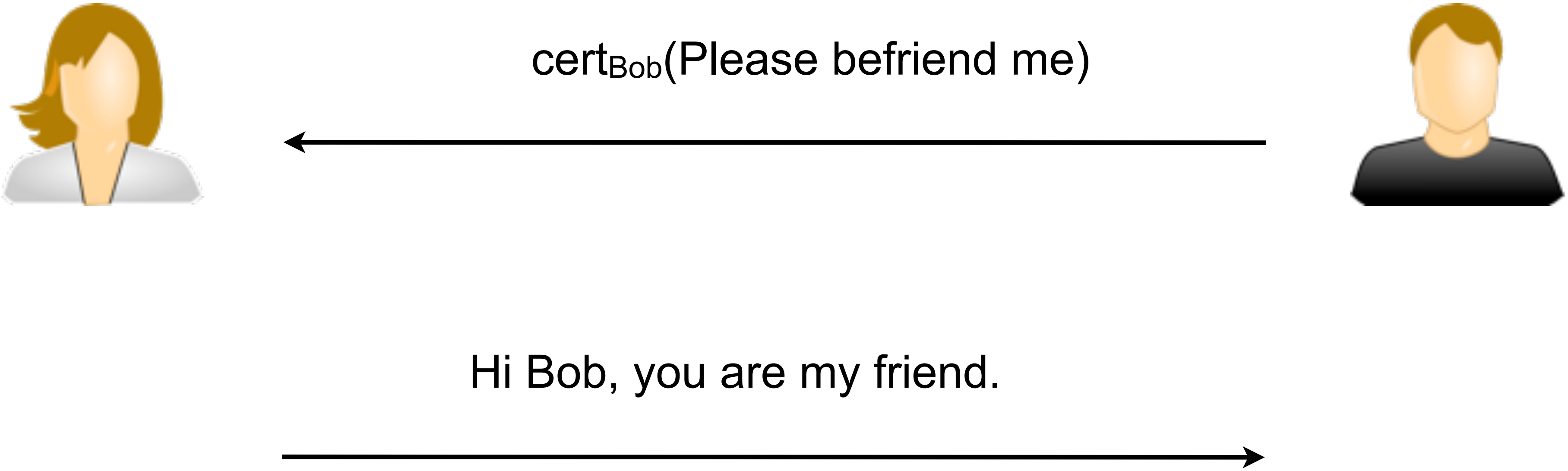


I am Bob.
Please befriend me.

Hi Bob, you are my friend.

- 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

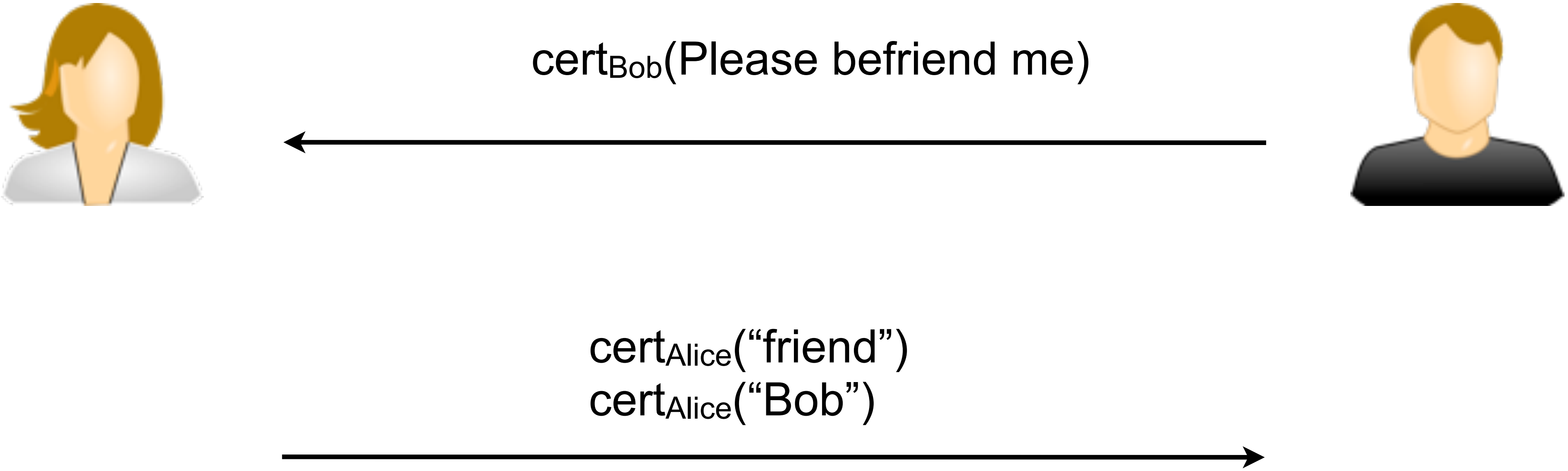
Certificates


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

Hi Bob, you are my friend.

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

Certificates

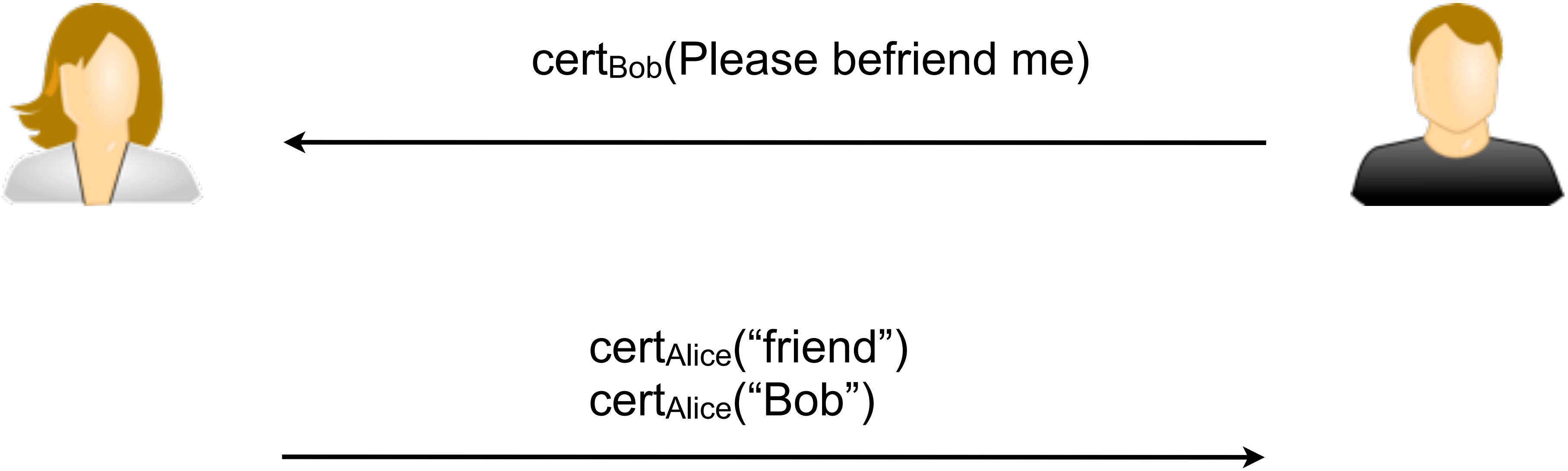


cert_{Bob}(Please befriend me)

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

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

Pseudonyms

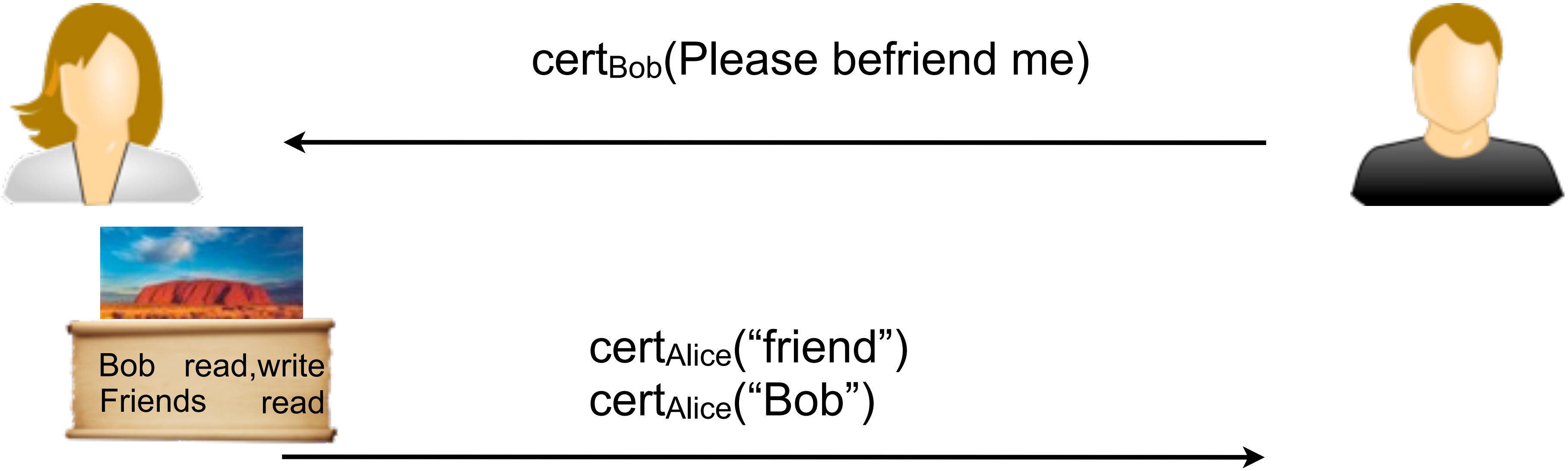


cert_{Bob}(Please befriend me)

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

- Plain names inhibit anonymity

Pseudonyms



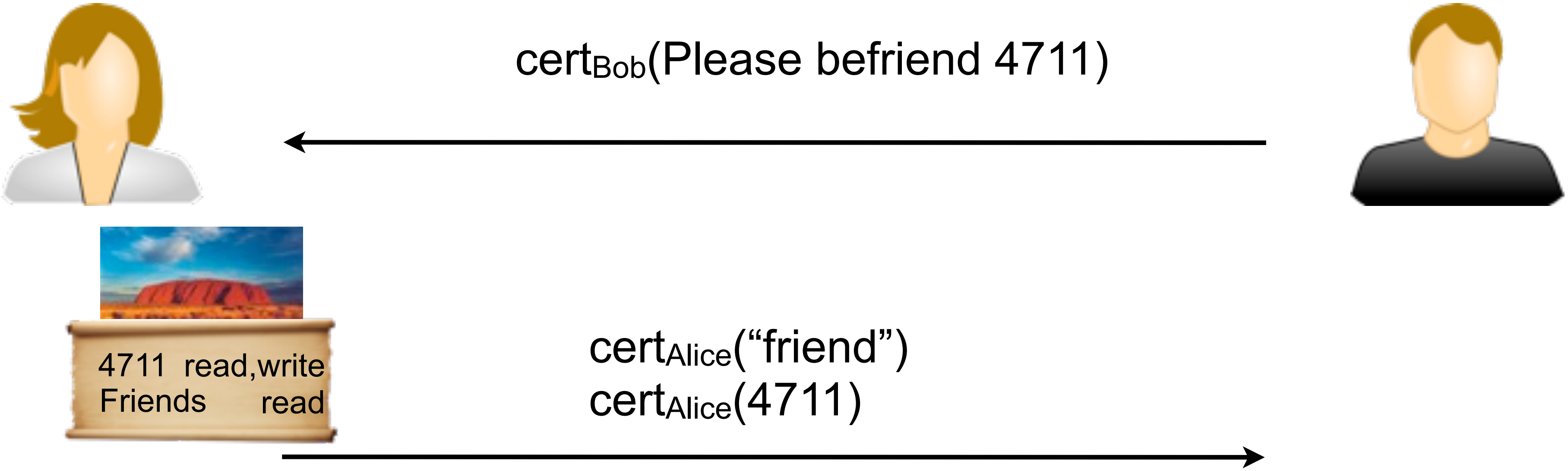
cert_{Bob}(Please befriend me)

Bob	read,write
Friends	read

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

- Plain names inhibit anonymity
    - ACLs reveal social graph

Pseudonyms



cert$_{Bob}$(Please befriend 4711)

4711	read,write
Friends	read

cert$_{Alice}$(“friend”)
cert$_{Alice}$(4711)

- 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

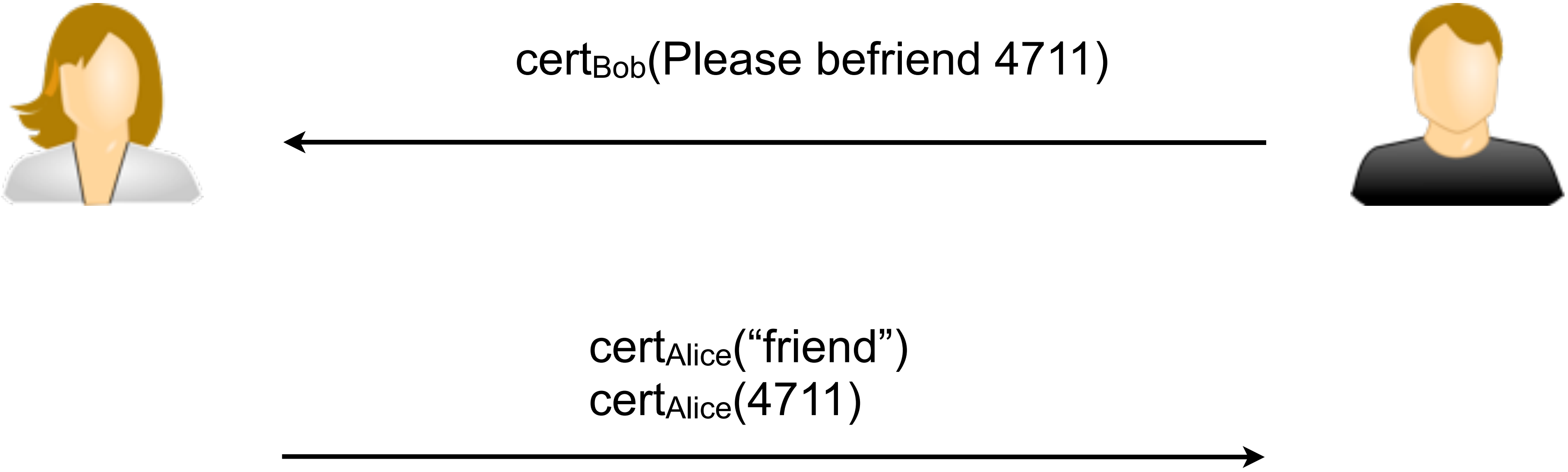
- Desired properties (similar to real names):
  - One pseudonym belongs to one user
    - Impersonation / identity theft impossible
  - Pseudonyms should be trackable
    - If desired
  - One user may own several pseudonyms
    - Increases anonymity
    - Prevents complete tracking
  - Do not reveal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

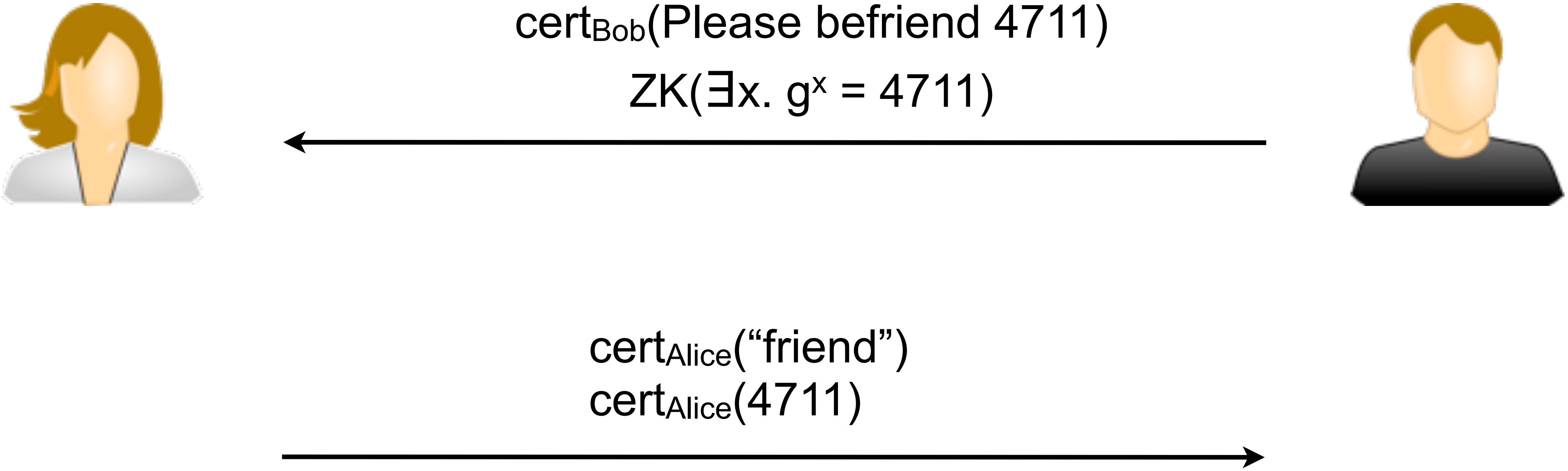


cert_{Bob}(Please befriend 4711)

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

- Prevent impersonation using a proof of pseudonym ownership

Zero-Knowledge Proofs



cert_{Bob}(Please befriend 4711)

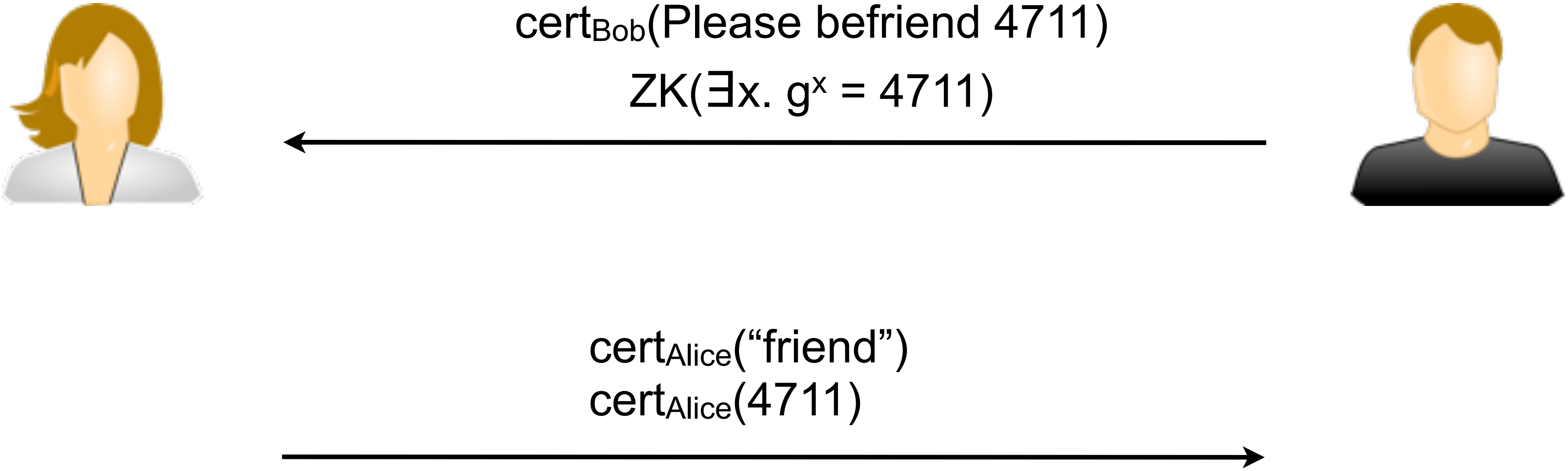
ZK($\exists x.\ g^x = 4711$)

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

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

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

Zero-Knowledge Proofs



cert_{Bob}(Please befriend 4711)

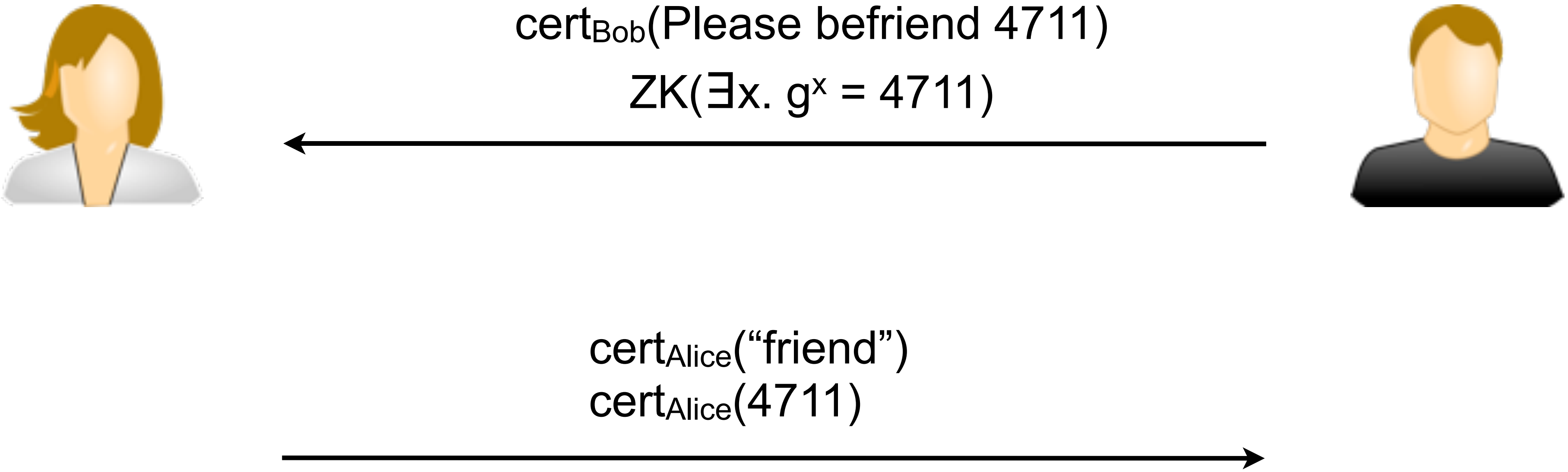
ZK($\exists x.\ g^x = 4711$)

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

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

- 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}(Please befriend 4711)

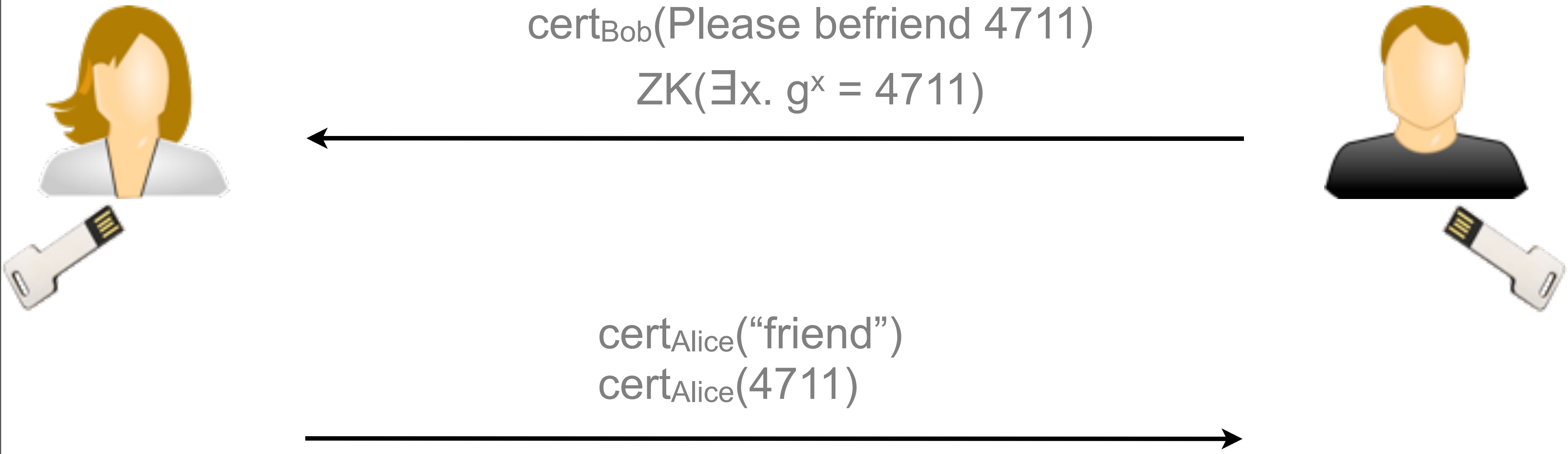
ZK($\exists x.\ g^x = 4711$)

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

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

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

Secure Storage Devices



cert_{Bob}(Please befriend 4711)

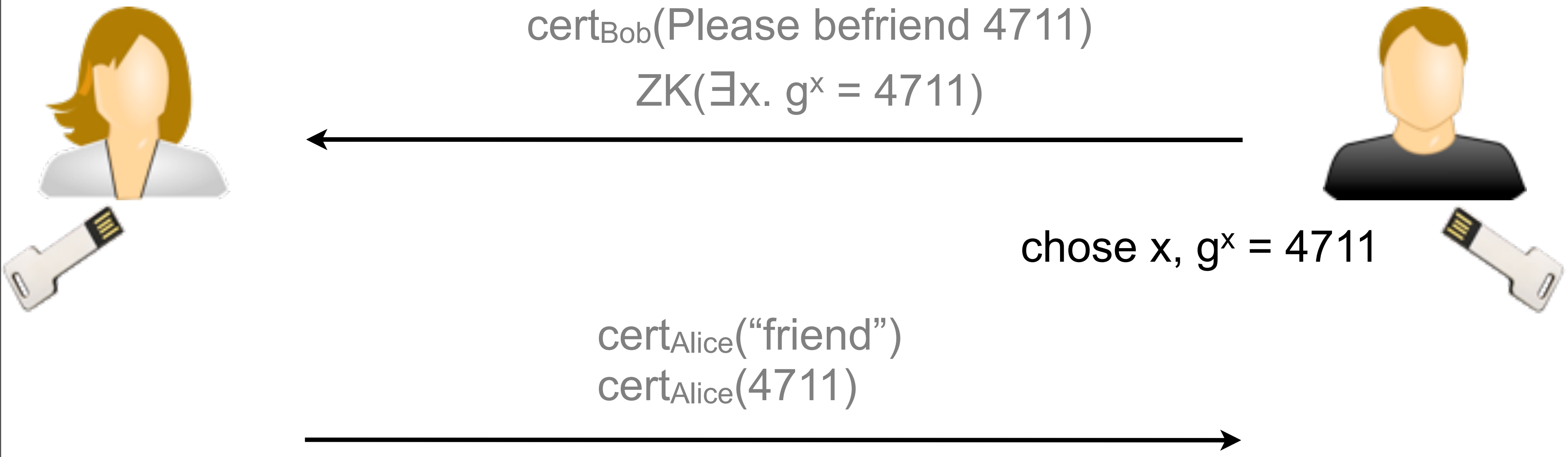
ZK($\exists x.\ g^x = 4711$)

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

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

Secret values exclusively stored on secure storage device

Secure Storage Devices



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

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

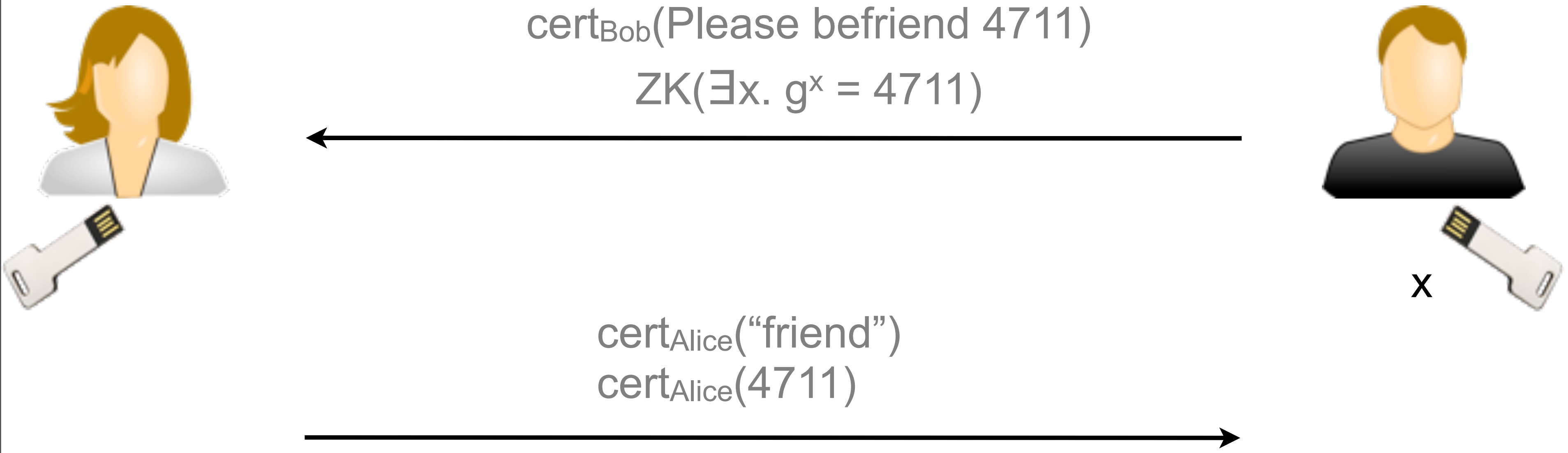
chose x, $g^x = 4711$

cert$_{\text{Alice}}$(“friend”)

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

Secret values exclusively stored on secure storage device

Secure Storage Devices



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

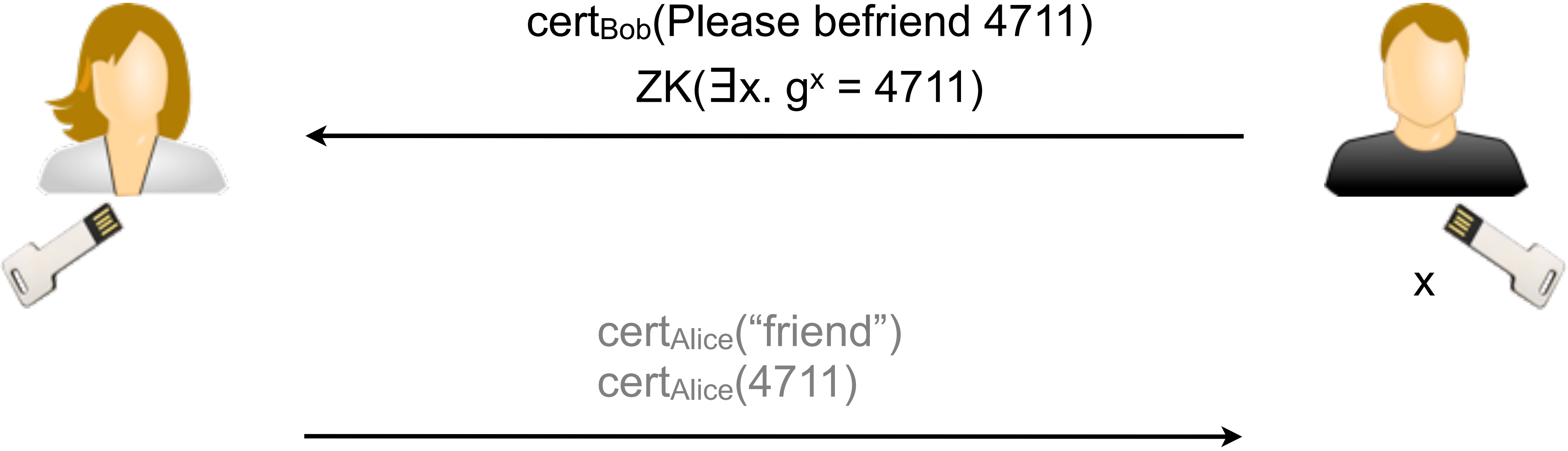
$\text{ZK}(\exists x.\ g^x = 4711)$

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

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

Secret values exclusively stored on secure storage device

Secure Storage Devices



cert_{Bob}(Please befriend 4711)

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

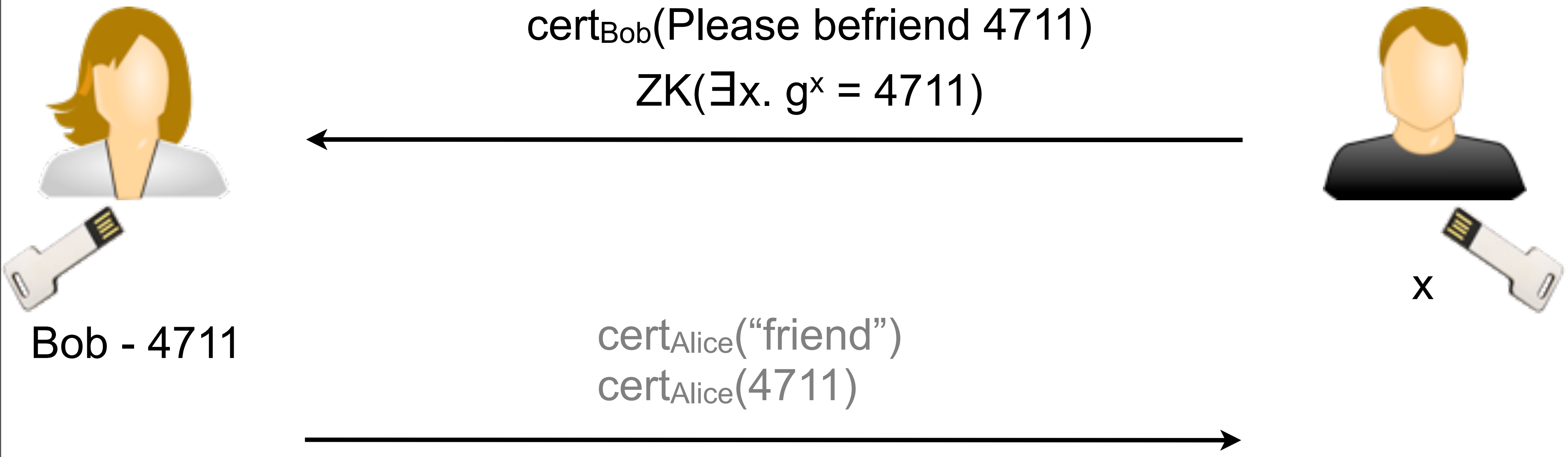
x

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

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

Secret values exclusively stored on secure storage device

Secure Storage Devices



cert_{Bob}(Please befriend 4711)

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

Bob - 4711

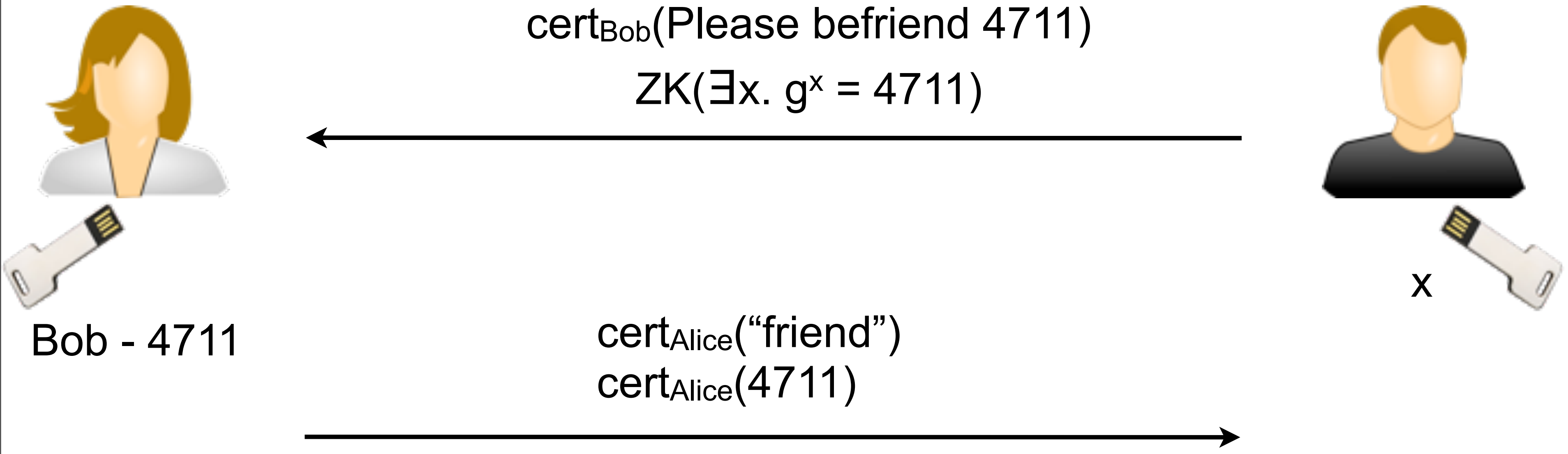
x

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

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

Secret values exclusively stored on secure storage device

Secure Storage Devices



cert_{Bob}(Please befriend 4711)

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

Bob - 4711

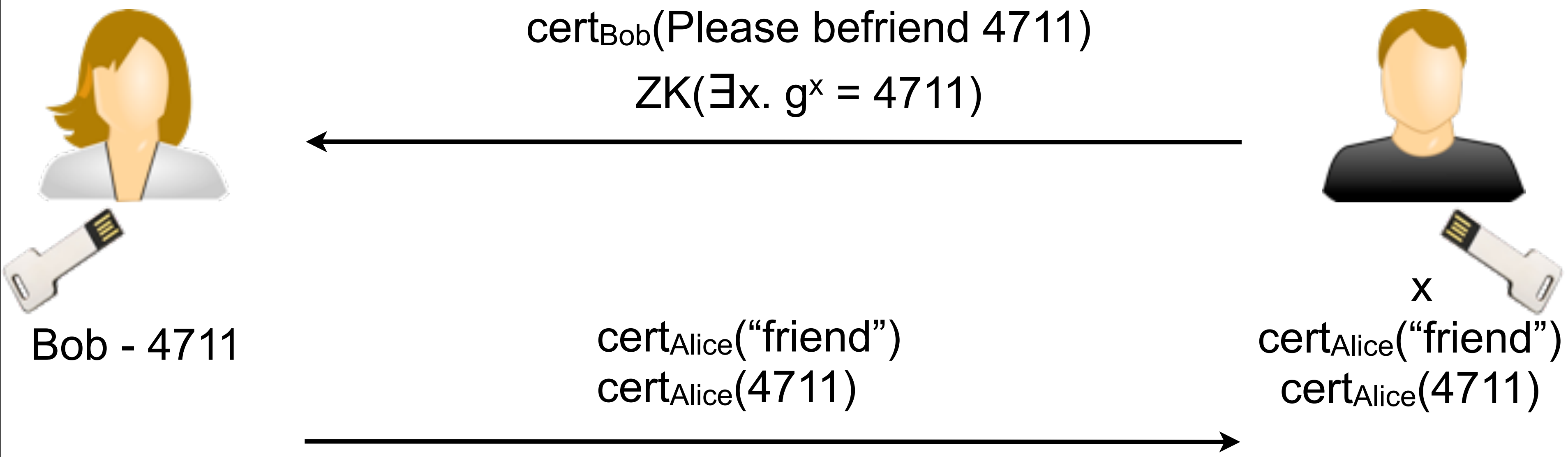
x

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

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

Secret values exclusively stored on secure storage device

Secure Storage Devices



cert$_{\mathrm{Bob}}$(Please befriend 4711)

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

Bob - 4711

x

cert$_{\mathrm{Alice}}$(“friend”)

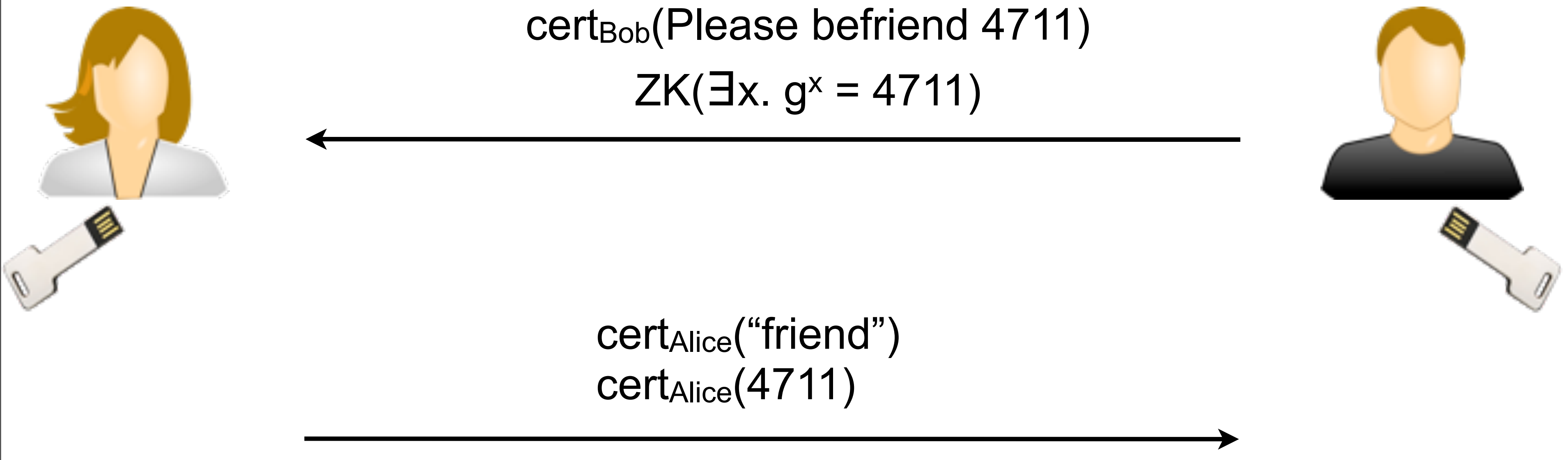
cert$_{\mathrm{Alice}}$(4711)

cert$_{\mathrm{Alice}}$(“friend”)

cert$_{\mathrm{Alice}}$(4711)

Secret values exclusively stored on secure storage device

Secure Storage Devices



cert_{Bob}(Please befriend 4711)

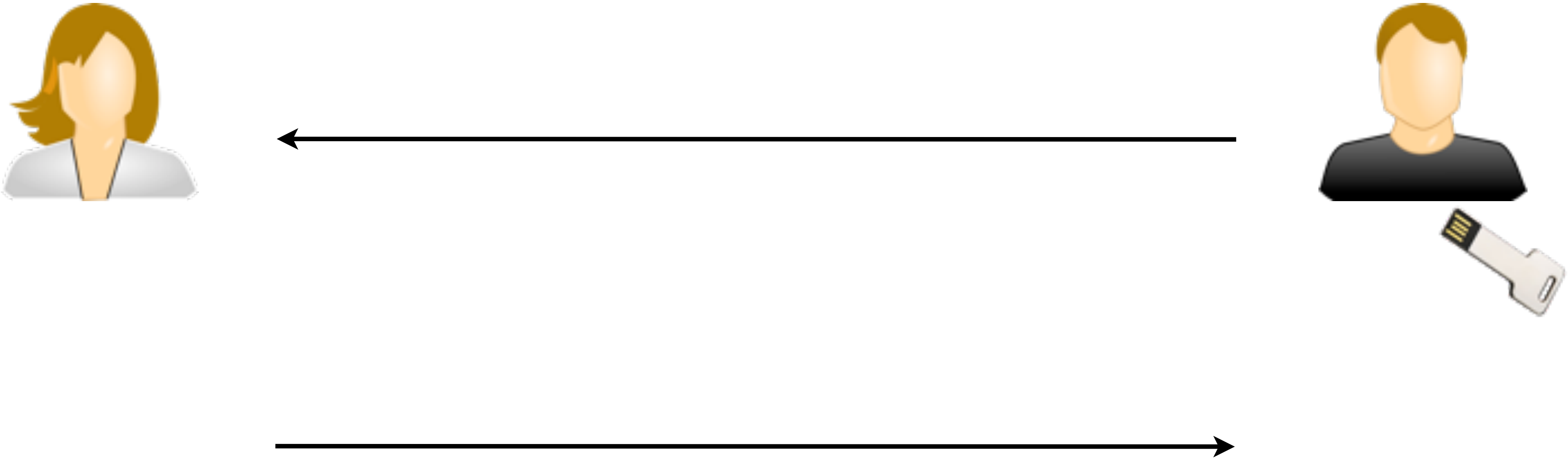
ZK($\exists x.\ g^x = 4711$)

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

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

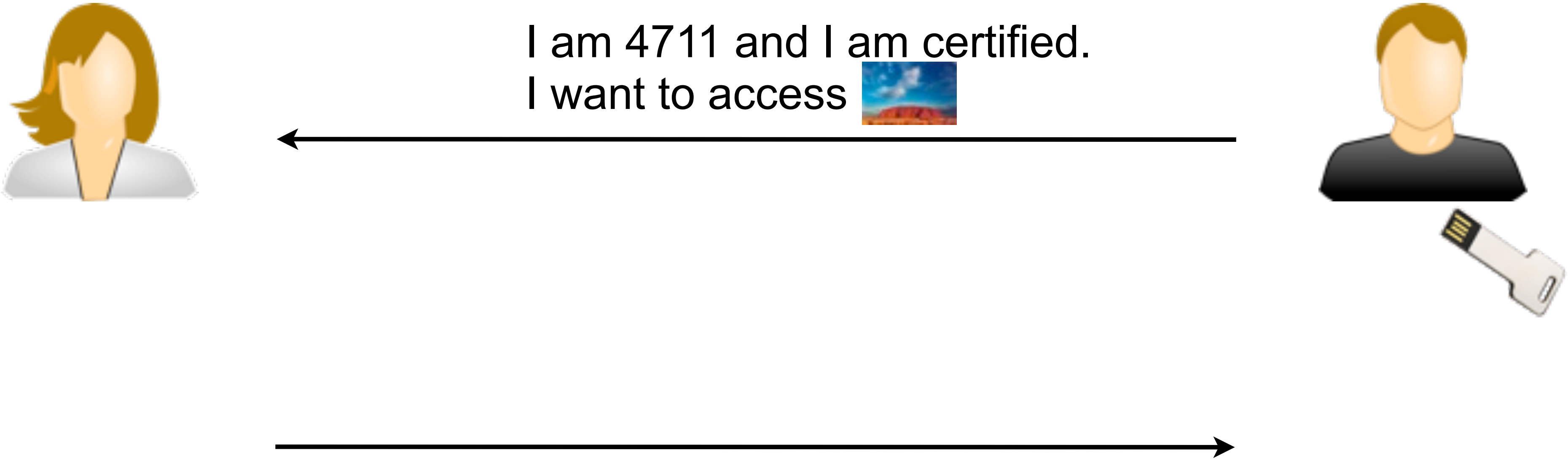
Secret values exclusively stored on secure storage device

Retrieving Resources



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

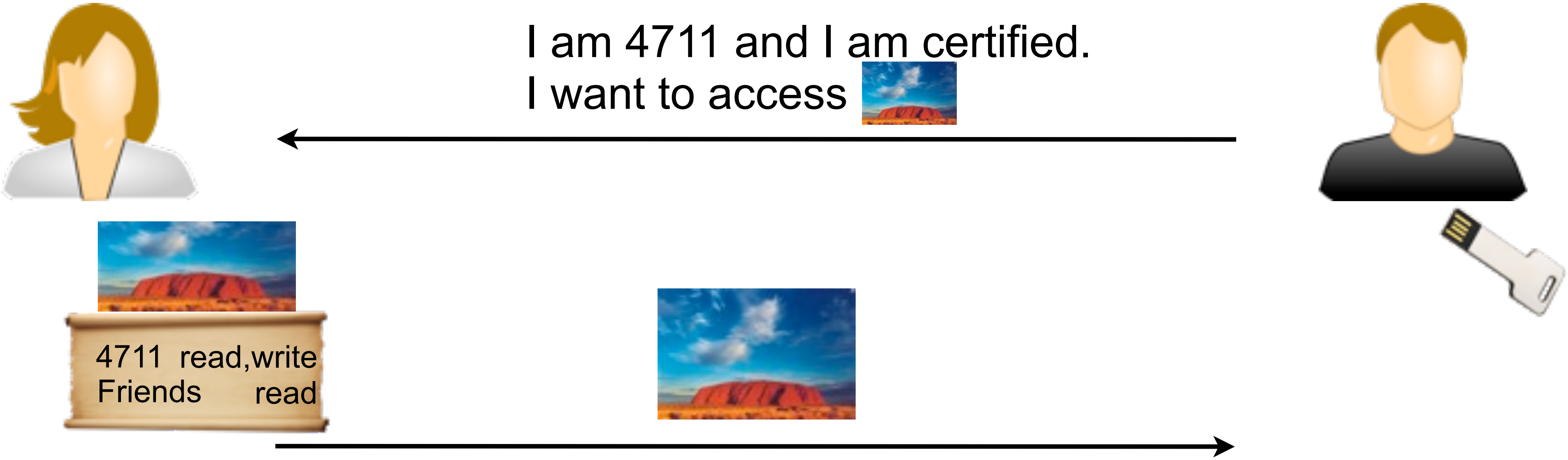
Retrieving Resources



I am 4711 and I am certified.
I want to access

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

Retrieving Resources

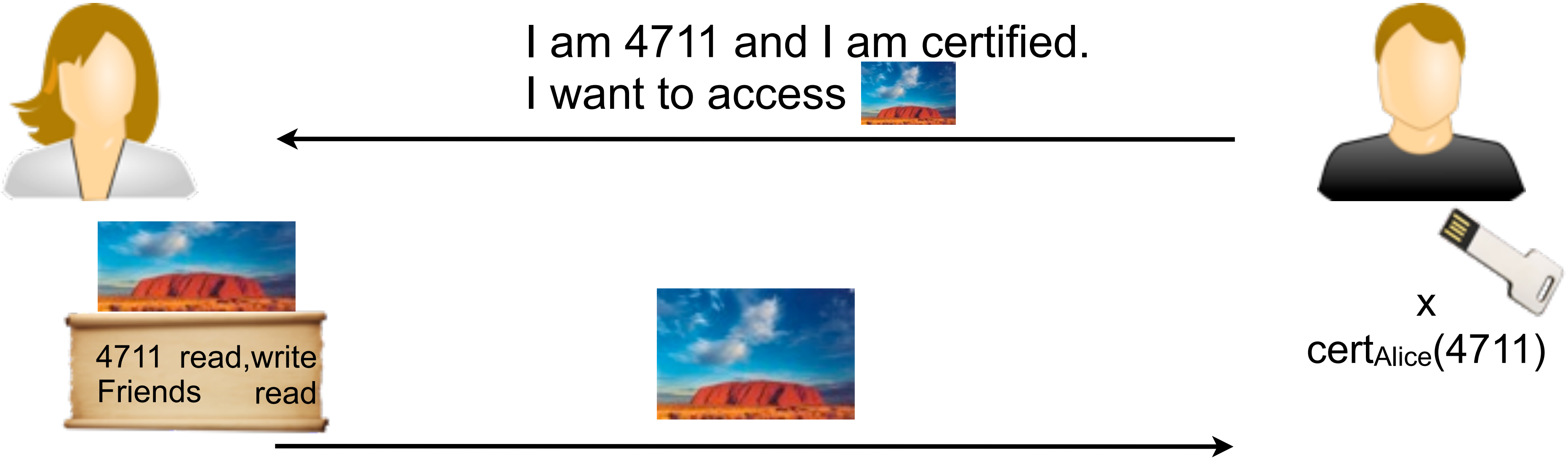


I am 4711 and I am certified.
I want to access

4711	read,write
Friends	read

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

Retrieving Resources



I am 4711 and I am certified.
I want to access

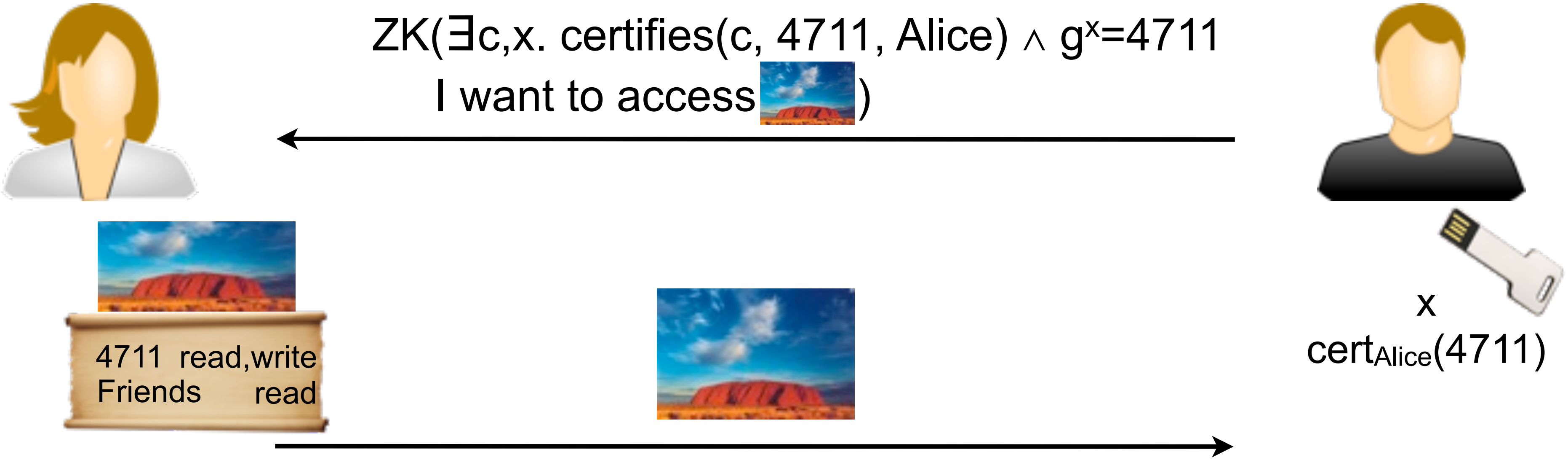
4711	read,write
Friends	read

x

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

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

Retrieving Resources



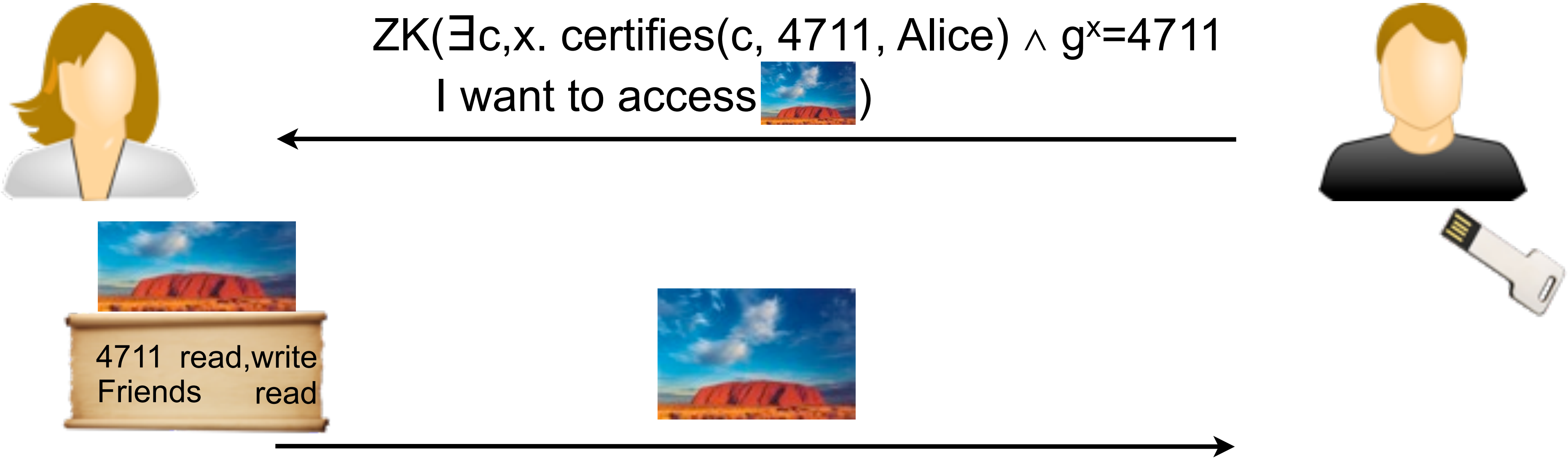
ZK(∃c,x. certifies(c, 4711, Alice) $\wedge$ g^x=4711
I want to access)

4711	read,write
Friends	read

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

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

Retrieving Resources



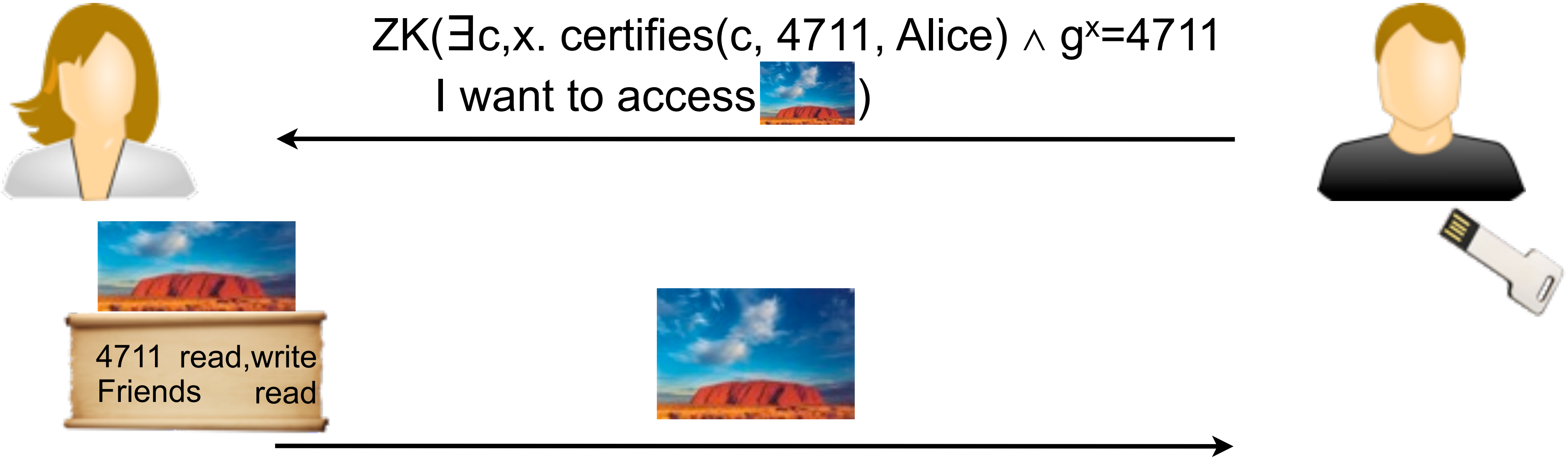
ZK(∃c,x. certifies(c, 4711, Alice) $\wedge$ g^x=4711

I want to access)

4711	read,write
Friends	read

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

Retrieving Resources



$\mathrm{ZK}(\exists c,x.\ \mathrm{certifies}(c,\ 4711,\ \mathrm{Alice}) \wedge g^{x}=4711$

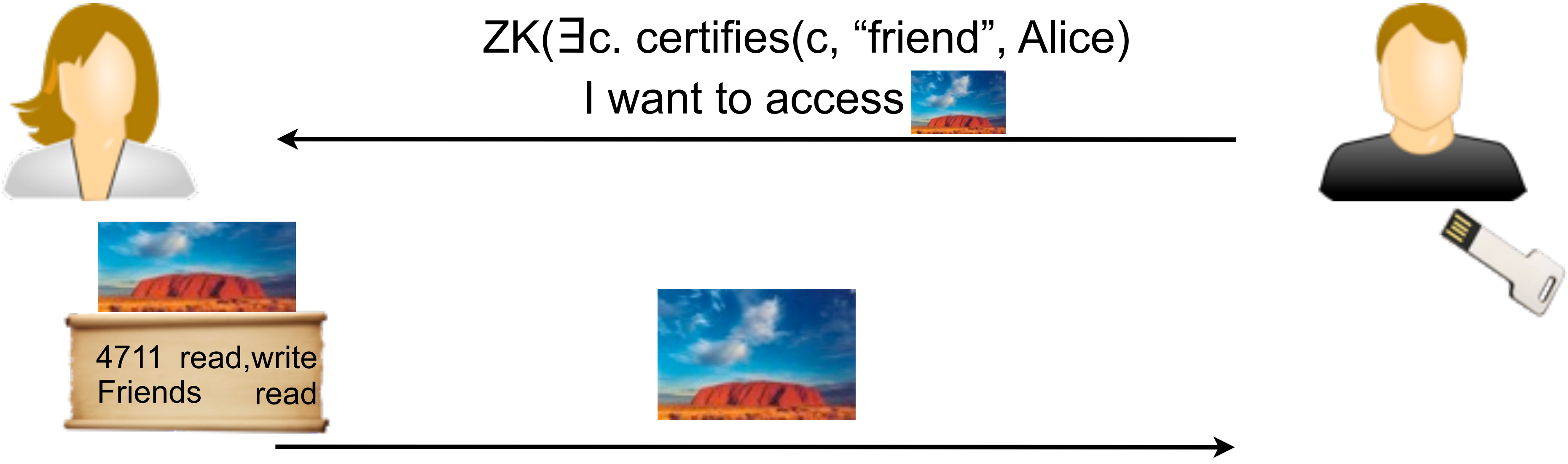
I want to access)

4711	read,write
Friends	read

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

- Zero-knowledge proof reveals
  - Pseudonym
  - Requested picture

Retrieving Resources



ZK(∃c. certifies(c, “friend”, Alice)

I want to access

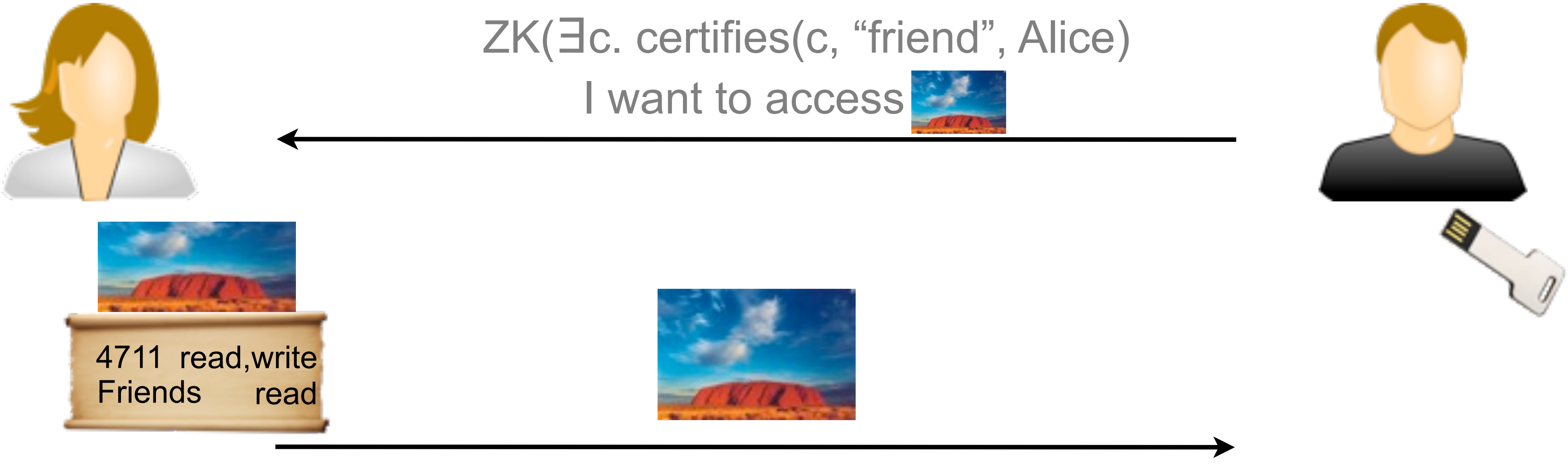
4711 read,write

Friends read

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

- Zero-knowledge proof reveals
  - Pseudonym
  - Requested picture

Retrieving Resources



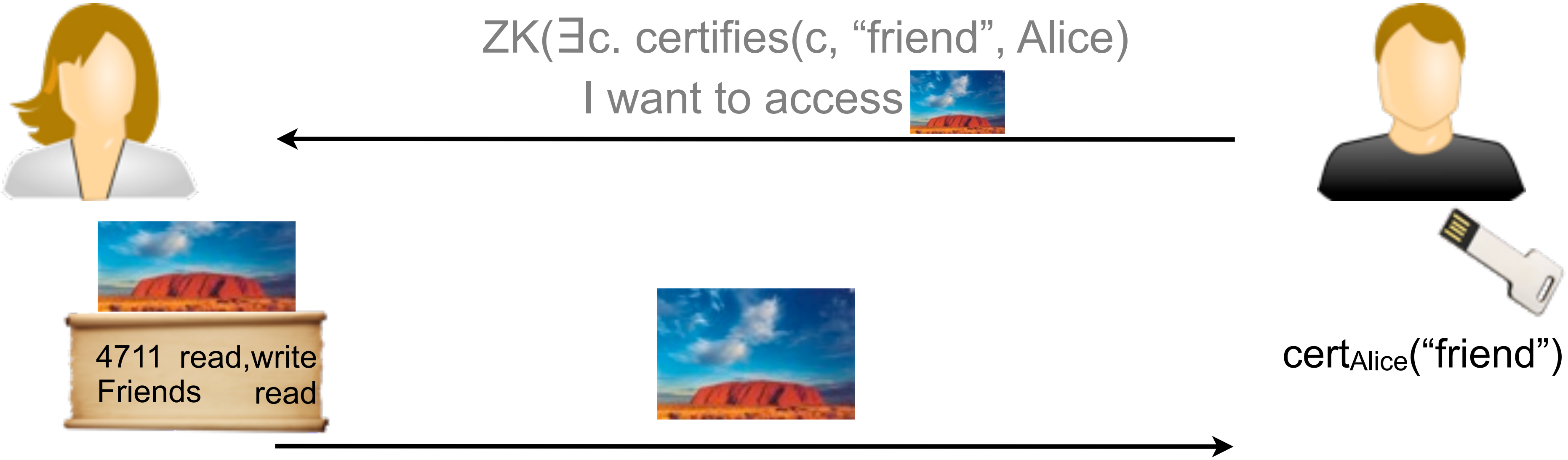
ZK(∃c. certifies(c, “friend”, Alice)

I want to access

4711	read,write
Friends	read

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

Retrieving Resources



ZK(∃c. certifies(c, “friend”, Alice)

I want to access

4711 read,write

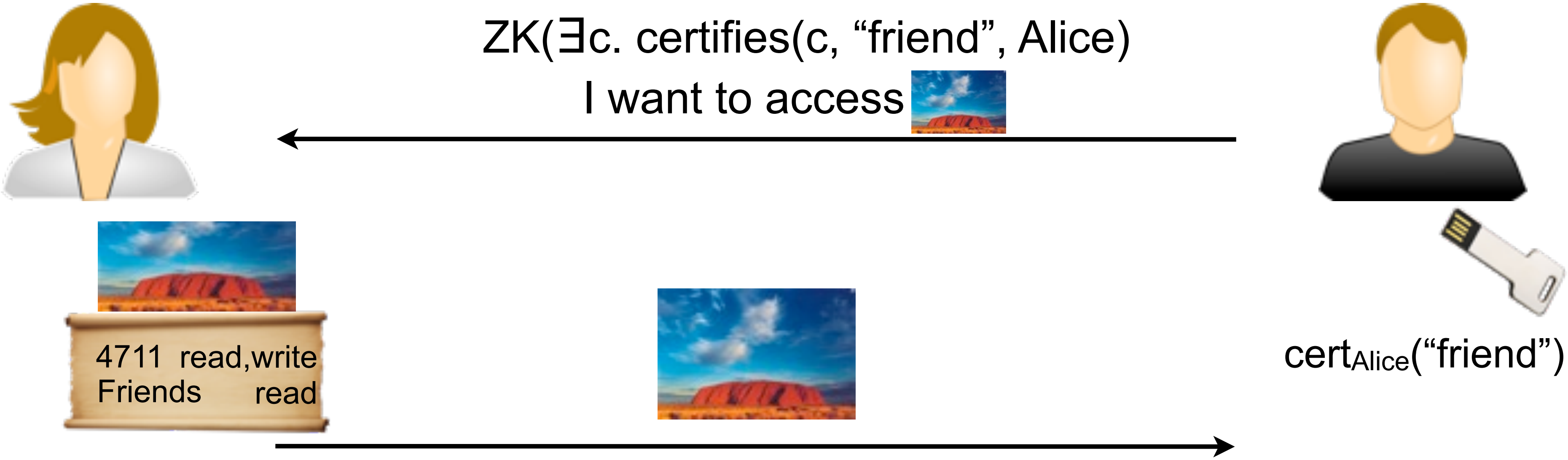
Friends read

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

Zero-knowledge proof hides the identity of the prover

and only reveals the social relation between verifier and prover

Retrieving Resources

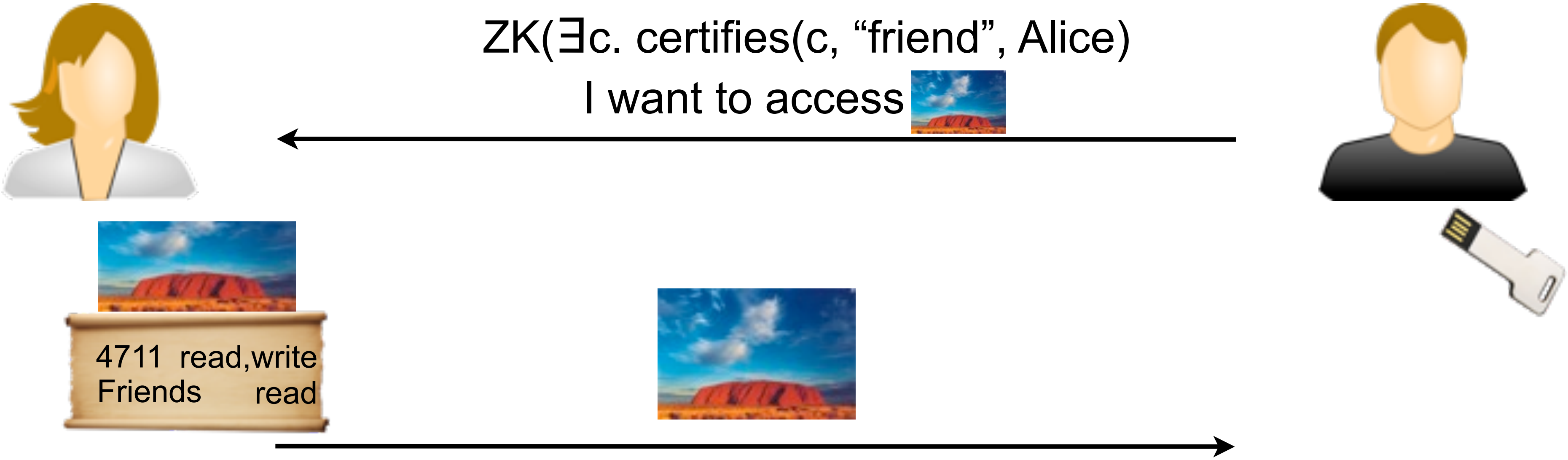


ZK(∃c. certifies(c, “friend”, Alice)

I want to access

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

Retrieving Resources



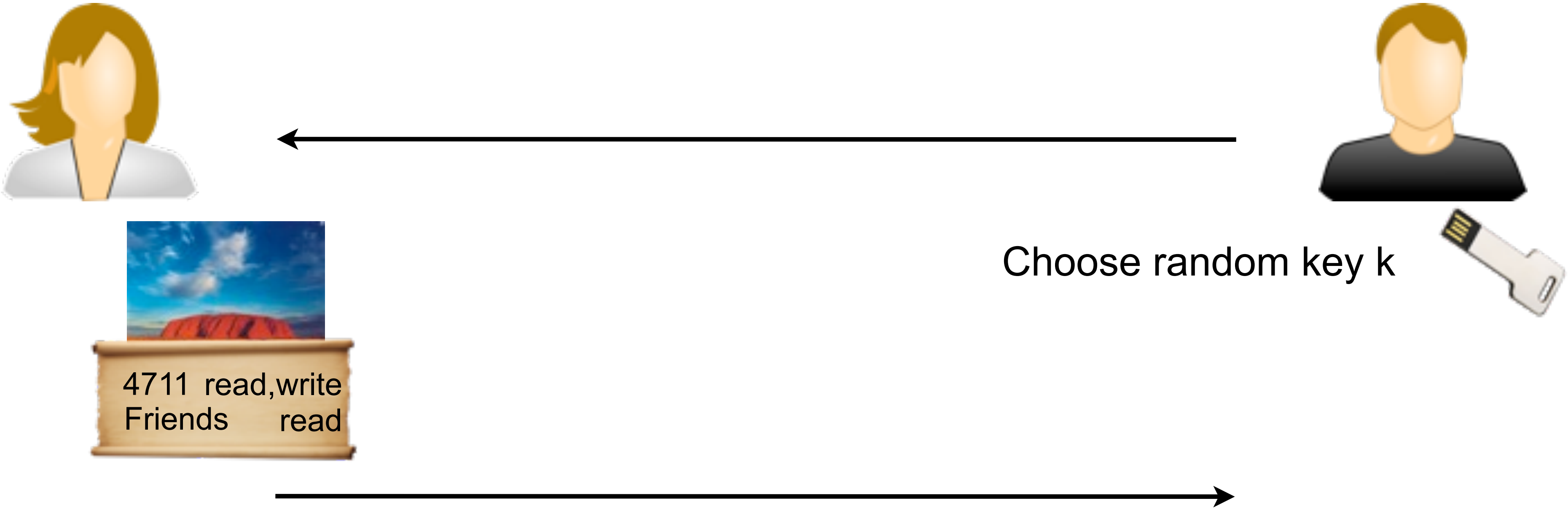
ZK(∃c. certifies(c, “friend”, Alice)

I want to access

4711	read,write
Friends	read

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

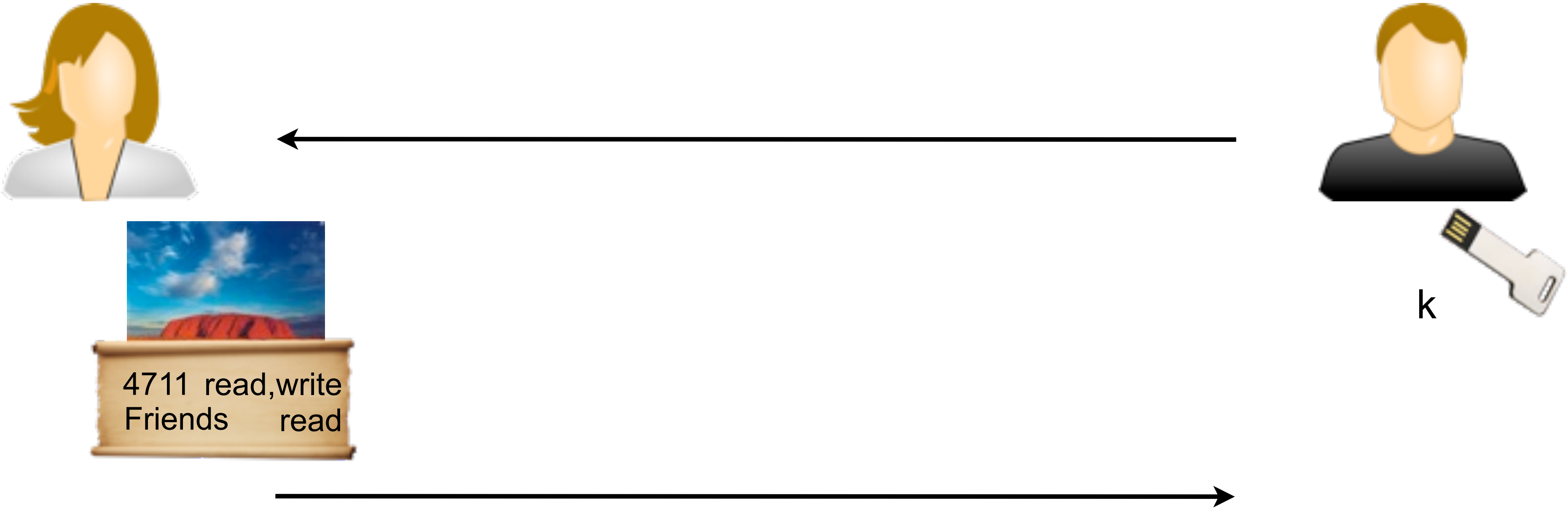
Retrieving Resources: Full Protocol



4711 read,write
Friends read

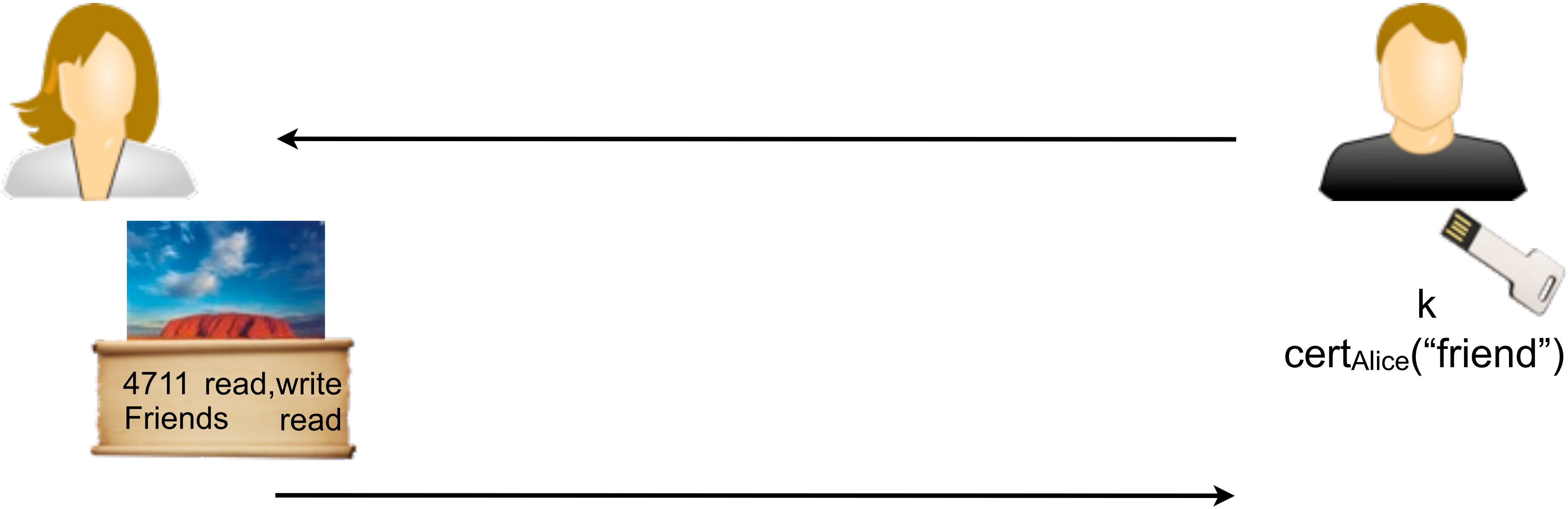
Choose random key k

Retrieving Resources: Full Protocol



4711	read,write
Friends	read

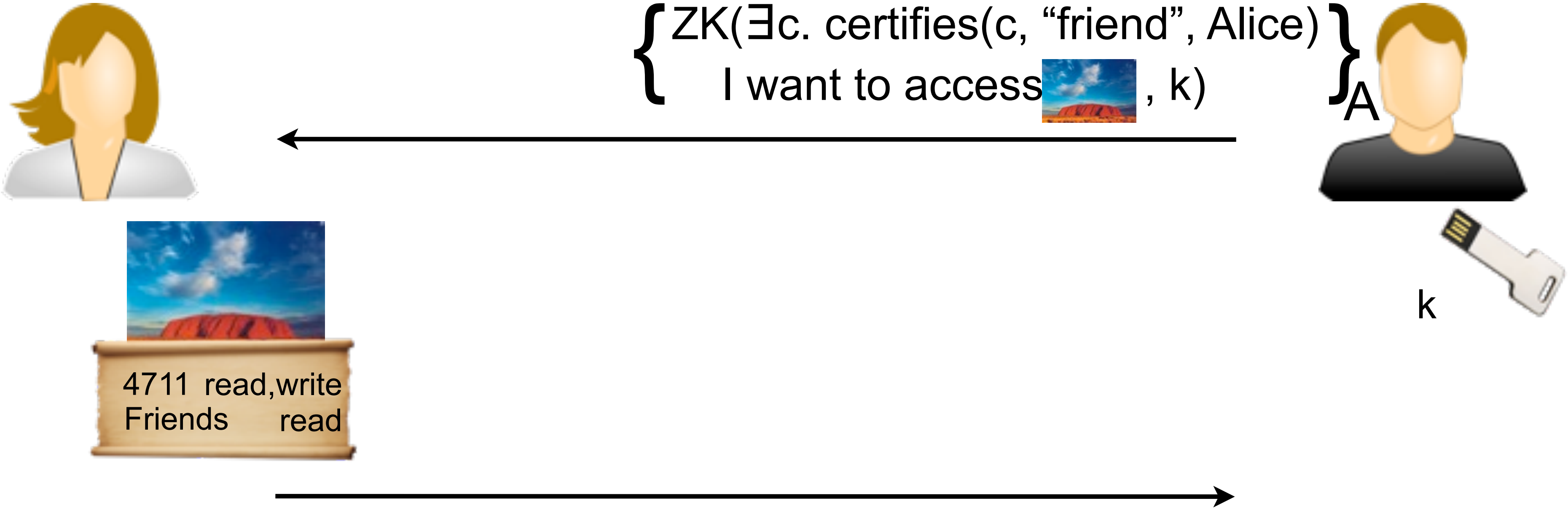
Retrieving Resources: Full Protocol



4711 read,write
Friends read

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

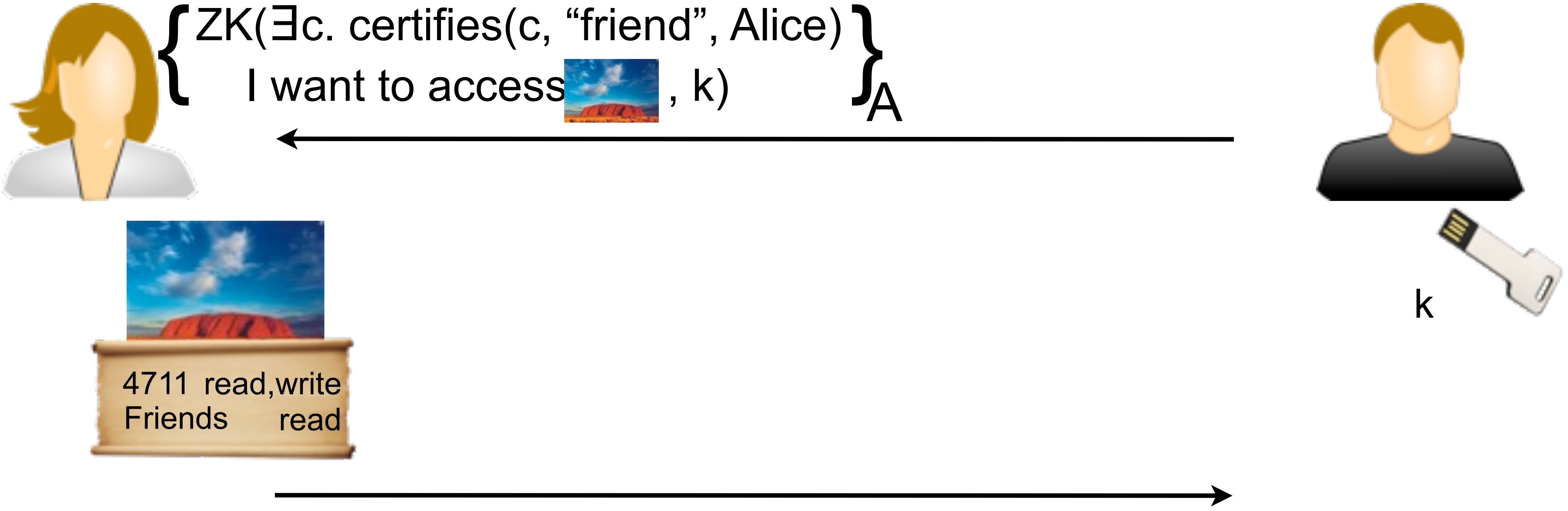
Retrieving Resources: Full Protocol



$\{$ZK(∃c. certifies(c, “friend”, Alice) I want to access , k)$\}_A$

4711	read,write
Friends	read

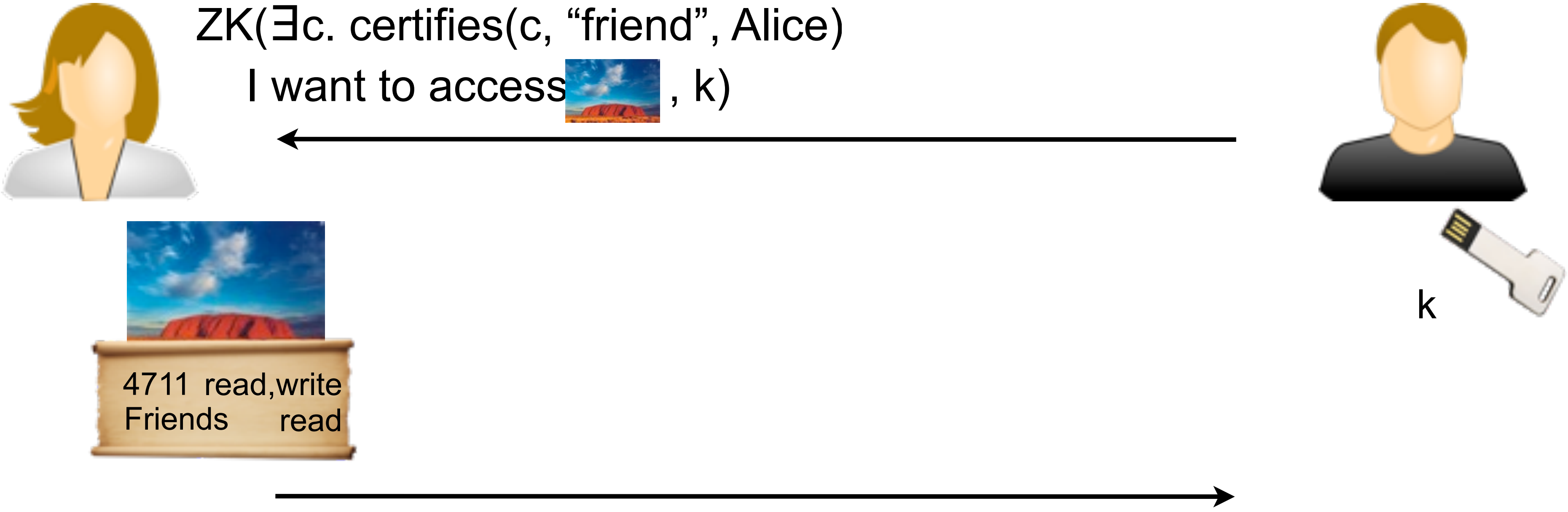
Retrieving Resources: Full Protocol



$\{$ZK(∃c. certifies(c, “friend”, Alice)
I want to access , k)$\}_A$

4711	read,write
Friends	read

Retrieving Resources: Full Protocol



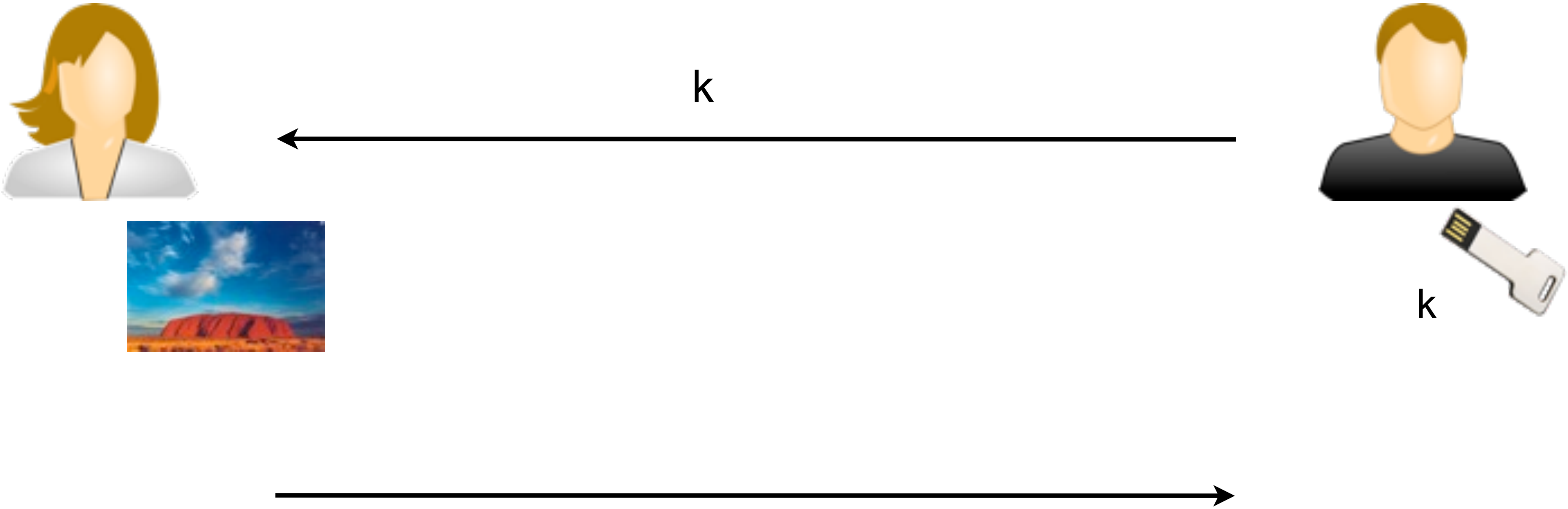
ZK(∃c. certifies(c, “friend”, Alice)

I want to access , k)

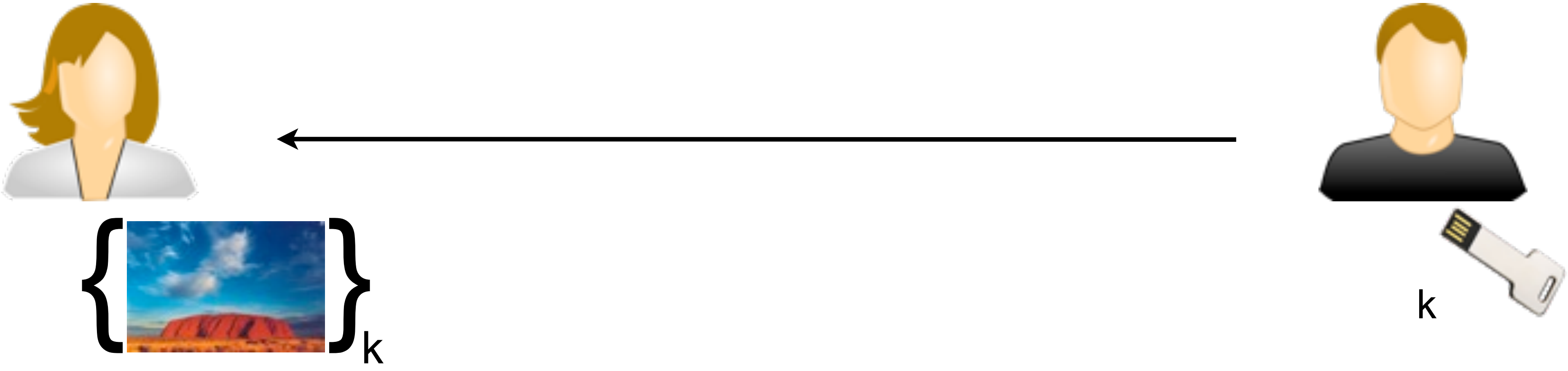
k

4711	read,write
Friends	read

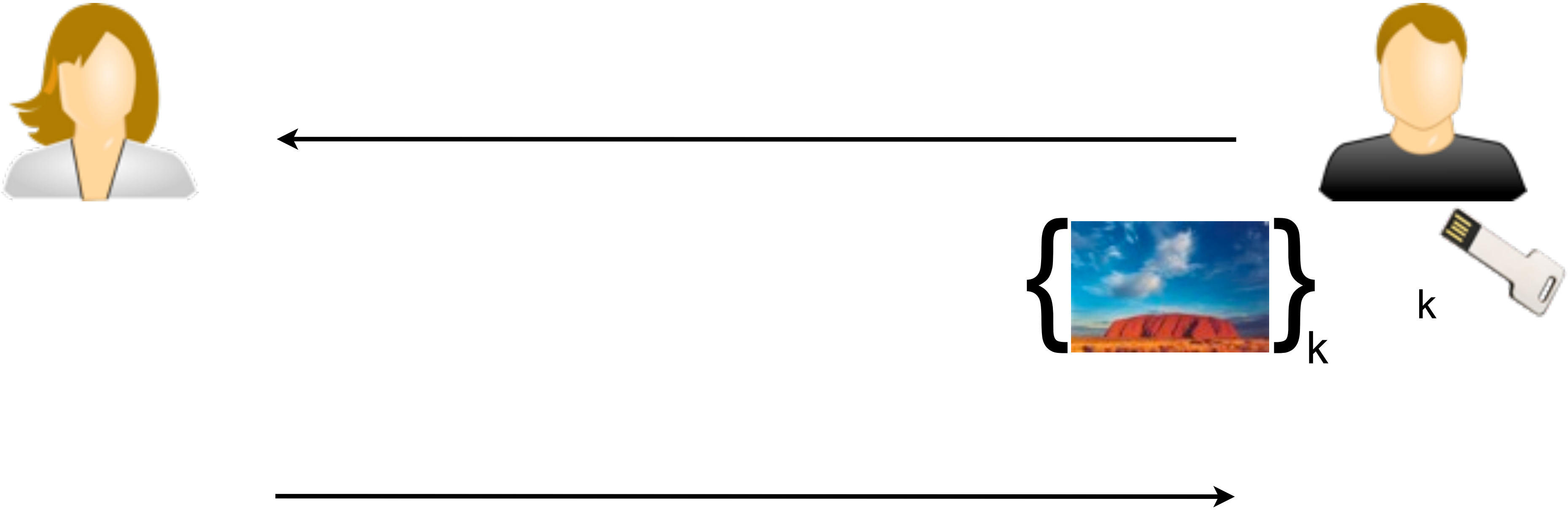
Retrieving Resources: Full Protocol



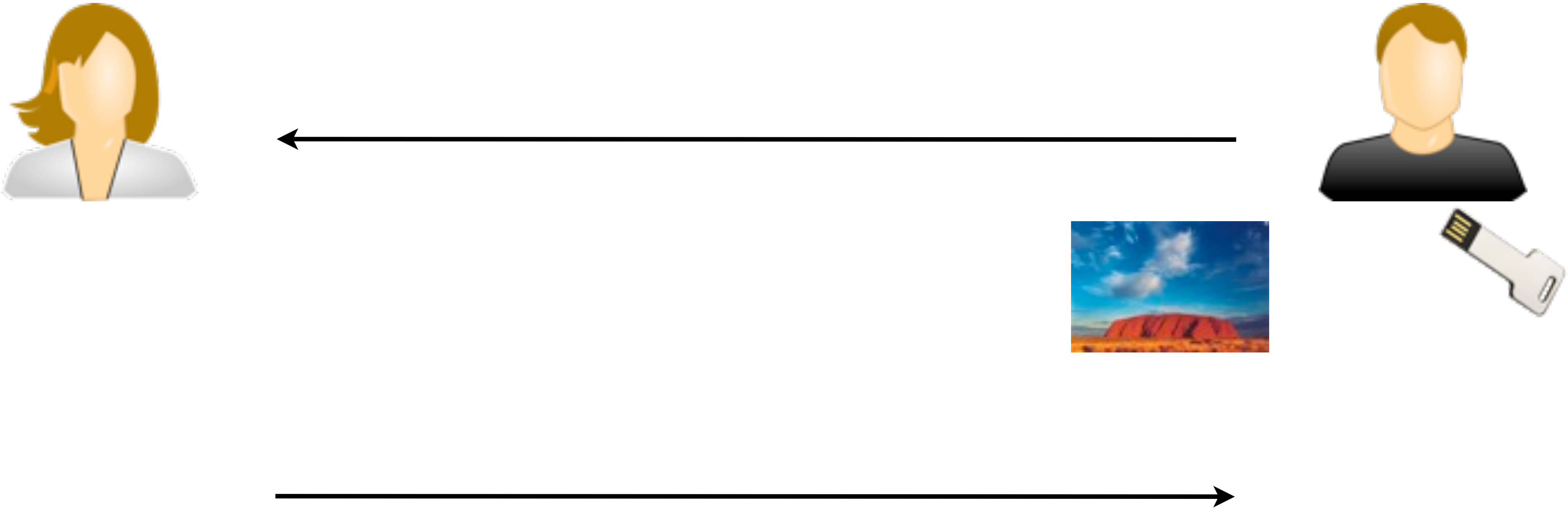
Retrieving Resources: Full Protocol



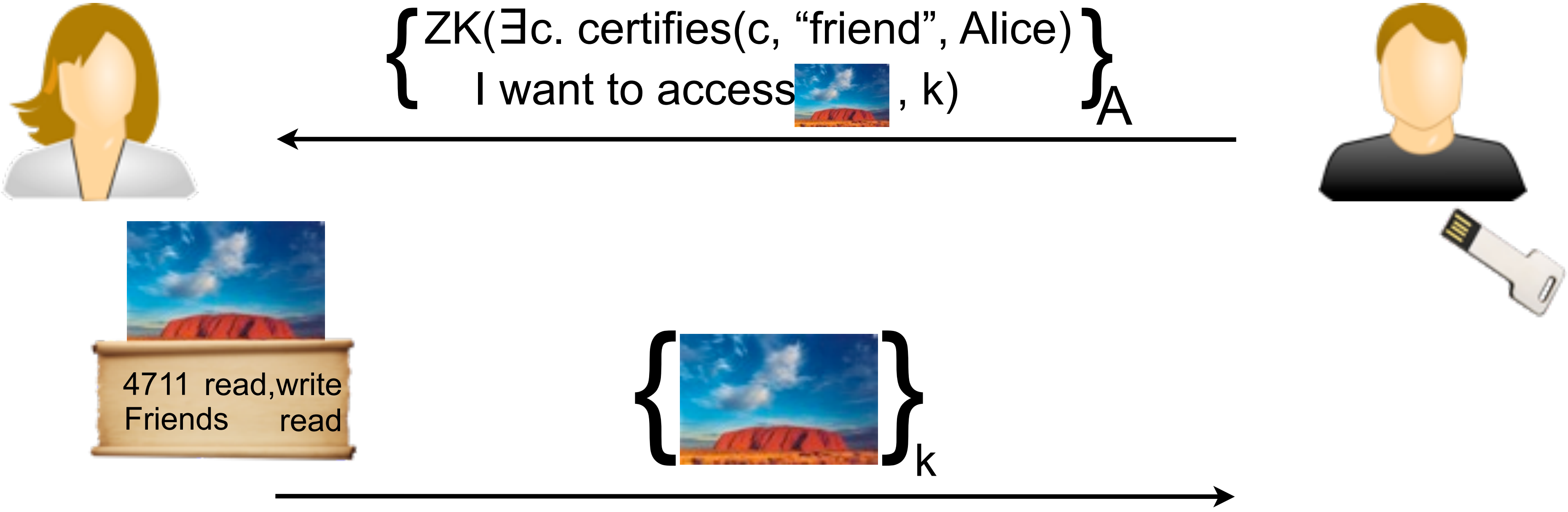
Retrieving Resources: Full Protocol



Retrieving Resources: Full Protocol



Retrieving Resources: Full Protocol

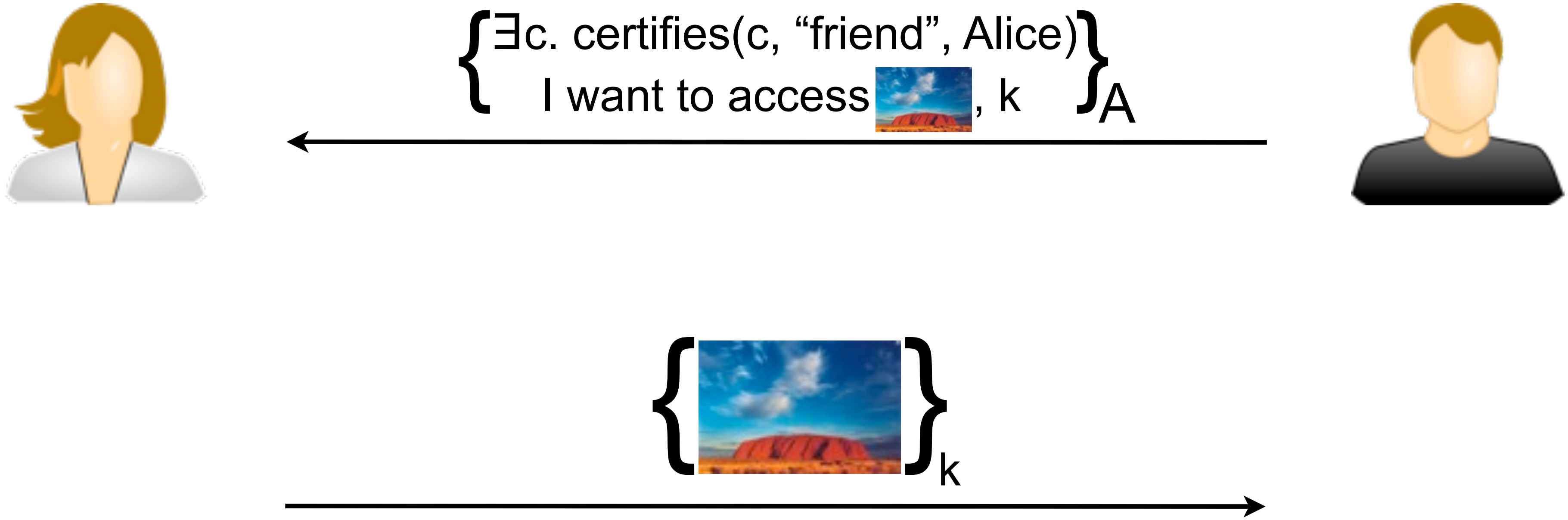


$\{$ZK($\exists$c. certifies(c, “friend”, Alice), I want to access , k)$\}_A$

$\{\ \}_k$

- Full protocol incorporates encryption
  - Asymmetric encryption ensures data privacy
  - Symmetric encryption facilitates anonymity of requester (Bob)

Resistance to Outside Attackers



$$\left\{ \begin{array}{c} \exists c.\ \text{certifies}(c, \text{“friend”}, \text{Alice}) \\ \text{I want to access}\ \ , k \end{array} \right\}_A$$

$$\{\ \}_k$$

Network traffic looks random

Resistance to Compromise

4711	read,write
Friends	read

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

Resistance to Compromise

4711	read,write
Friends	read

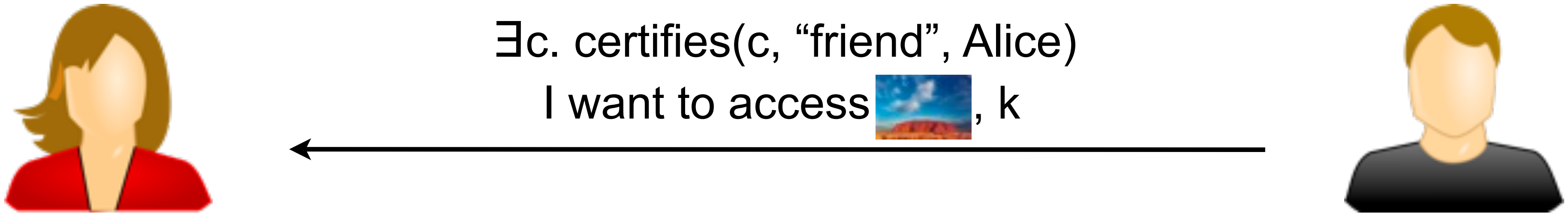
- Certificates on pseudonyms/social relations on secure device
- Pseudonym-user bindings stored on secure device
- Resources will be leaked
- ACL

Resistance to Compromise

4711	read,write
Friends	read
7331	write
1234	read,write

- 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



∃c. certifies(c, “friend”, 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

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

Automated Formal Verification

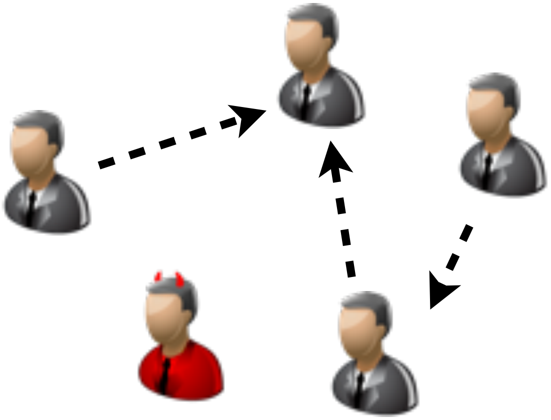
- Hand-made proofs error-prone

- 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

Formal Verification: Access Control

- Attacker model:
  - Attacker controls network topology
    - Number of principals
    - Social relations
  - Attacker dictates which protocols to run
    - Corrupted principals allowed



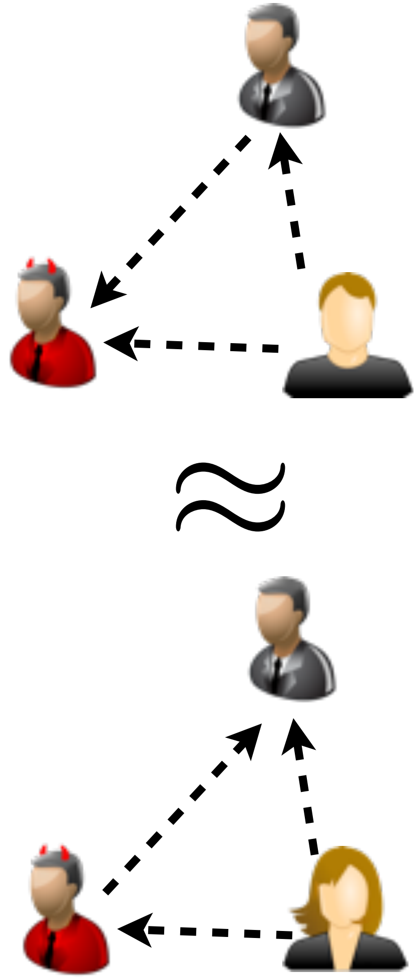
- Trace-based verification
  - Proven access control for all protocols ✓

Formal Verification: Anonymity

- 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 ✓



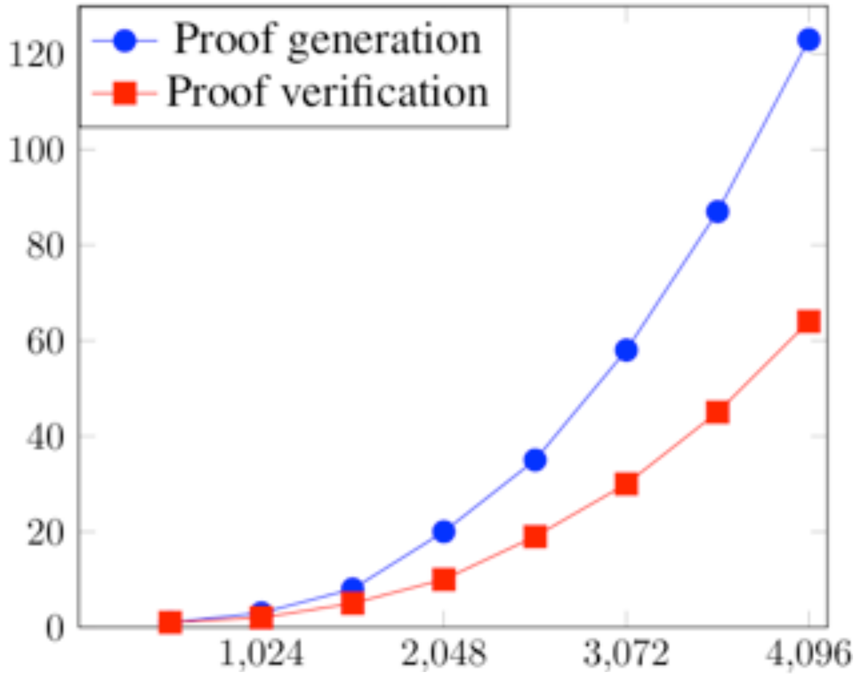
Experimental Evaluation

- 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)

Experimental Evaluation

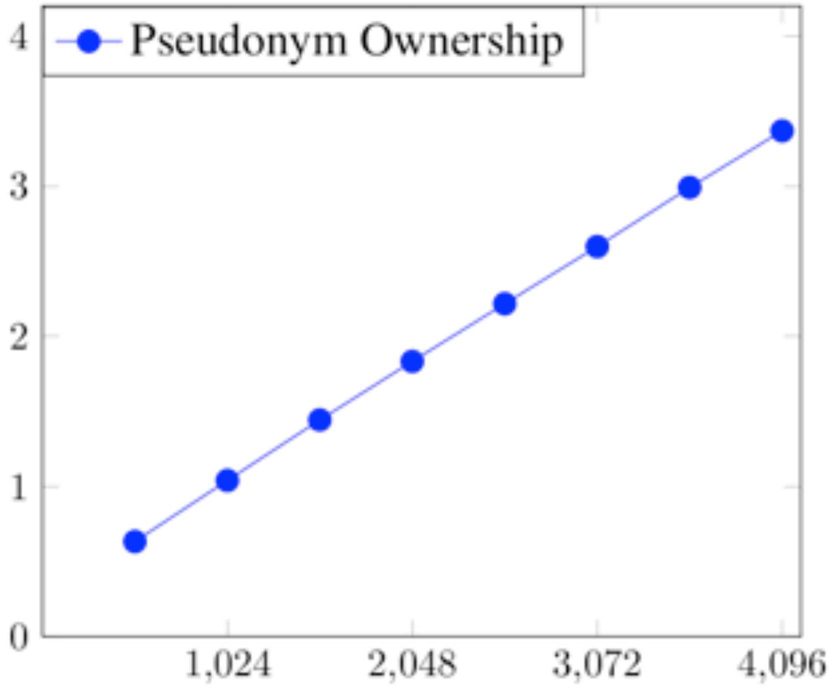
$$\exists x.\ g^{x} = 4711$$

Time in ms



Pseudonym size in bits

Size in kB

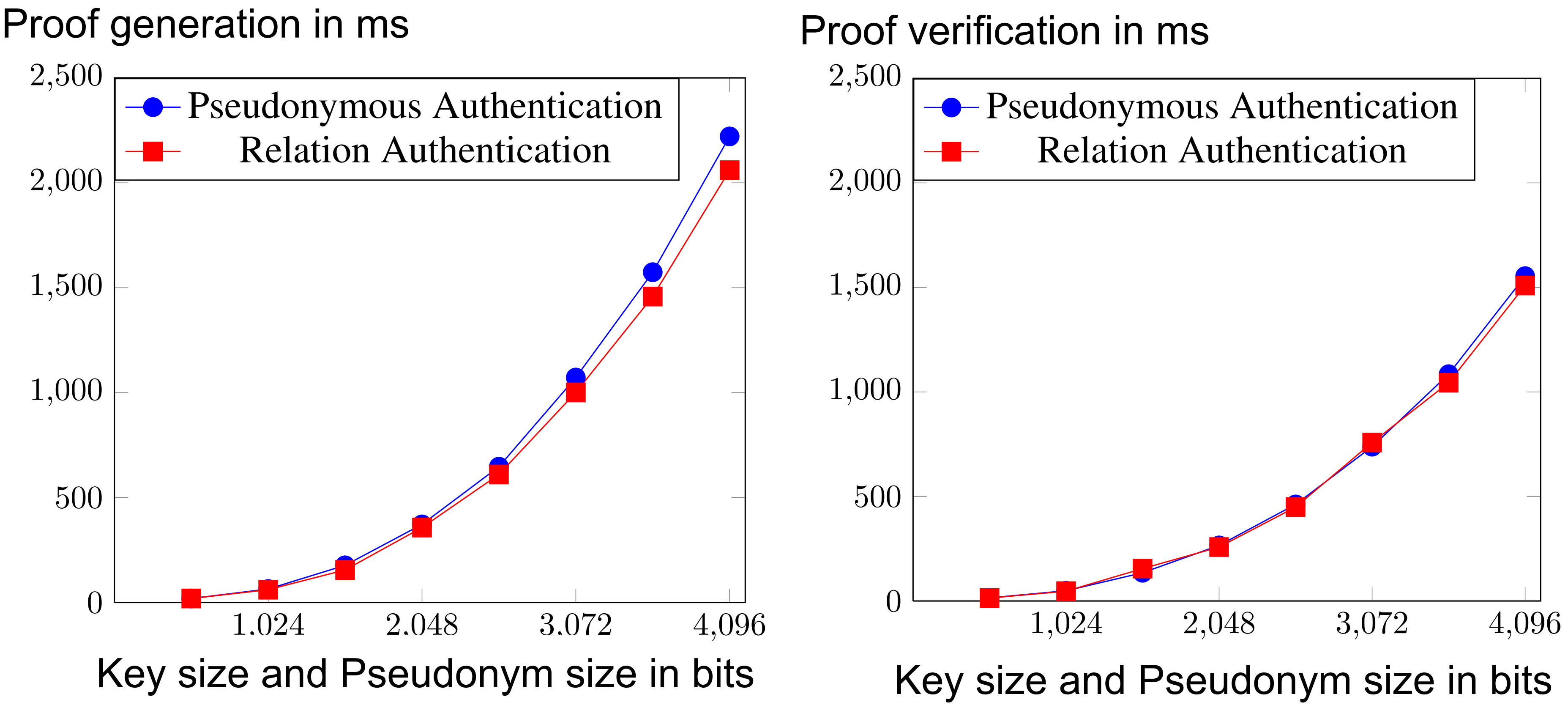


Pseudonym size in bits

Experimental Evaluation

$\exists c,x.\ \mathrm{certifies}(c, 4711, \mathrm{Alice}) \wedge g^x = 4711$

∃c. certifies(c,”friend”,Alice)

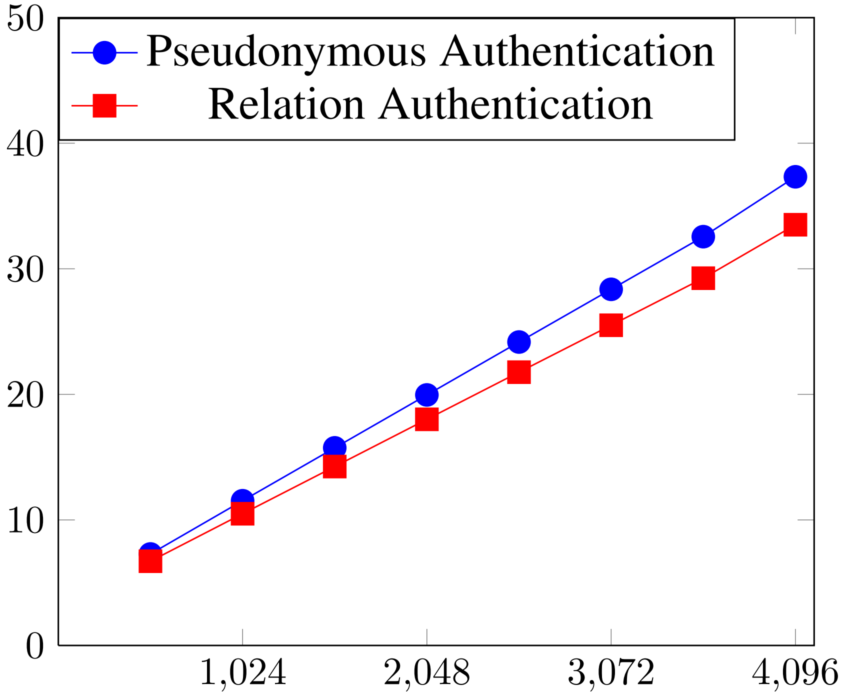


Experimental Evaluation

∃c,p. certifies(c, 4711, Alice) ∧ owns(p, 4711)

∃c. certifies(c,”friend”,Alice)

Proof size in kB

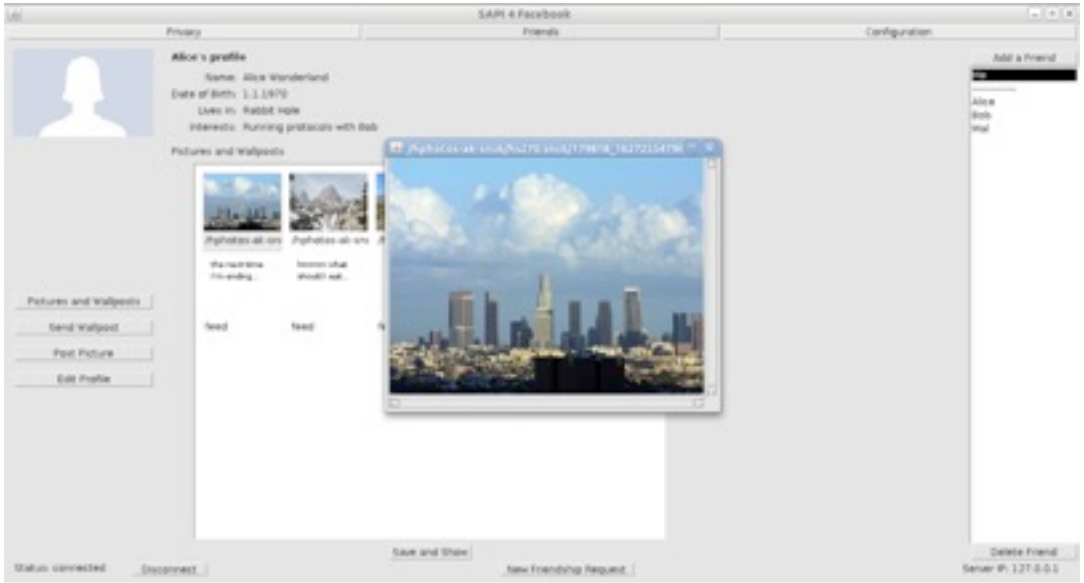


Key size and Pseudonym size in bits

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



Alice's profile

Name: Alice Wonderland
Date of Birth: 1.1.1970
Lives in: Rabbit Hole
Interests: Running protocols with Bob

Status: connected

Server IP: 127.0.0.1

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