KU LEUVEN

Practical Identity-Based Encryption for Online Social Networks

Bringing privacy control to Facebook users

by Stijn Meul

"I'm sure there are better ways to disguise sensitive information, but we don't have a big budget."

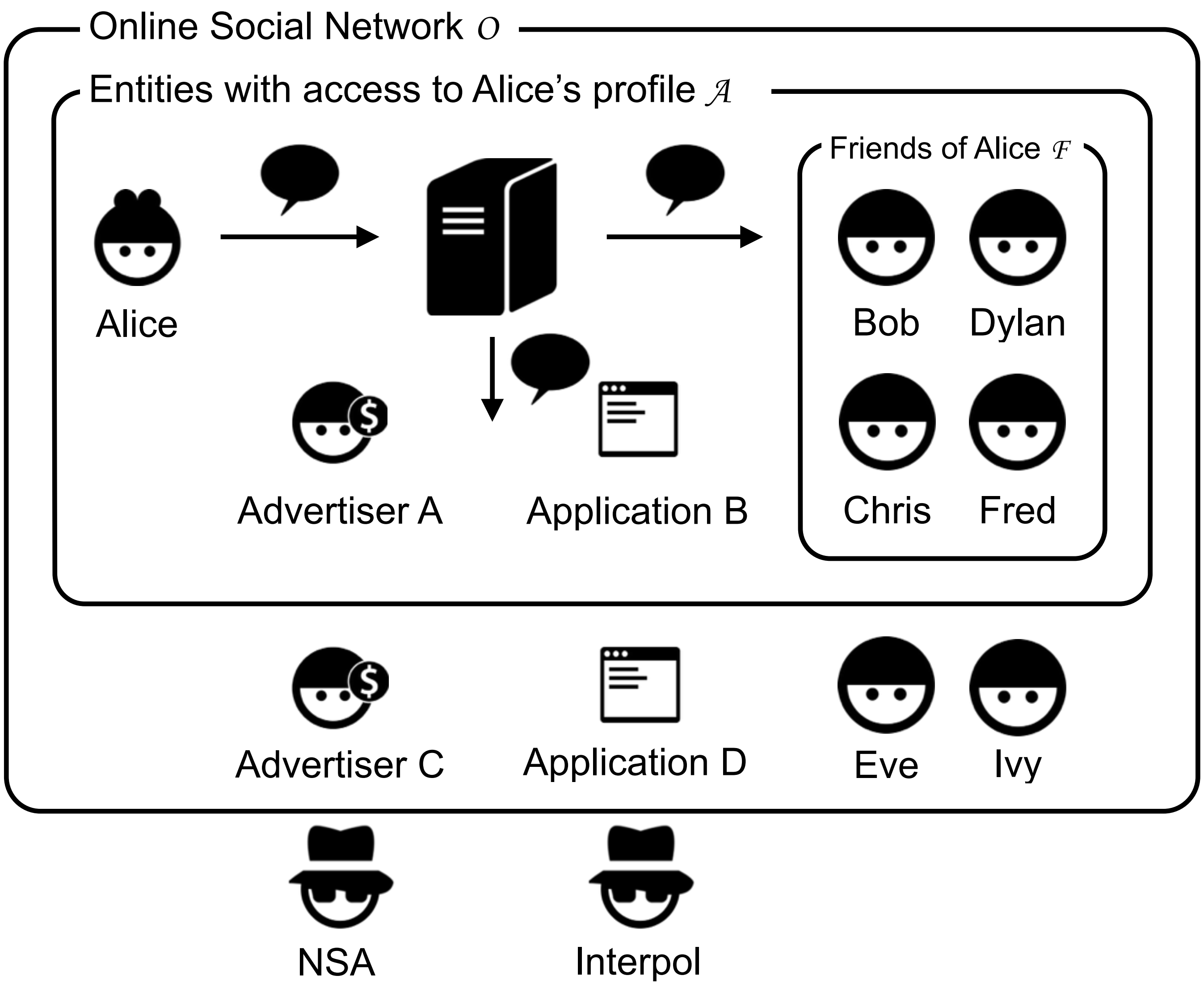
Overview

- Current situation
- Desired situation
- The proposal
  - Setup
  - KeyGen
  - Publish
  - Retrieve
- Implementation Details
- Conclusion
- Live demo

Current Situation

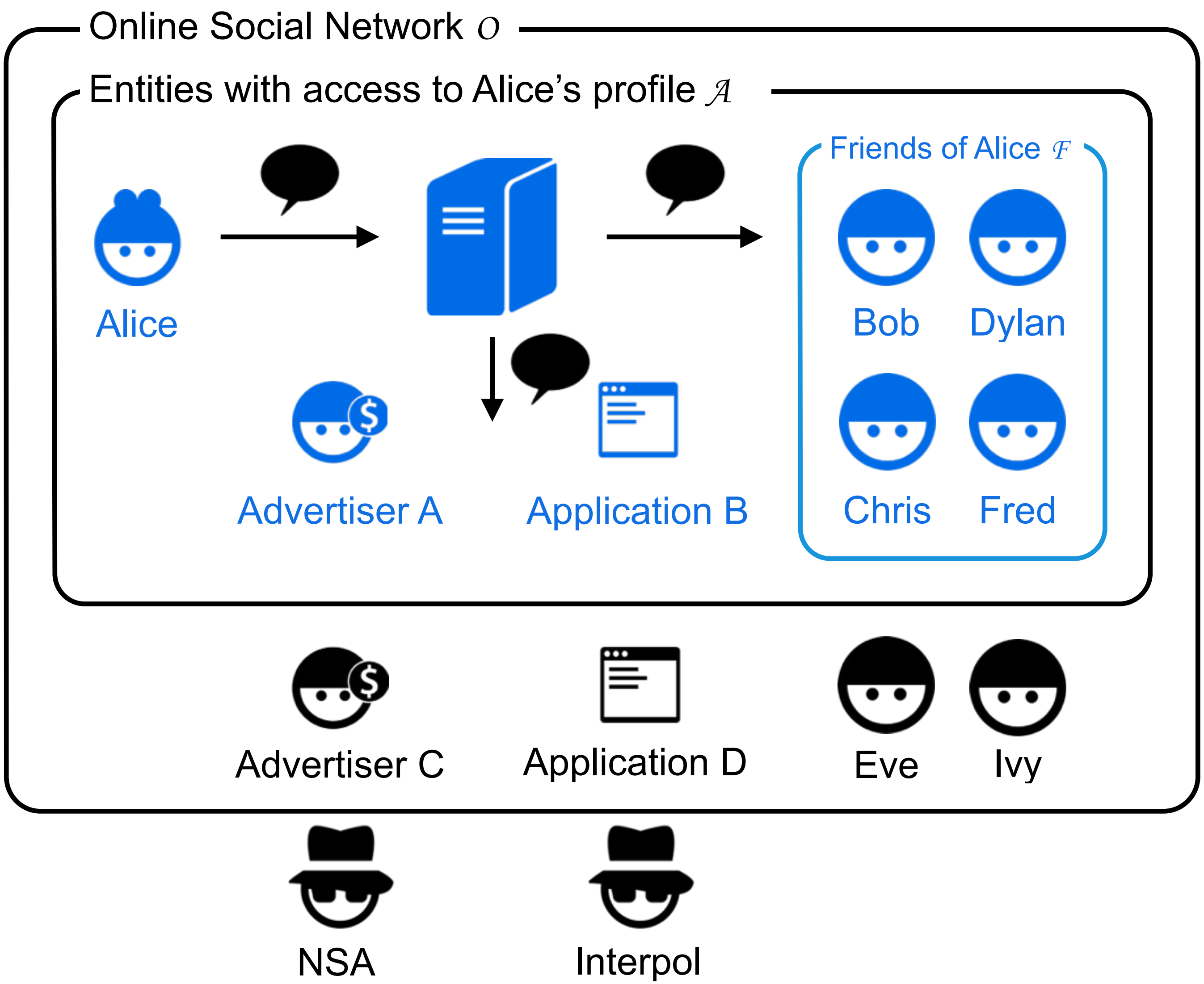
The model as it is today

Model of the current situation



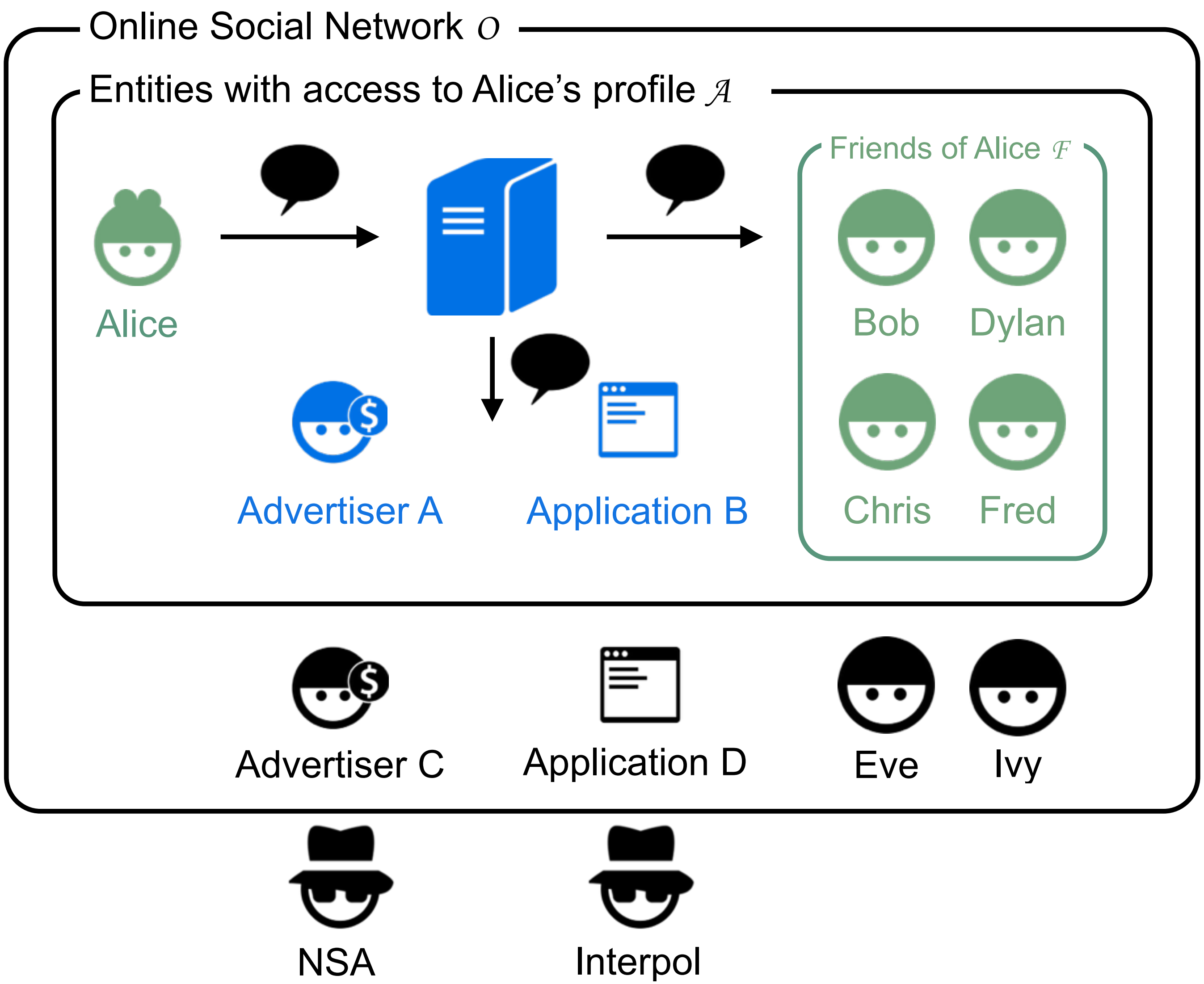
KU LEUVEN

Current situation - Who can read the message



KU LEUVEN

Current situation - What Alice expects

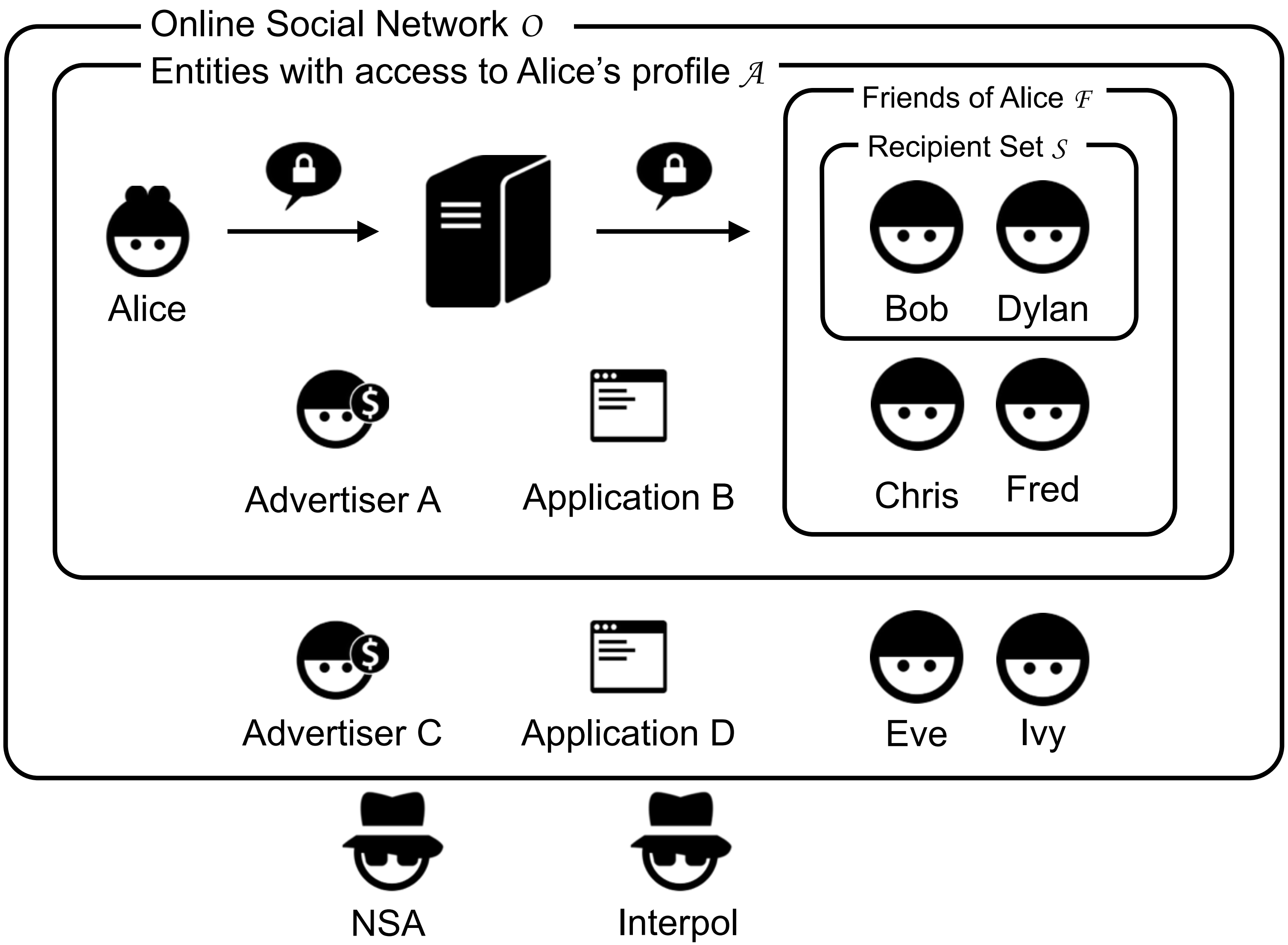


Utopia

The model as it should be

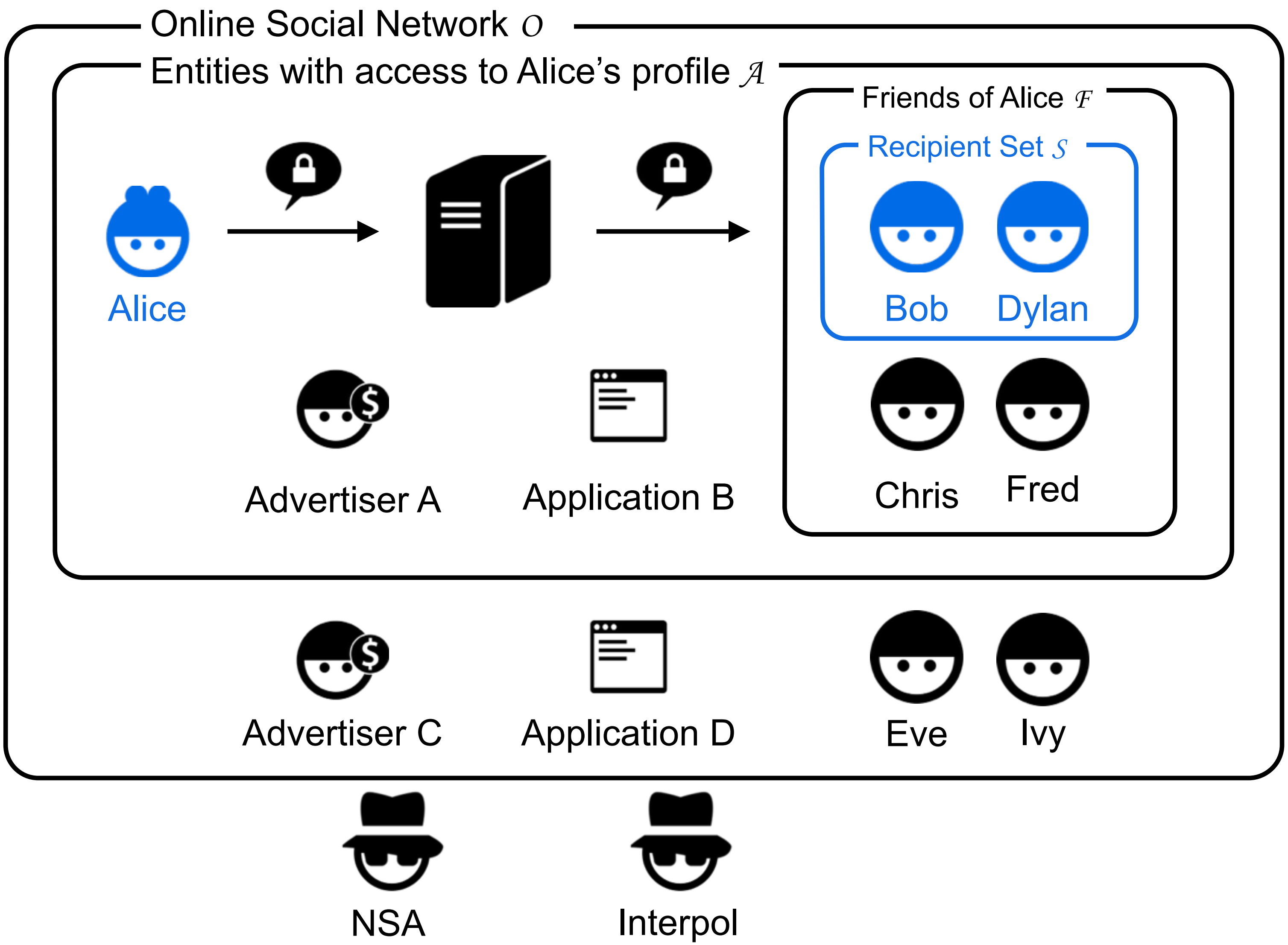
KU LEUVEN

Desired Security Model



KU LEUVEN

Desired Security Model



Design Goals

- **Applicable**

  OSN environment should not be altered
- **User friendly**

  usable for average user
- **Immediately ready to use**

  no additional registration required

KU LEUVEN

Security Goals

- Confidentiality
- (Outsider) Recipient Anonymity
  - Only authorised recipients are aware of the members in the recipient set $\mathcal{S}$
- Data Integrity and authenticity

Non Goals

- Traffic analysis
- Recipient leakage

The Proposal

Achieve Design Goals with IBE

- **Applicable**
  OSN environment should not be altered
  ⇨ use unique profile identifier as a public key

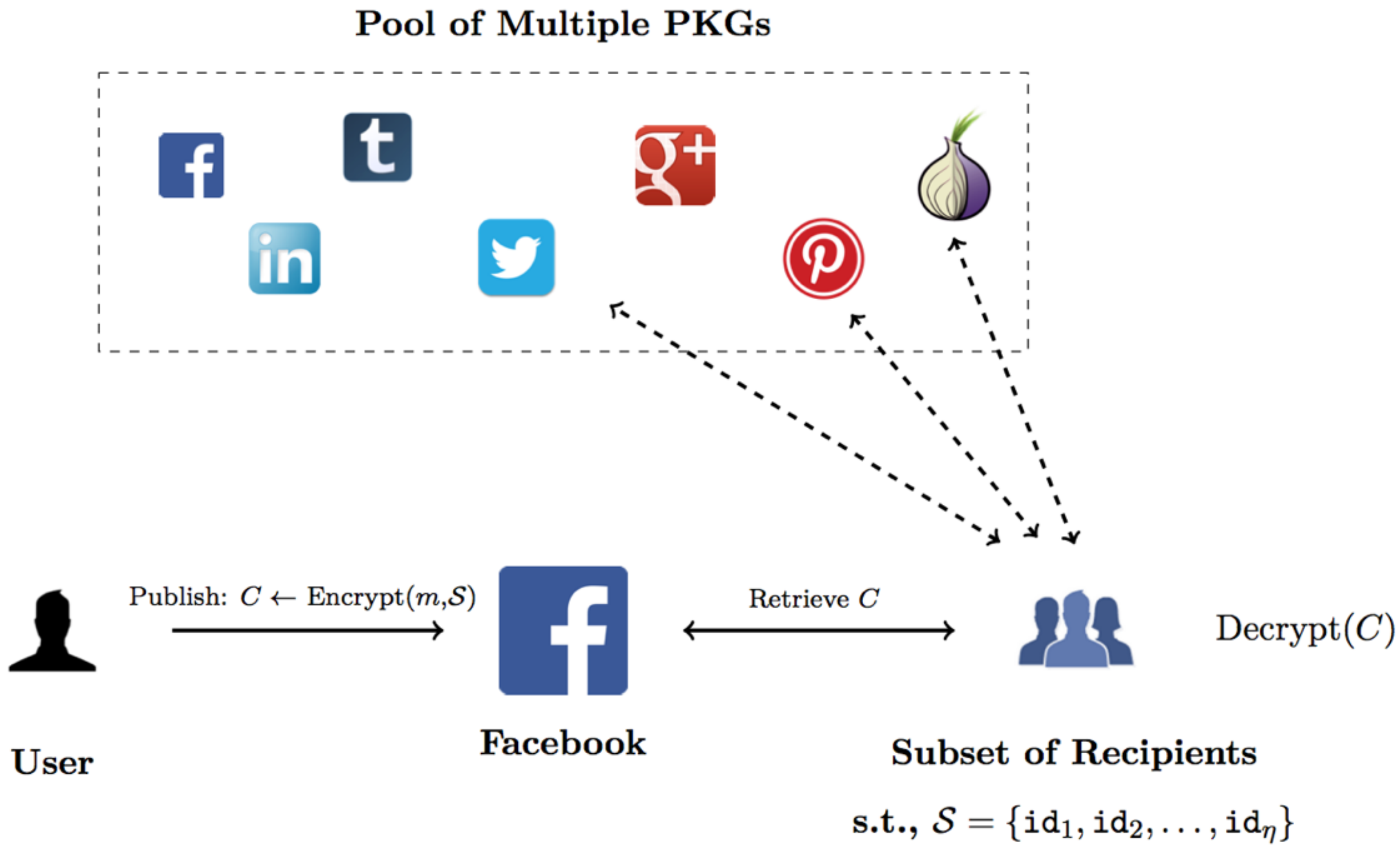
- **User friendly**
  usable for average user
  ⇨ no knowledge of asymmetric cryptography required

- **Immediately ready to use**
  no additional registration required
  ⇨ every user already is assigned a profile identifier in an OSN

- **Cryptographic design goals**
  IBE does not achieve confidentiality since it suffers from key escrow
  ⇨ can be solved using distributed key generation

KU LEUVEN

Overview



Pool of Multiple PKGs
Publish: $C \leftarrow \mathrm{Encrypt}(m, \mathcal{S})$
Retrieve C
Decrypt(C)
User
Facebook
Subset of Recipients
s.t., $\mathcal{S} = \{\mathtt{id}_1, \mathtt{id}_2, \ldots, \mathtt{id}_\eta\}$

Generic Algorithms

- **Setup(**λ, t, n**):**

  Outputs the public *params* of the system with respect to the security parameter λ, the number of PKGs n and the threshold t.

- **KeyGen(***params*, $\{PKG_1, \ldots, PKG_t\}$, id_i**):**

  On input of *params*, a user identifier id_i and a subset of size t PKG servers, generates a valid private key for id_i

- **Publish(***params*, $\mathcal{S}$, m**):**

  On input of *params*, the recipient set $\mathcal{S}$ and a plaintext message m, generates a broadcast message B

- **Retrieve(***params*, sk_{id}, B**):**

  On input of *params*, the private key sk_{id} and a broadcast message B, reconstructs the plaintext message m

KU LEUVEN

Generic Algorithms

- **Setup(**λ, t, n**):**
  Outputs the public *params* of the system with respect to the security parameter λ, the number of PKGs n and the threshold t.
- **KeyGen(***params*, $\{PKG_1, \dots, PKG_t\}$, id_i**):**
  On input of *params*, a user identifier id_i and a subset of size t PKG servers, generates a valid private key for id_i
- **Publish(***params*, $\mathcal{S}$, m**):**
  On input of *params*, the recipient set $\mathcal{S}$ and a plaintext message m, generates a broadcast message B
- **Retrieve(***params*, sk_{id}, B**):**
  On input of *params*, the private key sk_{id} and a broadcast message B, reconstructs the plaintext message m

Setup(λ, t, n)

PKG 1

Security parameter λ, a prime q, two groups G_1 and G_2 of prime q and the bilinear map $e: G_1 \times G_2 \rightarrow G_T$ are fixed.

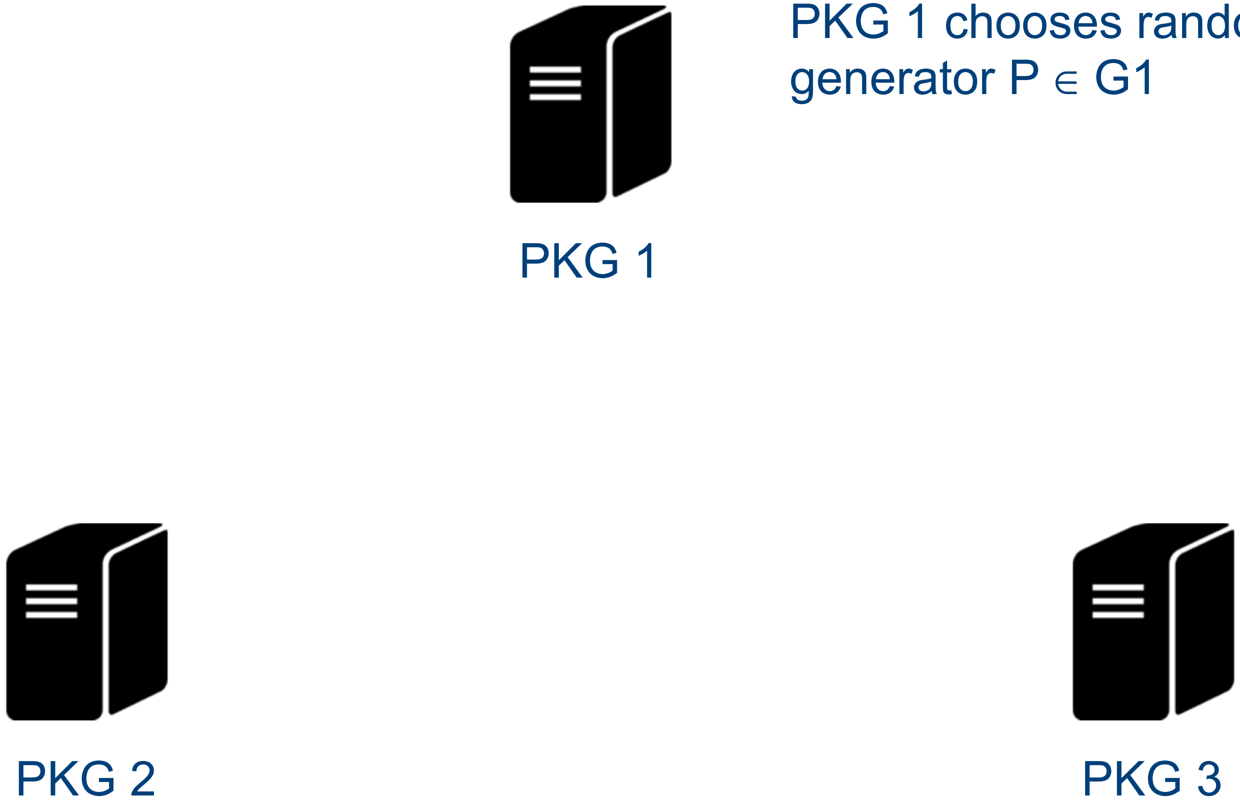
PKG 2

PKG 3

for n = 3, t = 2

Setup(λ, t, n)

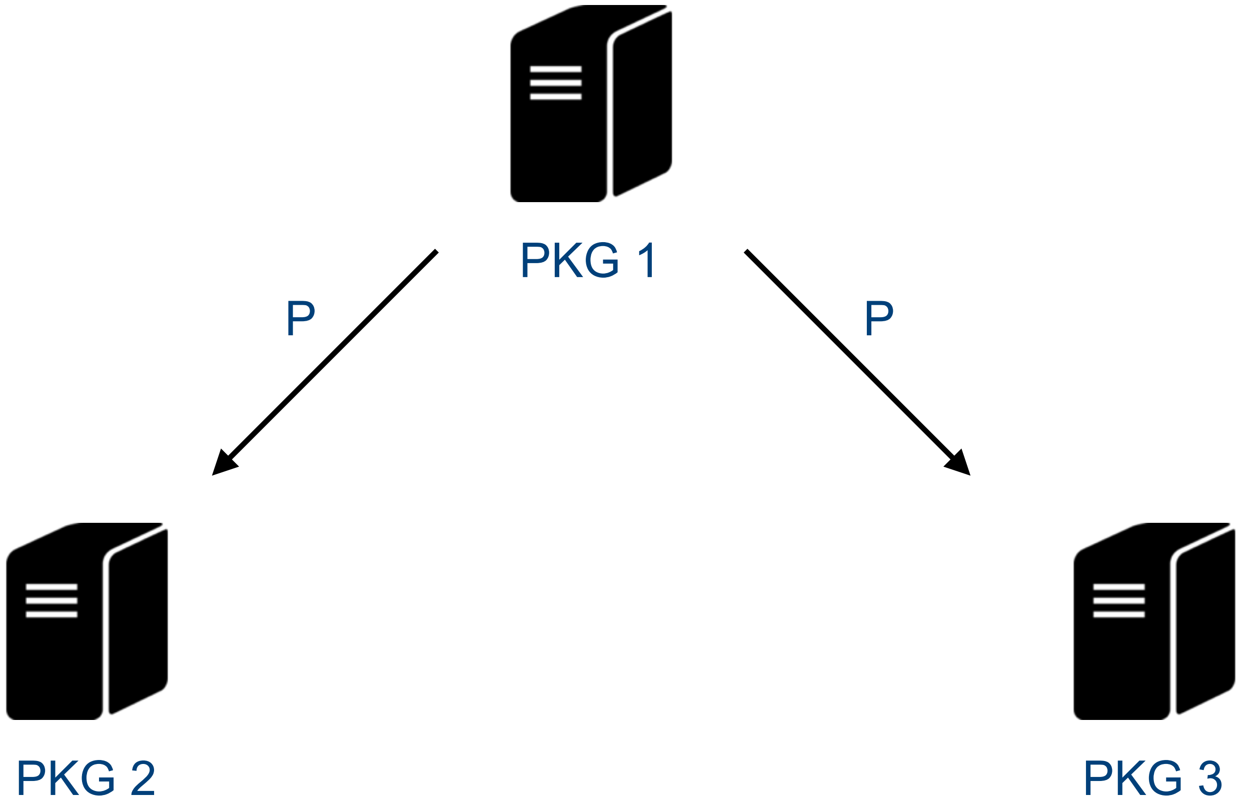
PKG 1 chooses random generator P $\in$ G1



for n = 3, t = 2

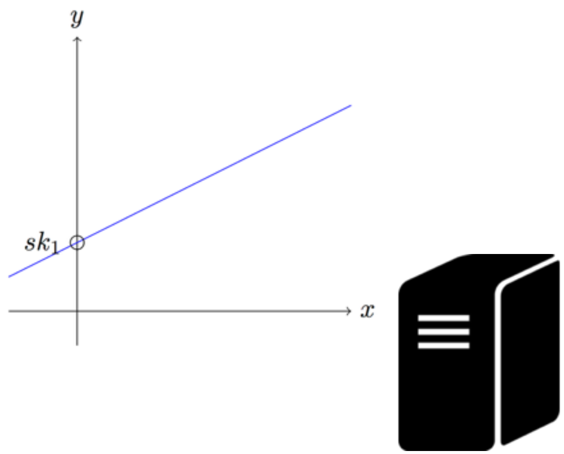
KU LEUVEN

Setup(λ, t, n)



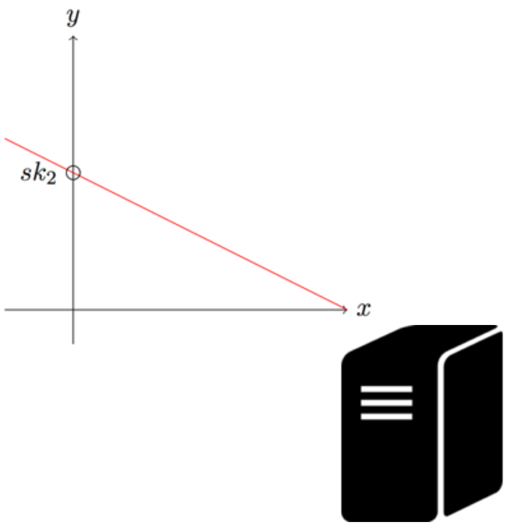
for n = 3, t = 2

KU LEUVEN

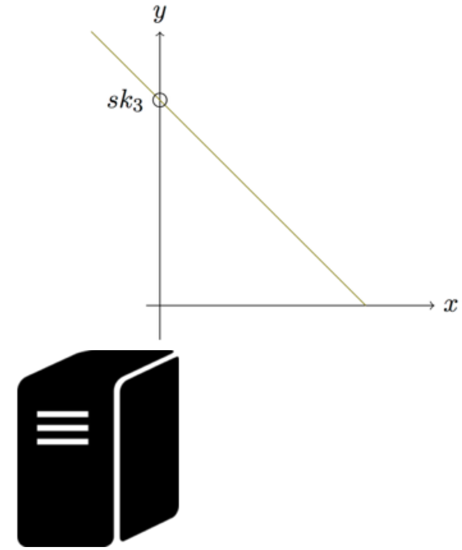


PKG 1

Each PKG picks a random polynomial of degree *t - 1* in $\mathbb{Z}_q$



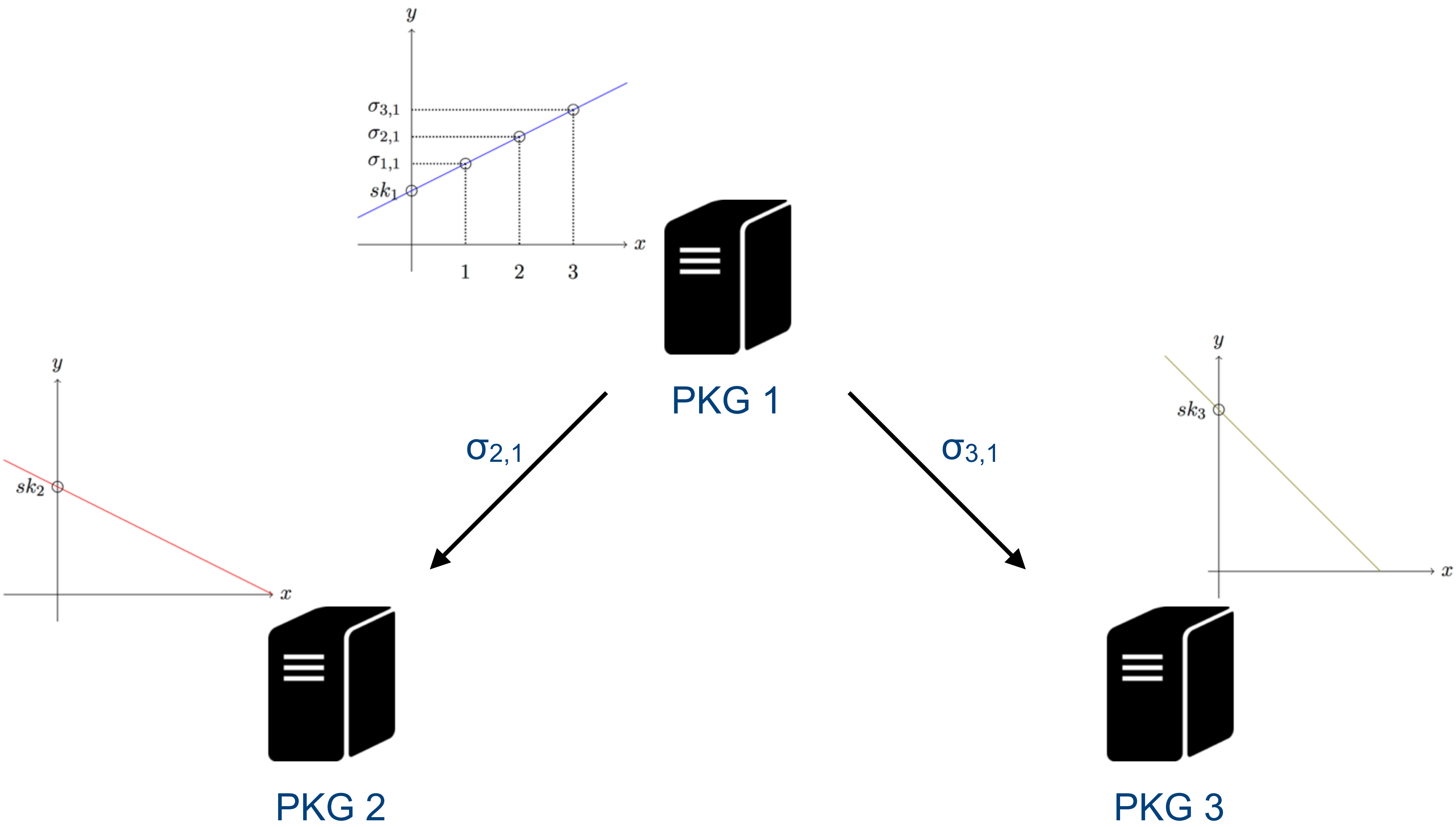
PKG 2



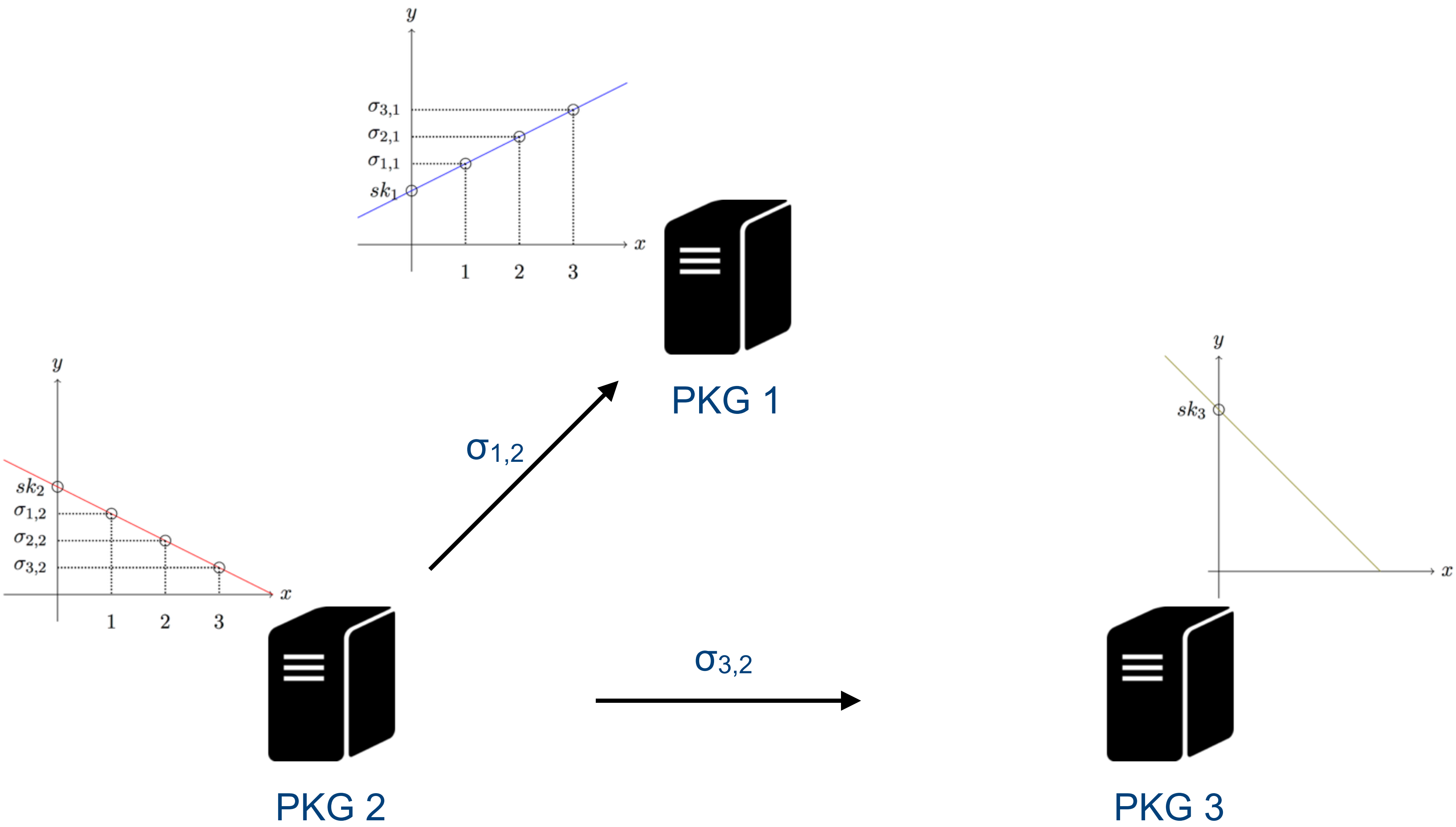
PKG 3

for n = 3, t = 2

KU LEUVEN

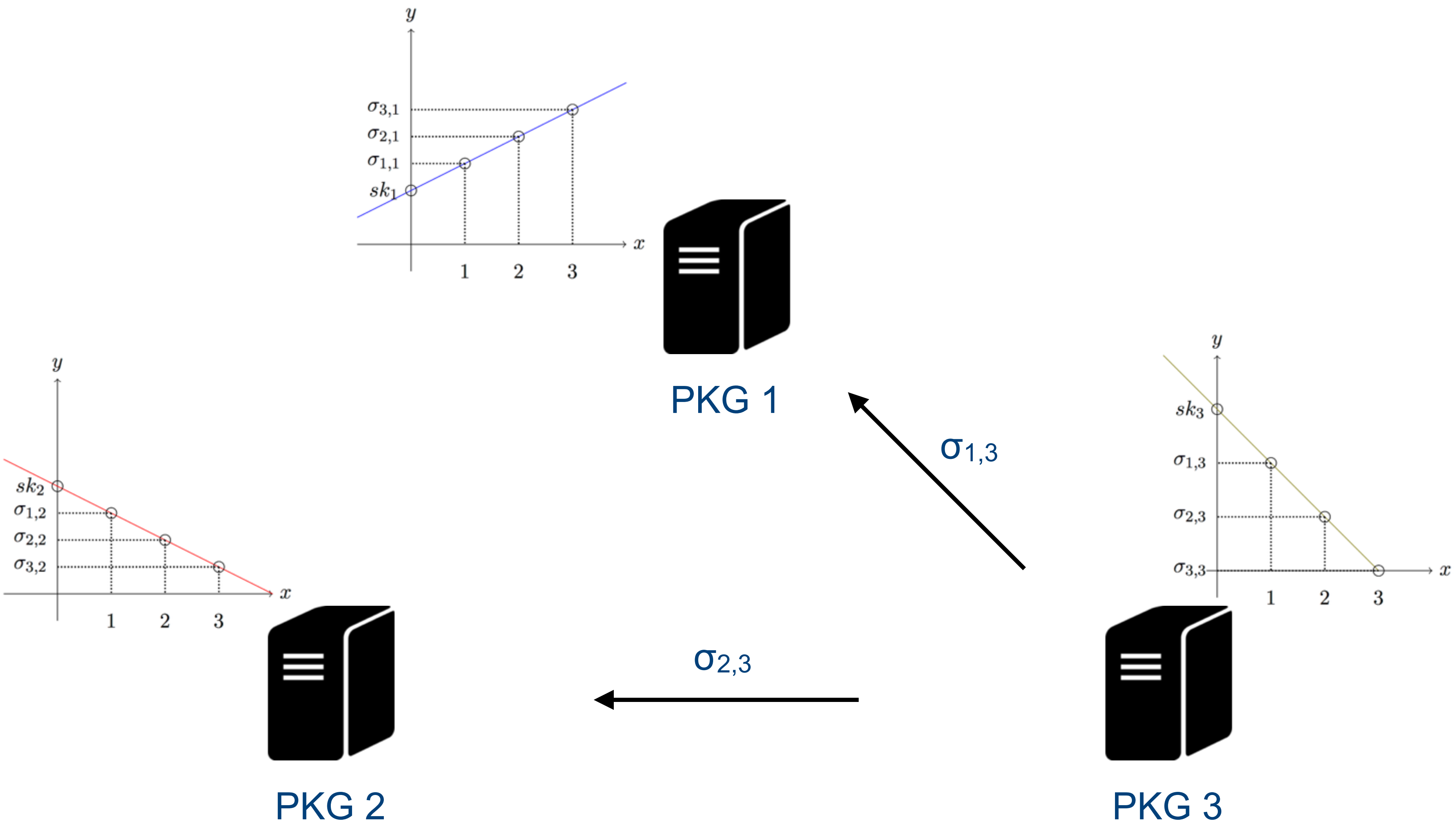

y
x
$\sigma_{3,1}$
$\sigma_{2,1}$
$\sigma_{1,1}$
sk_1
1
2
3
PKG 1
$\sigma_{2,1}$
$\sigma_{3,1}$
y
x
sk_2
PKG 2
y
x
sk_3
PKG 3

for n = 3, t = 2



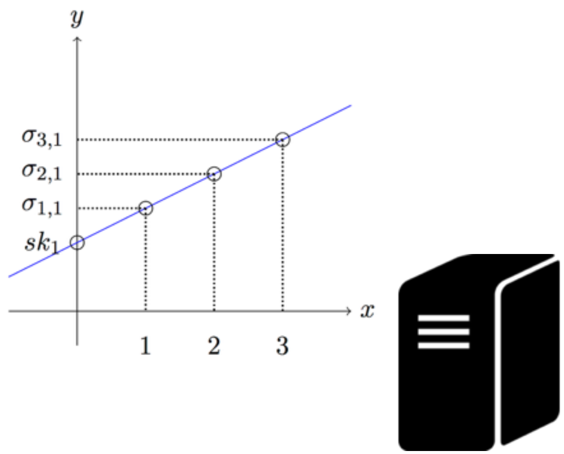
for n = 3, t = 2

KU LEUVEN

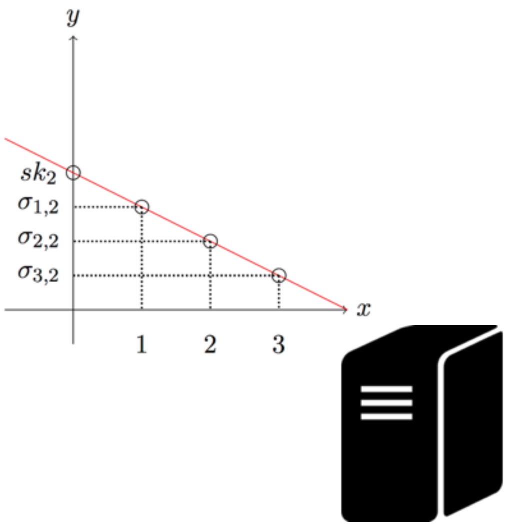


PKG 1
PKG 2
PKG 3
σ1,3
σ2,3
sk1
σ1,1
σ2,1
σ3,1
sk2
σ1,2
σ2,2
σ3,2
sk3
σ1,3
σ2,3
σ3,3
x
y
1
2
3

for n = 3, t = 2



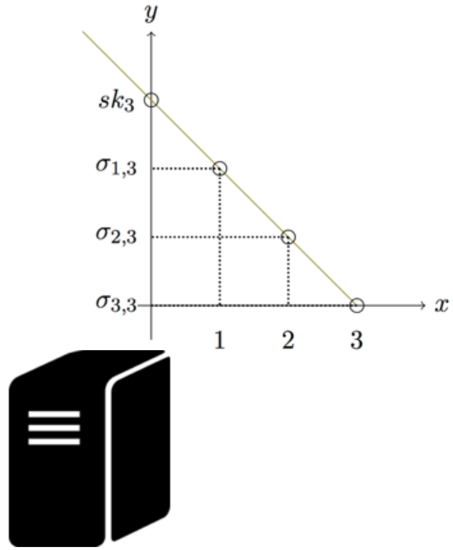
PKG 1



PKG 2

Every PKG server calculates the sum of his received shares $\sigma_{j,v}$ in $\mathbb{Z}_q$ to obtain a unique secret s_j:

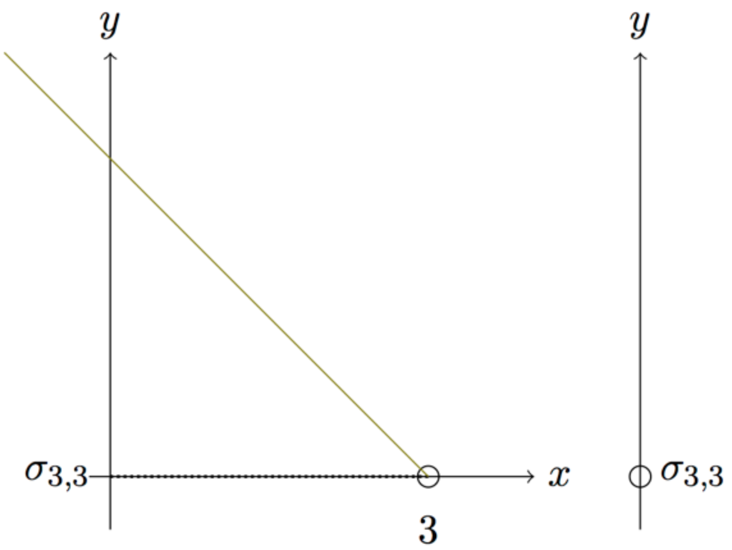
$$s_j = \sum_{v=1}^{n} \sigma_{jv}$$



PKG 3

KU LEUVEN

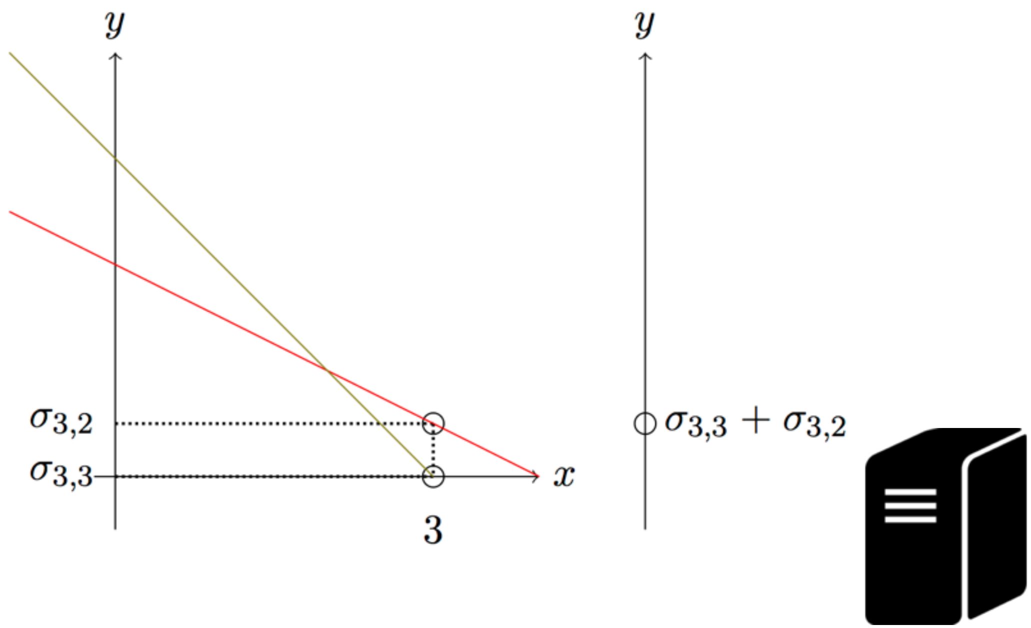
Example given for PKG 3:



PKG 3

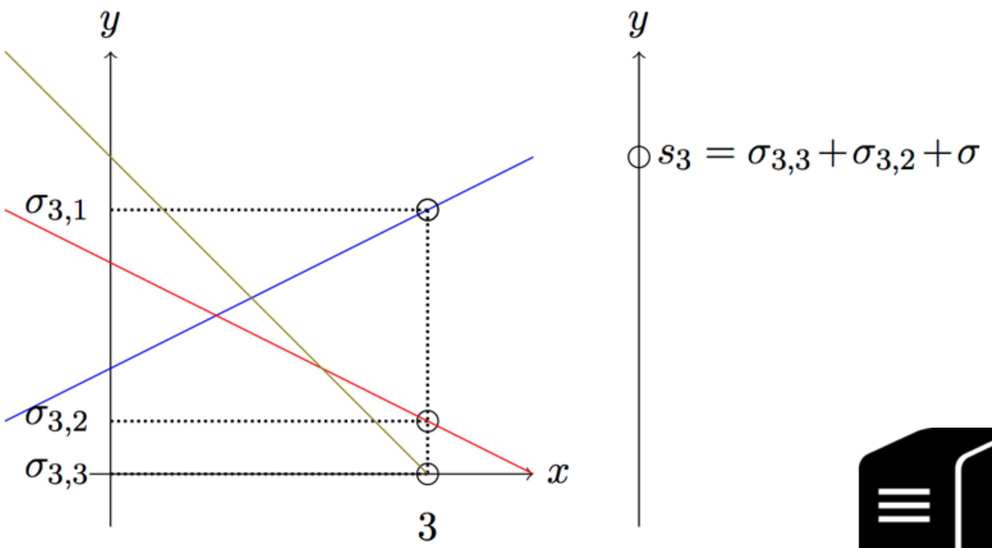
KU LEUVEN

Example given for PKG 3:



PKG 3

Example given for PKG 3:



PKG 3

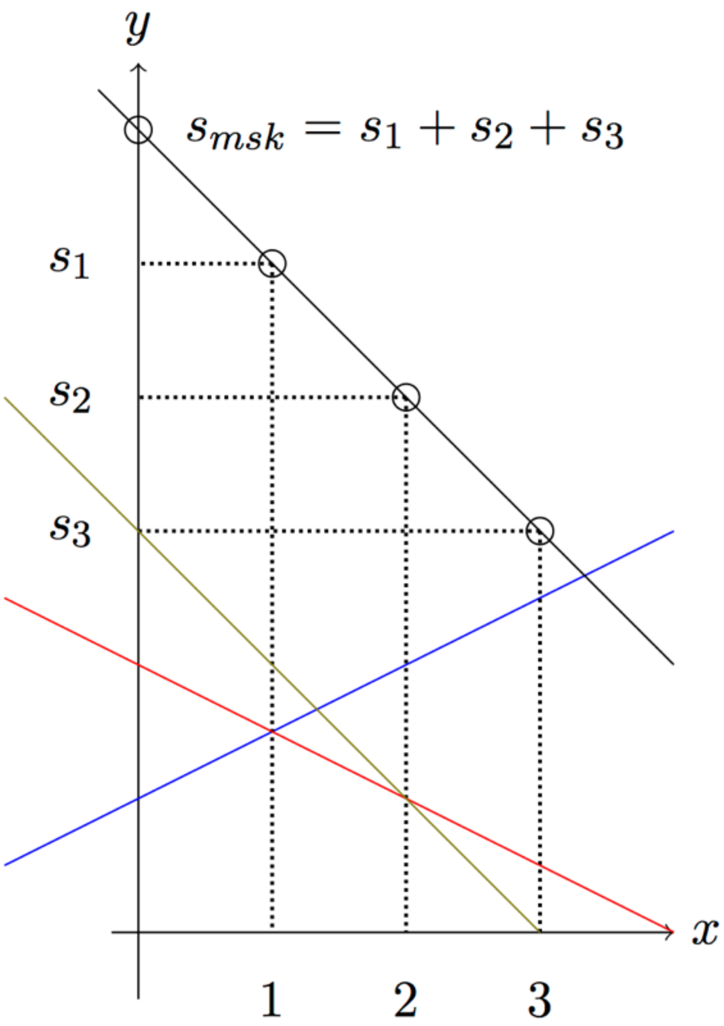
KU LEUVEN

If all PKGs compute

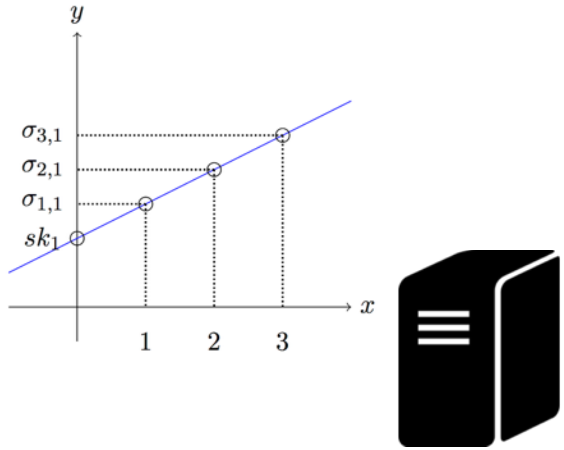
$$s_j = \sum_{v=1}^{n} \sigma_{jv}$$

s_{msk} could be found
by Lagrange interpolating
two different s_i values

PKGs are assumed to
never share their s_i values

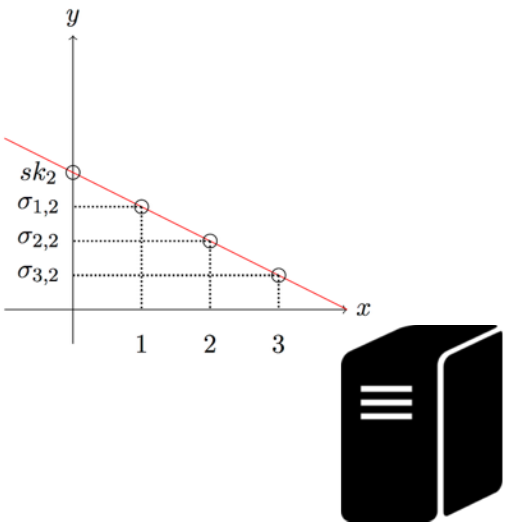


KU LEUVEN



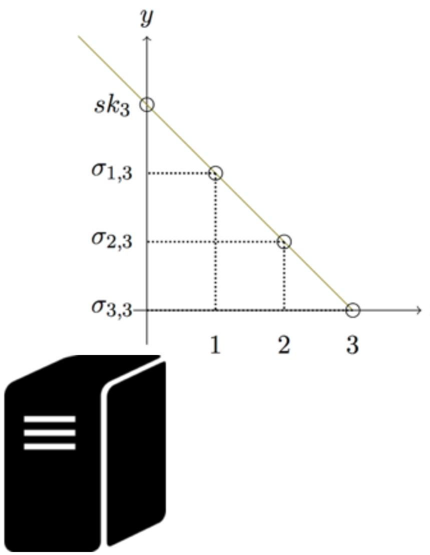
PKG 1 publishes:

- $P_{pub} = s_1.P$
- P



PKG 2 publishes:

- $P_{pub} = s_2.P$
- P



PKG 3 publishes:

- $P_{pub} = s_3.P$
- P

Setup(λ, t, n) - Remarks

- DKG algorithm is only a proof of concept:
  - Pedersen DKG is unsafe
    - Use improvement from Genaro et al.
    - Rely on asynchronous implementation from Kate et al.
  - One DKG can completely define the P value which is potentially unsafe
  - All PKGs are expected to follow the protocol honestly

KU LEUVEN

Generic Algorithms

- **Setup(**λ, t, n**):**

  Outputs the public *params* of the system with respect to the security parameter λ, the number of PKGs n and the threshold t.

- **KeyGen(***params*, $\{PKG_1, \ldots, PKG_t\}$, id_i**):**

  On input of *params*, a user identifier id_i and a subset of size t PKG servers, generates a valid private key for id_i

- **Publish(***params*, $\mathcal{S}$, m**):**

  On input of *params*, the recipient set $\mathcal{S}$ and a plaintext message m, generates a broadcast message B

- **Retrieve(***params*, sk_{id}, B**):**

  On input of *params*, the private key sk_{id} and a broadcast message B, reconstructs the plaintext message m

KeyGen(*params*, {PKG_1, … , PKG_t}, id_i)

- $P_{pub1} = s_1.P$
- P

PKG 1

- $P_{pub2} = s_2.P$
- P

PKG 2

Alice

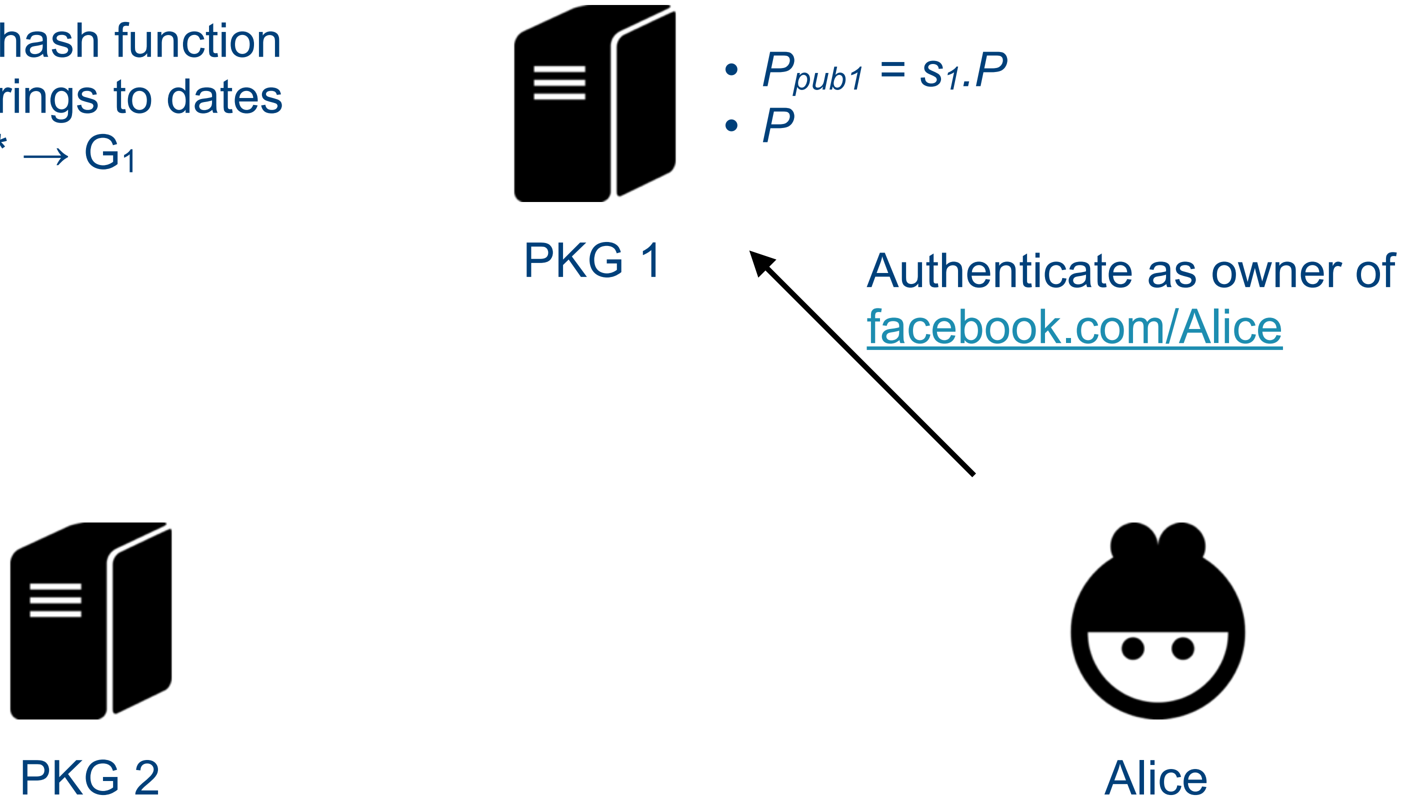
KU LEUVEN

KeyGen(*params*, {PKG_1, … , PKG_t}, id_i)

- H_{date} is a hash function mapping strings to dates
- $H_1 : \{0,1\}^* \rightarrow G_1$

- $P_{pub1} = s_1.P$
- P

- $P_{pub2} = s_2.P$
- P



KU LEUVEN

KeyGen(*params*, {PKG_1, … , PKG_t}, id_i)

- H_{date} is a hash function mapping strings to dates
- $H_1 : \{0,1\}^* \rightarrow G_1$

- $P_{pub1} = s_1.P$
- P

PKG 1

PKG 1 calculates:
1. $Q_{Alice} = H_1(Alice \,||\, H_{date}(Alice))$
2. $Q_{priv1,\,Alice} = s_1.Q_{Alice}$

- $P_{pub2} = s_2.P$
- P

PKG 2

Alice

KU LEUVEN

KeyGen(*params*, {PKG_1, … , PKG_t}, id_i)

- H_{date} is a hash function mapping strings to dates
- $H_1 : \{0,1\}^* \rightarrow G_1$

PKG 1

- $P_{pub1} = s_1.P$
- P

PKG 1 returns $Q_{priv1,\ Alice}$

PKG 1 calculates:

1. $Q_{Alice} = H_1(Alice \mid\mid H_{date}(Alice))$
2. $Q_{priv1,\ Alice} = s_1.Q_{Alice}$

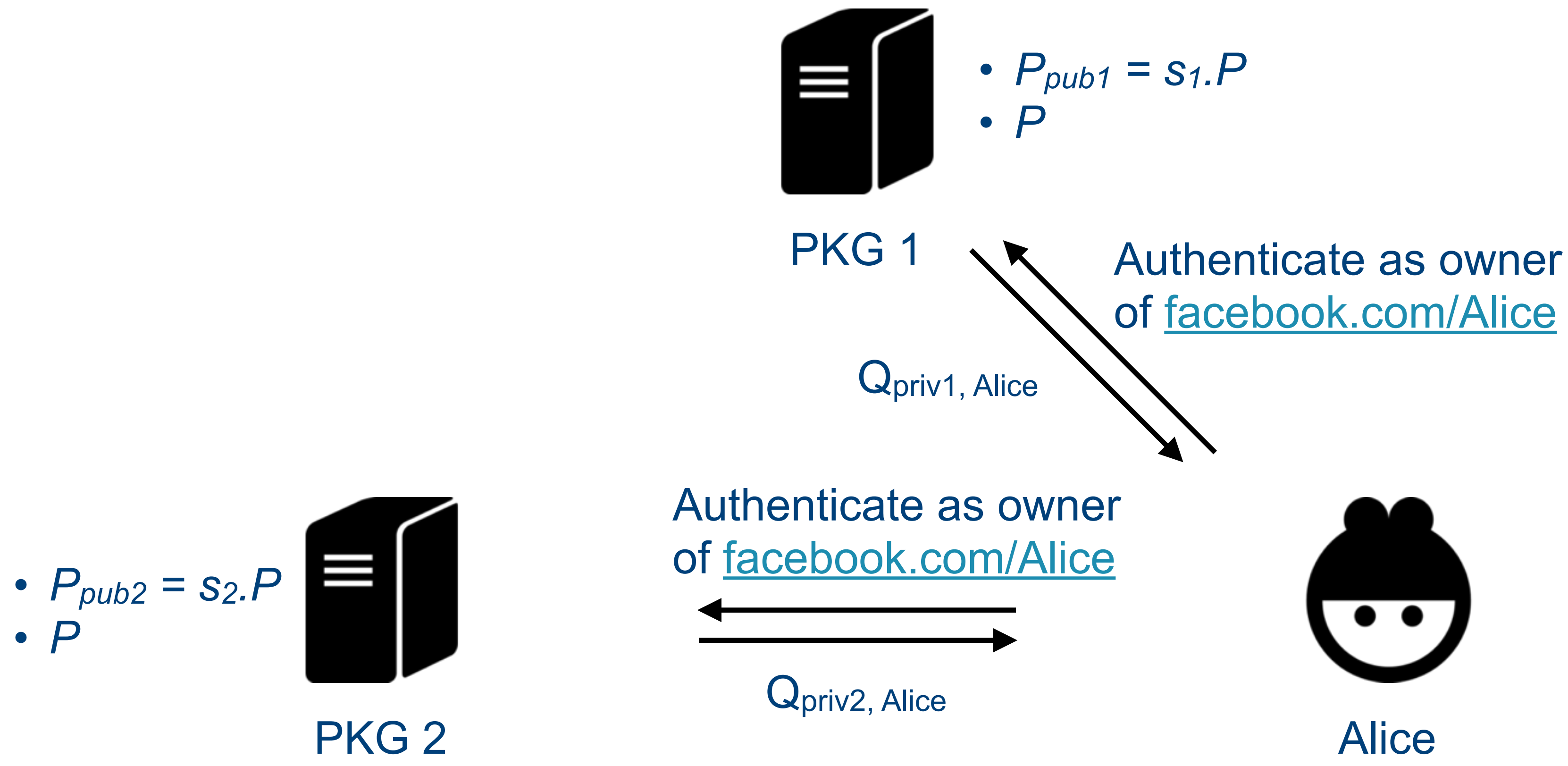
- $P_{pub2} = s_2.P$
- P

PKG 2

Alice

KU LEUVEN

KeyGen(*params*, {PKG_1, … , PKG_t}, id_i)



KeyGen(*params*, {*PKG*$_1$, … , *PKG*$_t$}, *id*$_i$)

Alice

1. Alice derives P_{pub} using Lagrange interpolation

$$P_{pub} = \sum_{j=\{1,2\}} b_j P_{pubj} \quad \text{for} \quad b_j = \prod_{z \in \{1,2\}} \frac{z}{z-j}$$

2. Alice calculates $Q_{Alice} = H_1(Alice \,||\, H_{date}(Alice))$ and verifies whether

$$e\left(Q_{privj,Alice}, P_{pub}\right) = e\left(Q_{Alice}, P_{pubj}\right)$$

3. Alice computes her private key as

$$sk_{Alice} = \sum_{j \in \{1,2\}} b_j Q_{privj,Alice} \quad \text{for} \quad b_j = \prod_{z \in \{1,2\}} \frac{z}{z-j}$$

KU LEUVEN

Generic Algorithms

- **Setup(**λ, *t*, *n***):**

  Outputs the public *params* of the system with respect to the security parameter λ, the number of PKGs n and the threshold t.

- **KeyGen(***params*, {PKG_1, … , PKG_t}, id_i**):**

  On input of *params*, a user identifier id_i and a subset of size *t* PKG servers, generates a valid private key for id_i

- **Publish(***params*, $\mathcal{S}$, *m***):**

  On input of *params*, the recipient set $\mathcal{S}$ and a plaintext message *m*, generates a broadcast message B

- **Retrieve(***params*, sk_{id}, B**):**

  On input of *params*, the private key sk_{id} and a broadcast message B, reconstructs the plaintext message *m*

KU LEUVEN

Publish(*params*, $\mathcal{S}$, *m*)

Alice

1. Alice derives P_{pub} using Lagrange interpolation

$$P_{pub} = \sum_{j=\{1,2\}} b_j P_{pubj} \quad \text{for} \quad b_j = \prod_{z\in\{1,2\}} \frac{z}{z-j}$$

2. Alice calculates $Q_{Alice} = H_1(Alice \,||\, H_{date}(Alice))$ and verifies whether

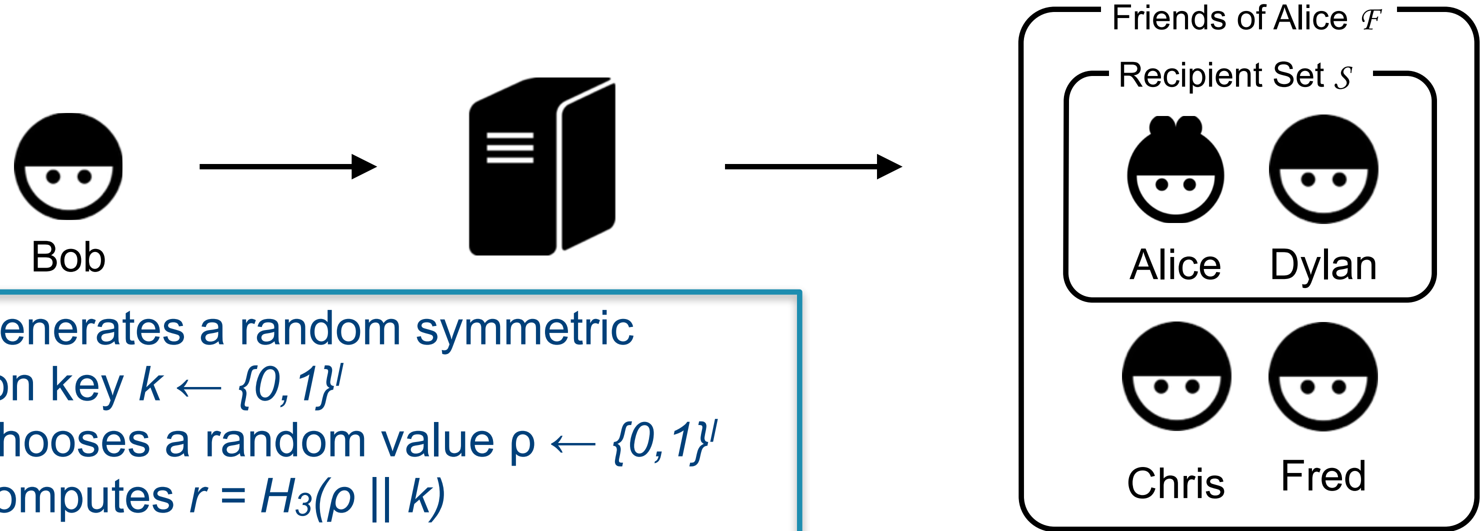
$$e\left(Q_{privj,Alice}, P_{pub}\right) = e\left(Q_{Alice}, P_{pubj}\right)$$

3. Alice computes her private key as

$$sk_{Alice} = \sum_{j\in\{1,2\}} b_j Q_{privj,Alice} \quad \text{for} \quad b_j = \prod_{z\in\{1,2\}} \frac{z}{z-j}$$

KU LEUVEN

Publish(*params*, $\mathcal{S}$, *m*)



1. Bob generates a random symmetric session key $k \leftarrow \{0,1\}^l$
2. Bob chooses a random value $\rho \leftarrow \{0,1\}^l$ and computes $r = H_3(\rho \,||\, k)$
3. For each recipient id_i in $\mathcal{S}$, Bob computes

$$w_i = \rho \oplus H_2\left(g^r_{\mathtt{id}_i}\right) \quad \text{where} \quad g_{\mathtt{id}_i} = e\left(Q_{\mathtt{id}_i}, P_{pub}\right) \in G_T$$

4. Compose the authenticated data as

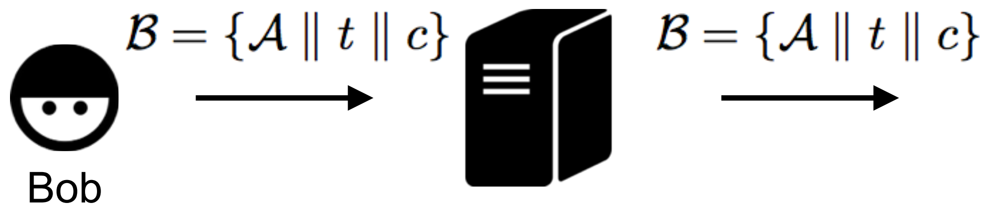
$$\begin{aligned} \mathcal{A} &= \{\eta \,\|\, \boxed{rP \,\|\, k \oplus H_3(\rho) \,\|\, w_1} \,\|\, w_2 \,\|\, \ldots \,\|\, w_\eta\} \\ &= \{\eta \,\|\, U \,\|\, v \,\|\, w\} \quad \text{for} \quad w = \{w_1 \,\|\, w_2 \,\|\, \ldots \,\|\, w_\eta\} \end{aligned}$$

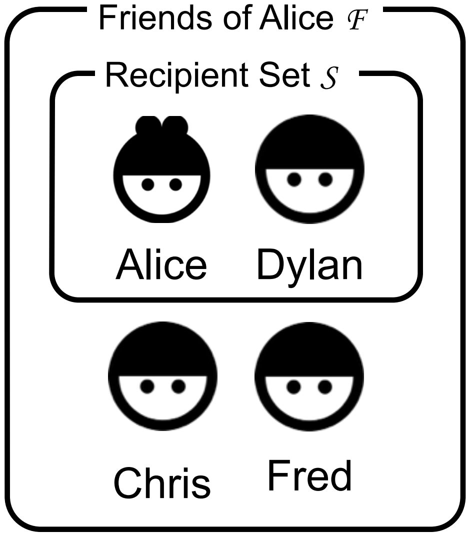
5. Bob concatenates the plaintext message *m* to the intended set of recipients $\mathcal{S}$, such that $M = \{m \,||\, \mathcal{S}\}$

Same as CCA Secure Franklin and Boneh IBE Encryption of symmetric key *k*

KU LEUVEN

Publish(*params*, $\mathcal{S}$, *m*)





5. Alice concatenates the plaintext message *m* to the intended set of recipients $\mathcal{S}$, such that $M = \{m \parallel \mathcal{S}\}$
6. Bob applies authenticated symmetric encryption

$$\langle c, t\rangle \leftarrow \mathrm{E}_k(\mathcal{M}, \mathcal{A})$$

7. Bob broadcasts the concatenation

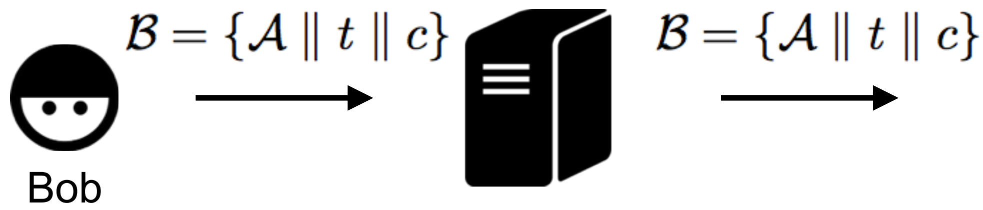
$$\mathcal{B} = \{\mathcal{A} \parallel t \parallel c\}$$

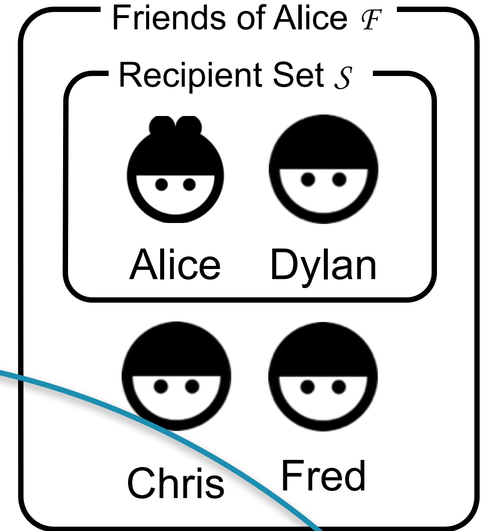
KU LEUVEN

Generic Algorithms

- **Setup(**λ, t, n**):**
  Outputs the public *params* of the system with respect to the security parameter λ, the number of PKGs n and the threshold t.
- **KeyGen(***params*, $\{PKG_1, \ldots, PKG_t\}$, id_i**):**
  On input of *params*, a user identifier id_i and a subset of size t PKG servers, generates a valid private key for id_i
- **Publish(***params*, $\mathcal{S}$, m**):**
  On input of *params*, the recipient set $\mathcal{S}$ and a plaintext message m, generates a broadcast message B
- **Retrieve(***params*, sk_{id}, B**):**
  On input of *params*, the private key sk_{id} and a broadcast message B, reconstructs the plaintext message m

Retrieve(*params*, sk_{id}, B)





1. Alice receives

$$\mathcal{B} = \{\mathcal{A} \parallel t \parallel c\}$$

$$\begin{aligned} \mathcal{A} &= \{\eta \parallel \boxed{rP \parallel k \oplus H_3(\rho) \parallel w_1} \parallel w_2 \parallel \ldots \parallel w_\eta\} \\ &= \{\eta \parallel \boxed{U \parallel v \parallel w}\} \quad \text{for} \quad w = \{w_1 \parallel w_2 \parallel \ldots \parallel w_\eta\} \end{aligned}$$

2. Alice computes

$$\rho = w_i \oplus H_2(e(sk_{Alice}, U)) \qquad k = v \oplus H_3(\rho) \qquad r = H_3(\rho \parallel k)$$

3. Verify whether $U = r.P$. If the check fails try next w_i and return to 1. Return $\perp$ if no w_i left.
4. Alice applies authenticated decryption $\langle \mathcal{M}, t' \rangle \leftarrow \mathtt{D}_k(c, \mathcal{A})$
5. Alice verifies whether $t = t'$ and returns *m*. Otherwise $\perp$.

Same as CCA Secure Franklin and Boneh IBE Decryption of symmetric key *k*

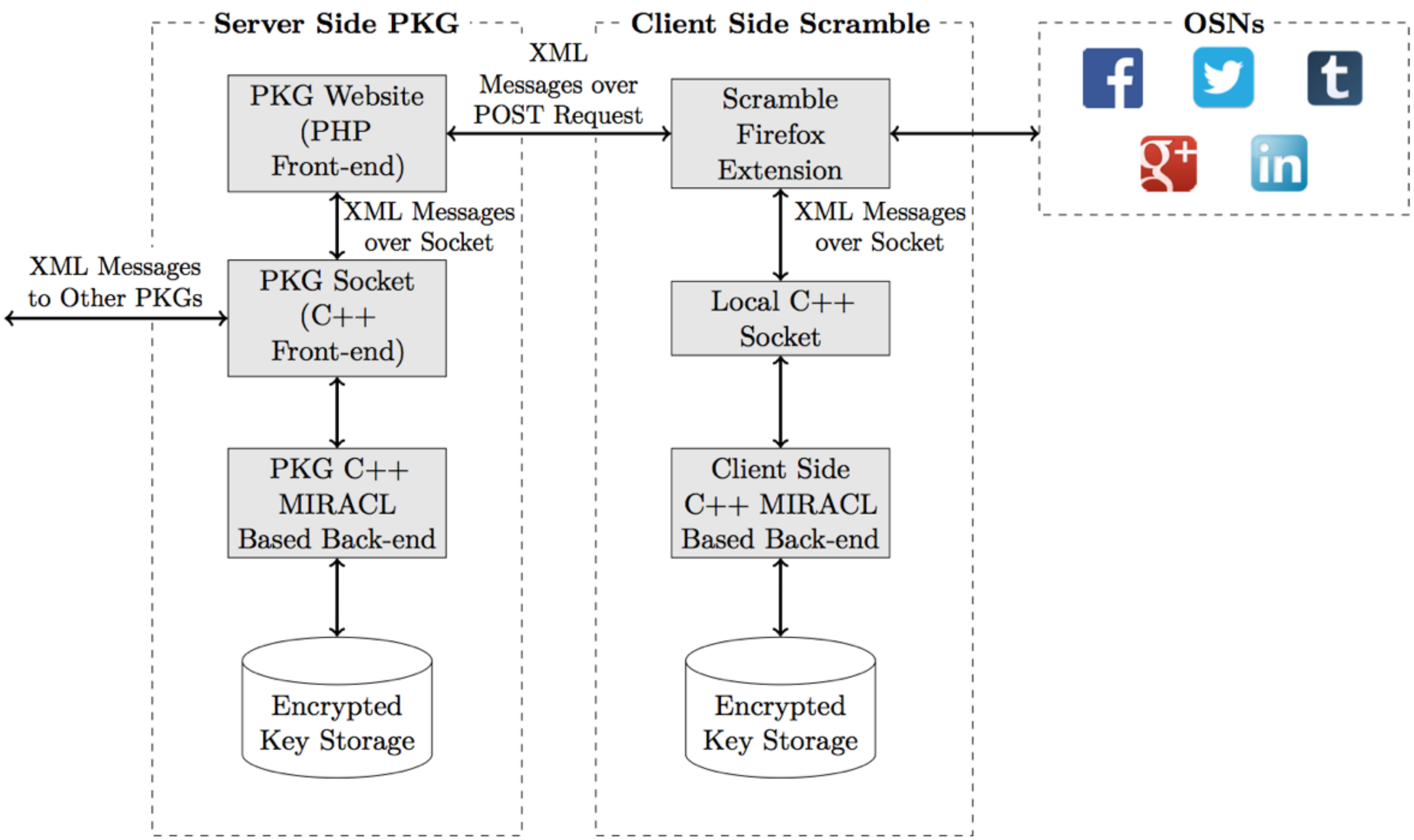
KU LEUVEN

Implementation

How the proposal is implemented

KU LEUVEN

Architectural Overview



Publish and Retrieve - Execution time

	Execution Time (ms)	
Number of Recipients	`Publish`	`Retrieve`
1	284.5	275.4
10	2564.5	460.9
15	3799.6	560.6
50	12300.5	1237.8
100	25867.7	2260.2

Performance of Publish and Retrieve algorithm in function of the total number of recipients

Conclusion

Conclusion

- Both the design goals and security goals are achieved
- IBE fits the needs for confidentiality in OSNs with an acceptable overhead
- Most important improvements to existing approaches:
  - Trust in a public key immediately follows from the content of the corresponding Facebook profile
  - Users can start receiving messages as soon as they are subscribed to the OSN (standard opt-in)

Future work

- A more formal security proof of the proposed scheme
- Authentication with the PKG servers is not implemented
- Support in Scramble for more OSNs than Facebook
- Explore the possibilities of randomness reuse
- PKGs should use SSL with asynchronous communication

Live Demo