# cryptography for peer-to-peer online social media

Ruben De Smet  Ann Dooms  An Braeken  Jo Pierson

12th November 2018

VRIJE UNIVERSITEIT BRUSSEL

# CLONING FACEBOOK

Components:

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)
- relational and/or NoSQL database (PostgreSQL, MySQL, …, MongoDB, …)

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)
- relational and/or NoSQL database (PostgreSQL, MySQL, …, MongoDB, …)
- back-end stack (PHP with Kohana, Ruby on Rails, …)

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)
- relational and/or NoSQL database (PostgreSQL, MySQL, …, MongoDB, …)
- back-end stack (PHP with Kohana, Ruby on Rails, …)
- front-end stack (React, Angular, …)

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)
- relational and/or NoSQL database (PostgreSQL, MySQL, …, MongoDB, …)
- back-end stack (PHP with Kohana, Ruby on Rails, …)
- front-end stack (React, Angular, …)

Let’s make it private.

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)
- relational and/or NoSQL database (PostgreSQL, MySQL, …, MongoDB, …)
- back-end stack (PHP with Kohana, Ruby on Rails, …)
- front-end stack (React, Angular, …)

Let's make it private.

- Give users a great ToS

# CLONING FACEBOOK

Components:

- web server stack (nginx, Apache, …)
- relational and/or NoSQL database (PostgreSQL, MySQL, …, MongoDB, …)
- back-end stack (PHP with Kohana, Ruby on Rails, …)
- front-end stack (React, Angular, …)

Let's make it private.

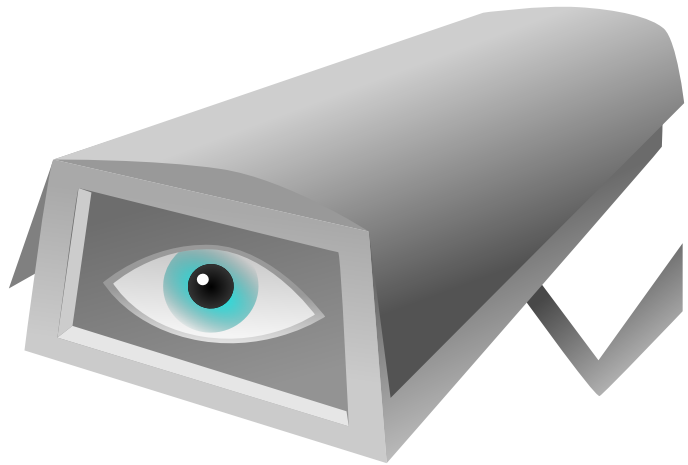
- Give users a great ToS; OR
- give users control over their data: decentralisation.

# cloning Facebook: private version

VRIJE
UNIVERSITEIT
BRUSSEL

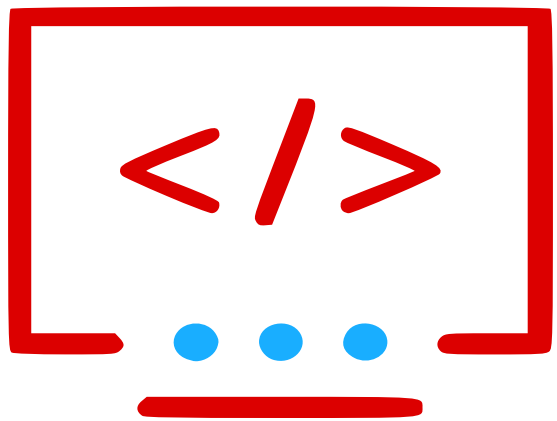
GOAL

private



GOAL

extensible



GOAL

performant

# STATE-OF-THE-ART

**federated** Mastodon, Diaspora*;

# STATE-OF-THE-ART

**federated** Mastodon, Diaspora*;

**peer-to-peer** overlay (PeerSoN, Buchegger et al., 2009), friend-to-friend (RetroShare).

# PEER-TO-PEER

## FRIEND-TO-FRIEND: ``RETROSHARE''

friend-to-friend (f2f), 100 % decentralised
Since October 2017: GXS, “Generic data eXchange System” (Soler, 2017):

**Services** defines groups

**Groups** a structured collection of messages

**Messages** hierarchical data items belonging to a group

**Identities** an “account”, user identification

**Circles** a set of identities

# introducing glycos

VRIJE
UNIVERSITEIT
BRUSSEL

# GOAL

- developer convenience: abstractions, tools; AND
- generality of data model; AND
- guarantees on privacy

# GOAL

- developer convenience: abstractions, tools; AND
- generality of data model; AND
- guarantees on privacy

additionally: performance $\Rightarrow$ mobile friendliness.

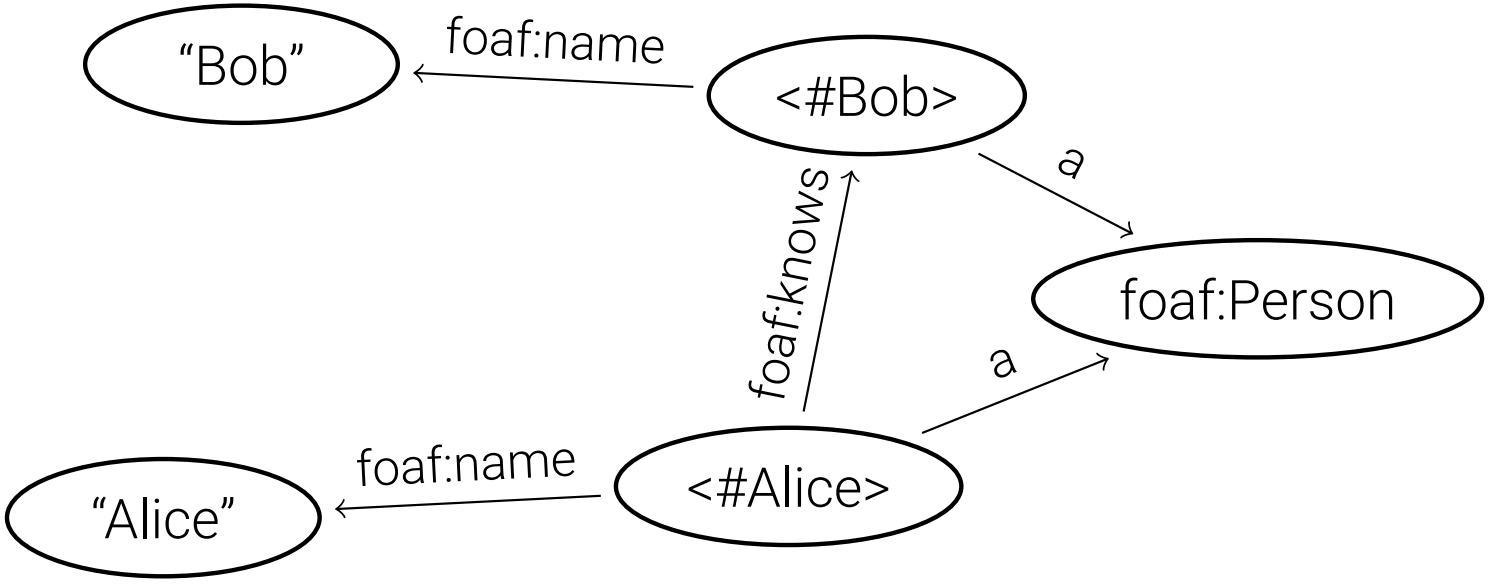



Figure: An example RDF graph. This graph signifies that Alice, a person, knows Bob, another person. Both persons have a name and type.

# GRAPH DATABASES (CONT.)

Graph databases are well studied (Angles & Gutierrez, 2008; Lanthaler, Cyganiak, & Wood, 2014; Lassila & Swick, 1997).

They can represent arbitrary structured data.

Glycos couples RDF graphs with modern cryptography to provide access control and anonymity w.r.t. outsiders.

# EXAMPLE

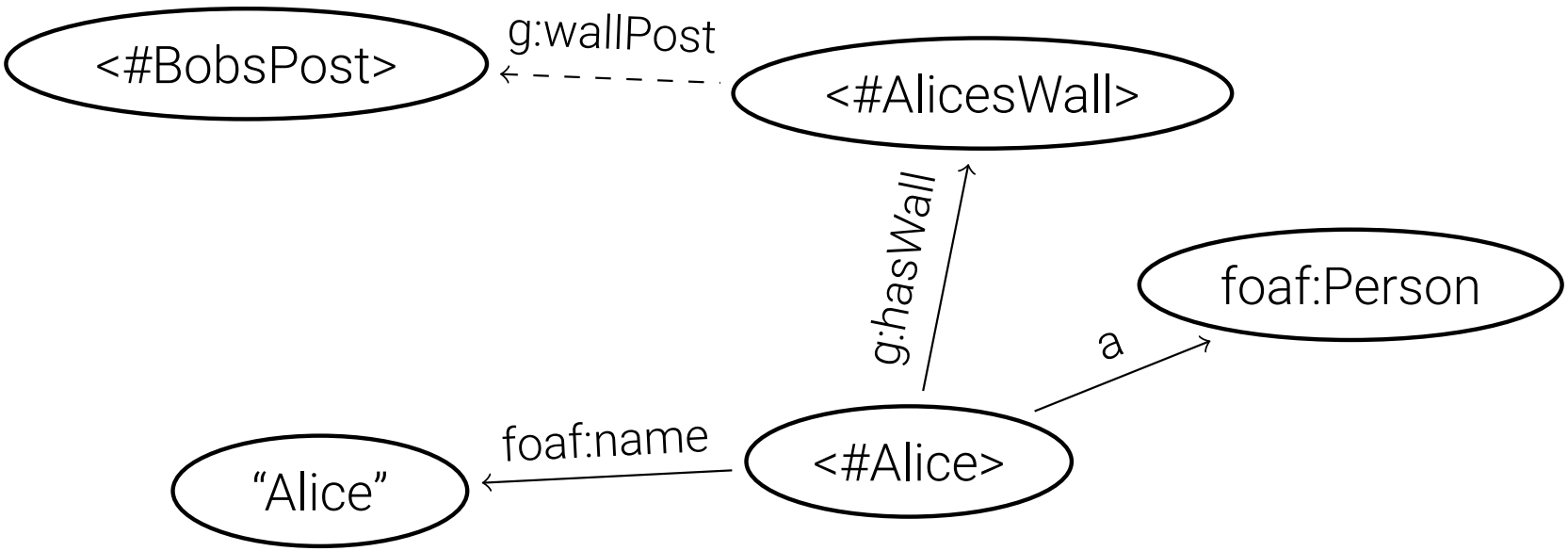



Figure: Bob writes a message on Alice's wall. This is only possible if Alice has granted Bob the rights to do so; otherwise, the network will not accept Bobs post (`<#BobsPost>`). The definition of those access rights are contained within every vertex.

On the `<#AlicesWall>` vertex, Alice has defined who are allowed to append other vertices.

# PSEUDONYMISATION

Bob generates an ephemeral (one-time) public key for Alice (with $A = aG$) as follows:

$$\begin{aligned} r &\leftarrow [0, \ell - 1] \\ R &\leftarrow rG, \\ pk_{\text{OT}}^{\text{alice}} &\leftarrow \mathcal{H}_s(rA)G + A, \end{aligned}$$

Alice can recover her private key using

$$sk_{\text{OT}}^{\text{alice}} \leftarrow \mathcal{H}_s(aR) + a.$$

# PSEUDONYMISATION

## PROPERTIES

Call $R$ recogniser: Alice can "recognise" a key belonging to her:

$$pk_{\mathrm{OT}}^{\mathrm{alice}} \leftarrow \mathcal{H}_s(aR)G + A,$$

$pk_{\mathrm{OT}}^{\mathrm{alice}}$ is indistinguishable from random: trivially reducible to the decisional Diffie-Hellman assumption.

# EDGES AND VERTICES

Vertex

- Identified by a random owner key;
- Contains a pseudonymised ACL.

# EDGES AND VERTICES

Vertex

- Identified by a random owner key;
- Contains a pseudonymised ACL.

Edge

- Encrypted “predicate” and “object”;
- ring signature over the subject’s ACL.

# RING SIGNATURES

Ring signatures (Rivest, Shamir, & Tauman, 2001) prove knowledge of one key in a set *R*.

# RING SIGNATURES

Ring signatures (Rivest et al., 2001) prove knowledge of one key in a set *R*.

Provides unlinkability: two edges from the same author are undistinguishable from two edges from different authors.

# ORIGINAL RST RING SIGNATURES

Based on RSA trapdoors. Define a "combining function":

$$C_{k,v}(y_1, y_2, \ldots, y_r)$$

Such that:

- Efficiently solved for any $y_i$ for fixed $y_j, j \neq i$
- Infeasible to solve without trapdoor information in $C_{k,v}(g_1(x_1), g_2(x_2), \ldots, g_r(x_r)) = z$

Original scheme:

$$C_{k,v}(y_1, y_2, \ldots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\cdots \oplus E_k(y_1 \oplus v) \ldots))))$$

# TOWARDS ECC RING SIGNATURES

$$C_{k,v}(y_1, y_2, \ldots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\cdots \oplus E_k(y_1 \oplus v) \ldots))))$$

- Based on RSA trapdoors: one of the $y_i = g_i(x_i)$ needs to be efficiently invertible. Maps exceptionally bad to ECC.
- The PRF $E_k$ requires input length $2^b$ equal to output length (often usage of a hash function instead[1])

[1] Breaks anonymity proof outside of ROM

# TOWARDS ECC RING SIGNATURES

$$C_{k,v}(y_1, y_2, \ldots, y_r) = E_k(y_r \oplus E_k(y_{r-1} \oplus E_k(y_{r-2} \oplus E_k(\cdots \oplus E_k(y_1 \oplus v) \ldots))))$$

- Based on RSA trapdoors: one of the $y_i = g_i(x_i)$ needs to be efficiently invertible. Maps exceptionally bad to ECC.
- The PRF $E_k$ requires input length $2^b$ equal to output length (often usage of a hash function instead[1])

Abe et al., 2002 present some interesting schemes, one of which:

$$\sum_{i=0}^{n-1} c_i \cong \mathcal{H}_s\left(\mathcal{R}_s||m||\left(sG + \sum_{i=0}^{n-1} c_i Y_i\right)\right) \mod q$$

[1] Breaks anonymity proof outside of ROM

# ABE RING SIGNATURES

Based on the idea of (classical) Schnorr signatures (Schnorr, 1991), with $Y_i = x_i G$:

$$e = \mathcal{H}_s(m||sG + eY)$$

(signature is $(s, e)$)

$$\sum_{i=0}^{n-1} c_i \cong \mathcal{H}_s\left(\mathcal{R}_s||m||\left(sG + \sum_{i=0}^{n-1} c_i Y_i\right)\right) \mod q$$

(signature is $(s, c_0, c_1, \ldots, c_{n-1})$)

# IMPLICATIONS AND SUMMARY

Graph databases are well studied, manipulation is **easy**, and they are generic.
Tooling can be provided for developers, (roughly) same abstraction level as web development.

Basic implementation written in Rust, tested cross-platform (Intel/ARM), including networking, cryptography, very basic API.

# RESULTS

Benchmarks[2]:

| | lower bound | median | upper bound |
|---|---|---|---|
| decrypt vertex | 3198 $s^{-1}$ | 3341 $s^{-1}$ | 3493 $s^{-1}$ |
| edge verification | 108 µs | 113 µs | 118 µs |

[2] Intel Xeon, single threaded, 95 % confidence interval

# c&c, q&a

mailto:rubedesm@vub.ac.be @rubdos