Thoth: Comprehensive Policy Compliance in Data Retrieval Systems

Eslam Elnikety, Aastha Mehta, Anjo Vahldiek-Oberwagner,
Deepak Garg, and Peter Druschel

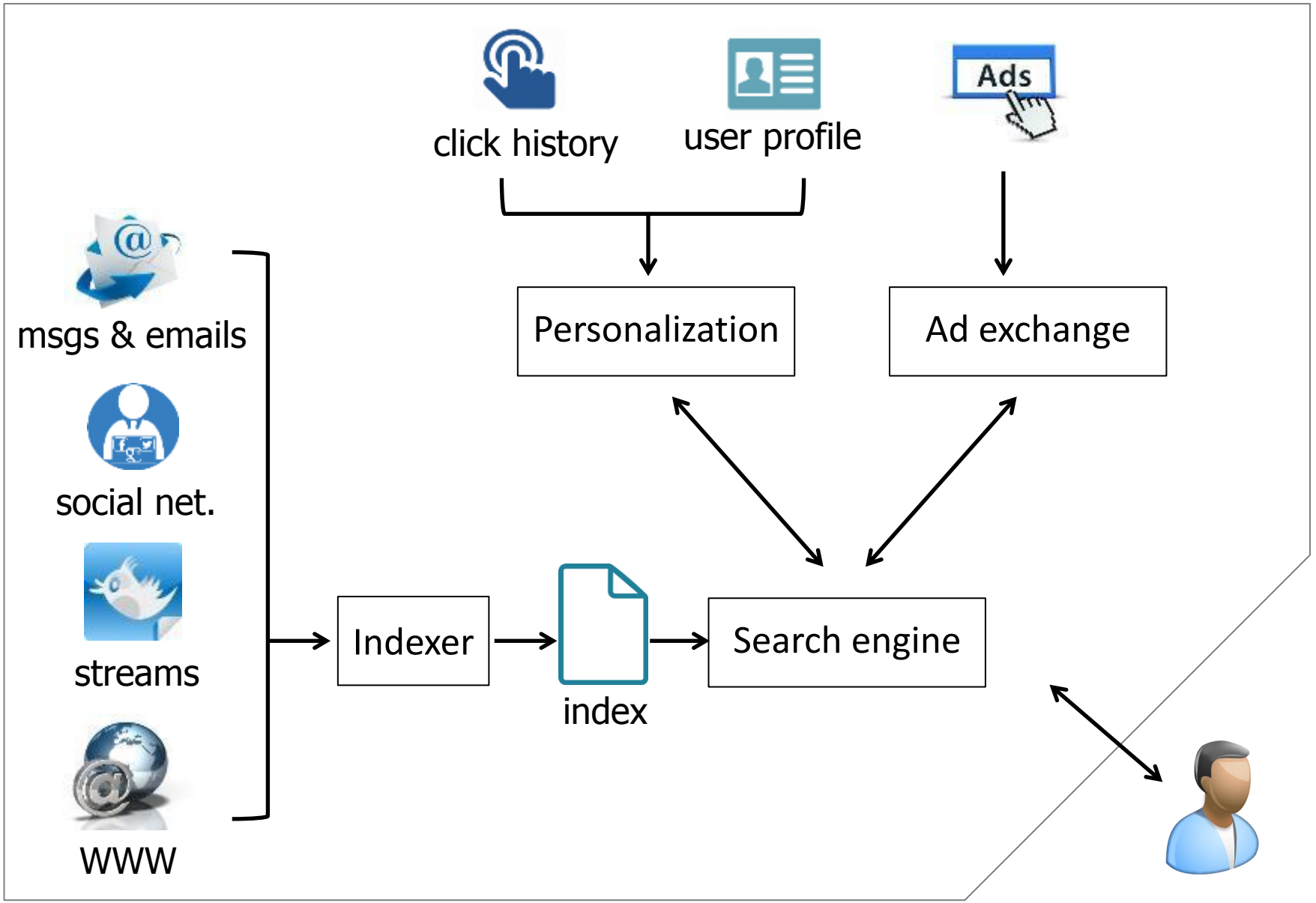
Max Planck Institute for Software Systems

Data retrieval systems

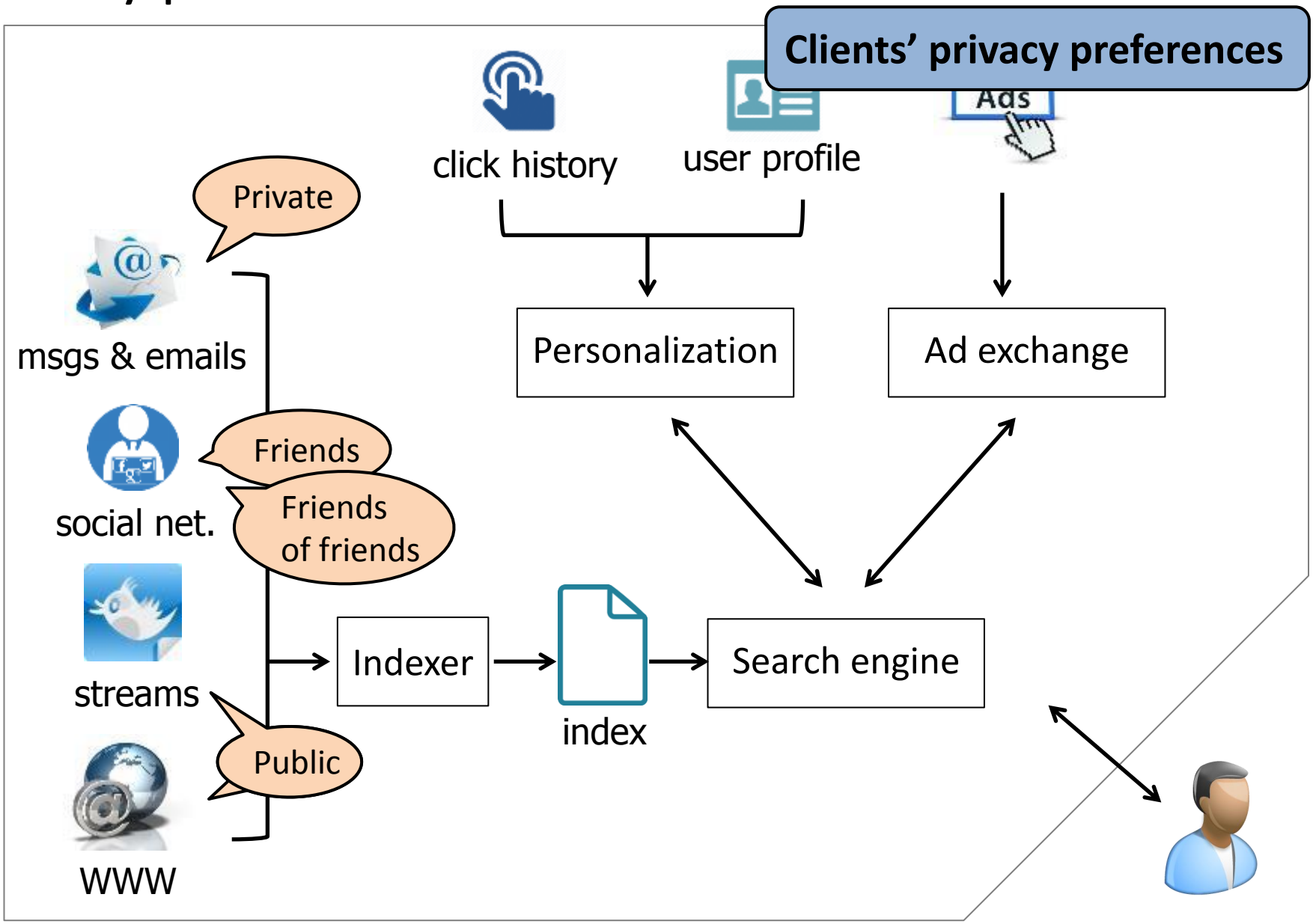
collect, process, and serve data

- Searching, browsing, advertising
- Social networking
- Blogging, publishing, news

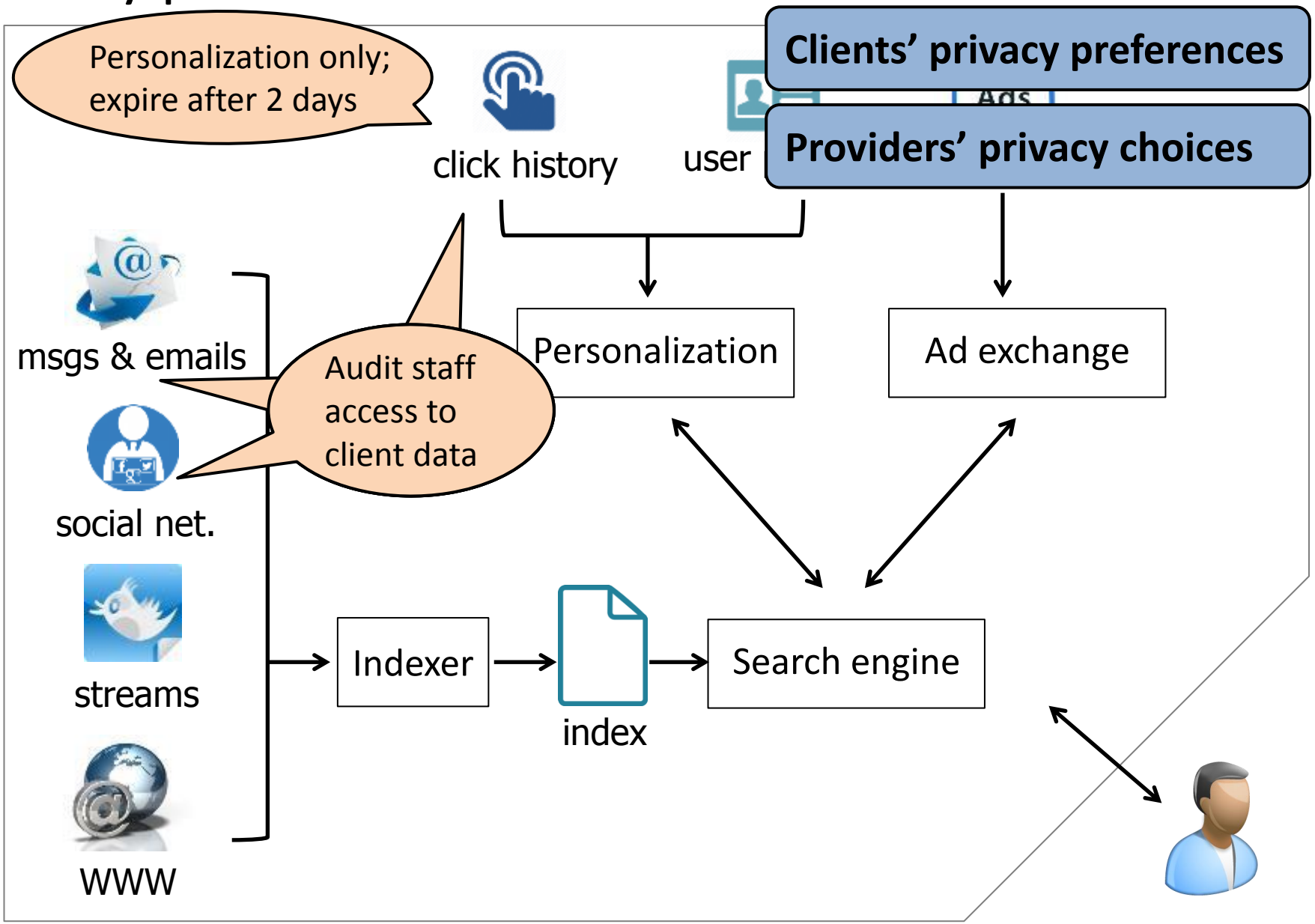
Data retrieval systems: many data sources



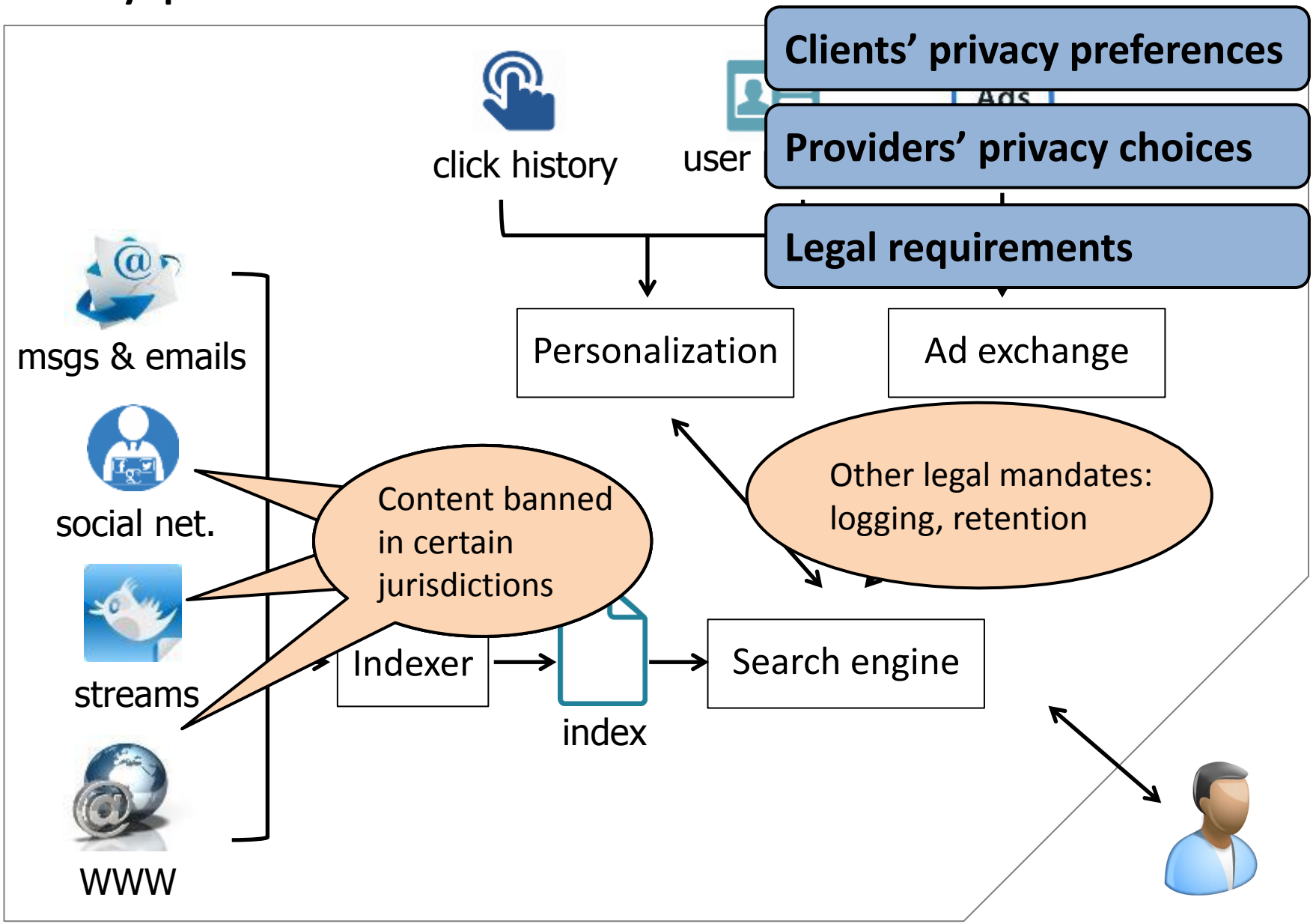
Many policies



Many policies



Many policies



Policy compliance challenges

- Many data items and complex policies
- Policy implicit in configuration files, code
- Enforcement spread over many components, layers
- Complex, fast evolving applications

Goal: Prevent *inadvertent* policy violations due to application bugs and misconfigurations

Outline

Motivation

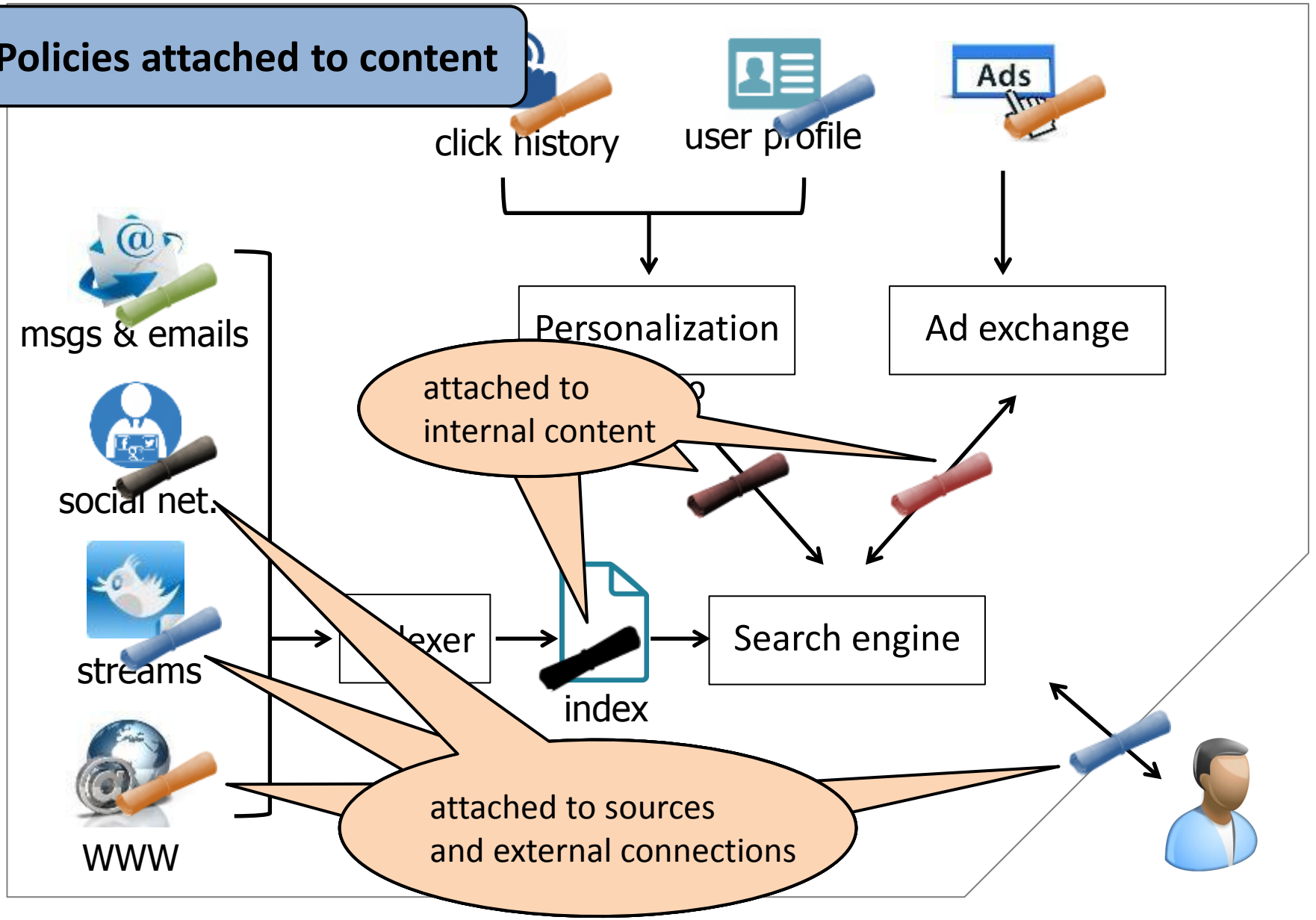
- Data retrieval systems, policies
- Challenges and goals

Thoth: policy compliance layer

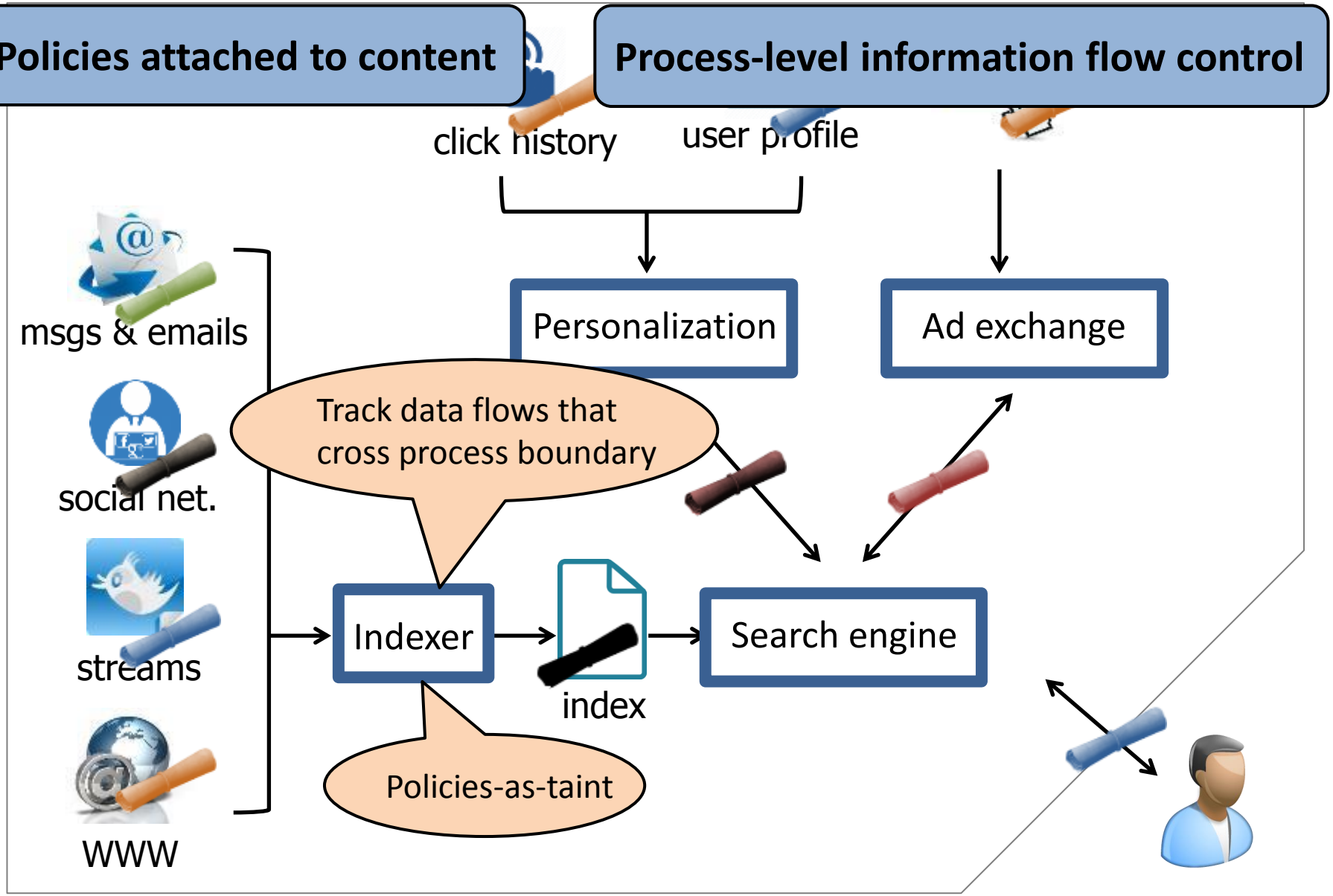
- Overview, threat model
- Policy language and examples
- Typed declassification
- Runtime enforcement

Prototype and evaluation

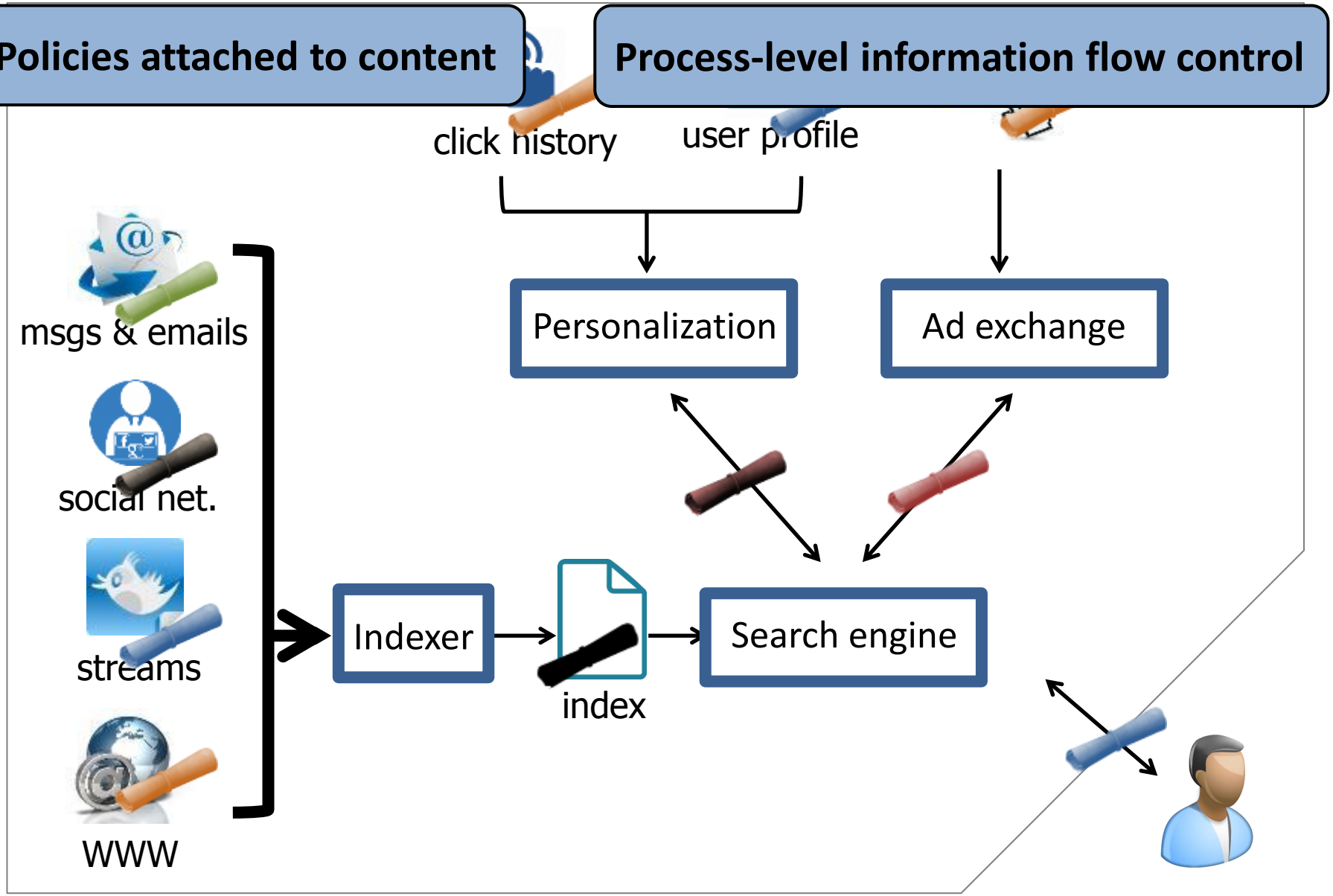
Thoth: overview



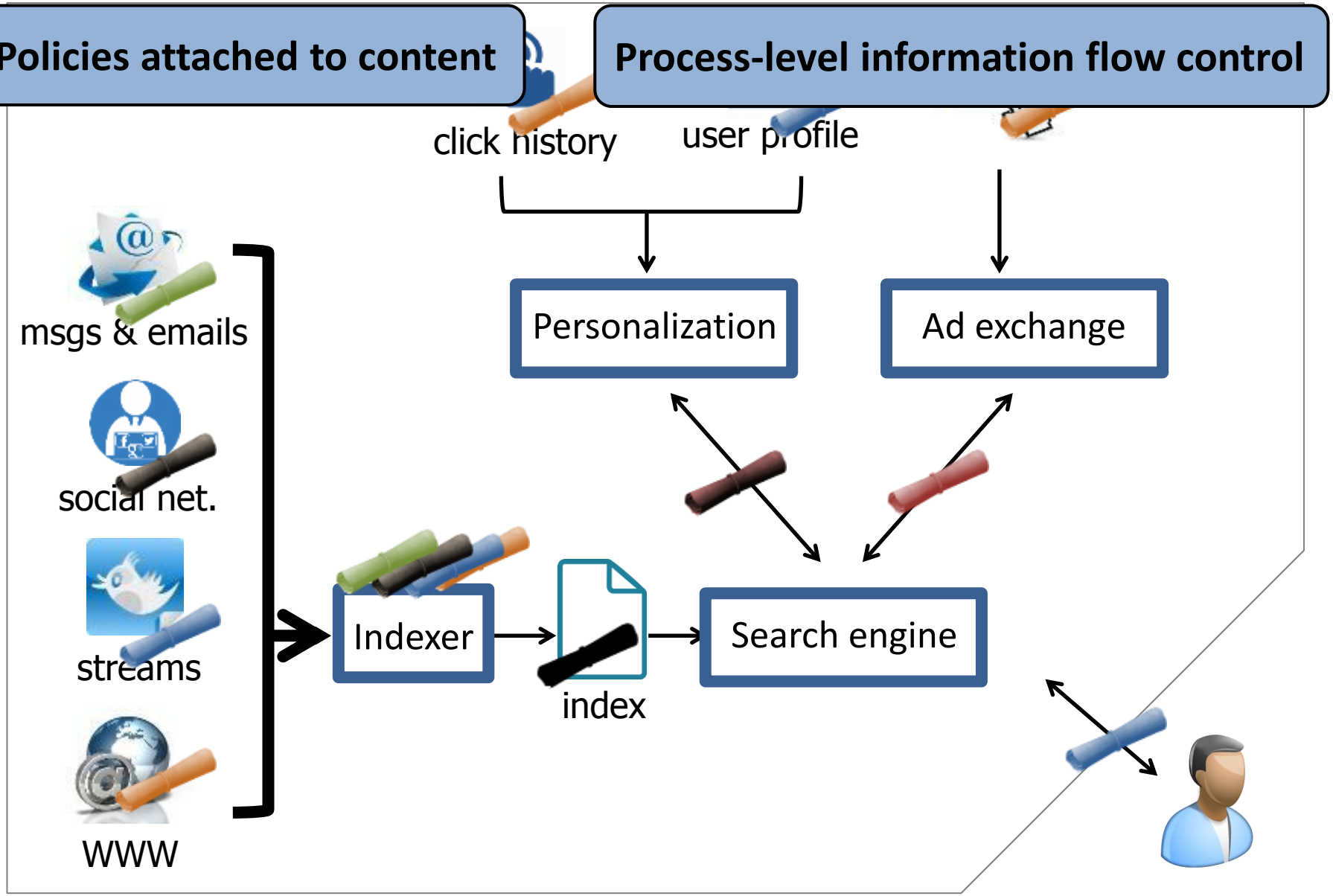
Thoth: overview



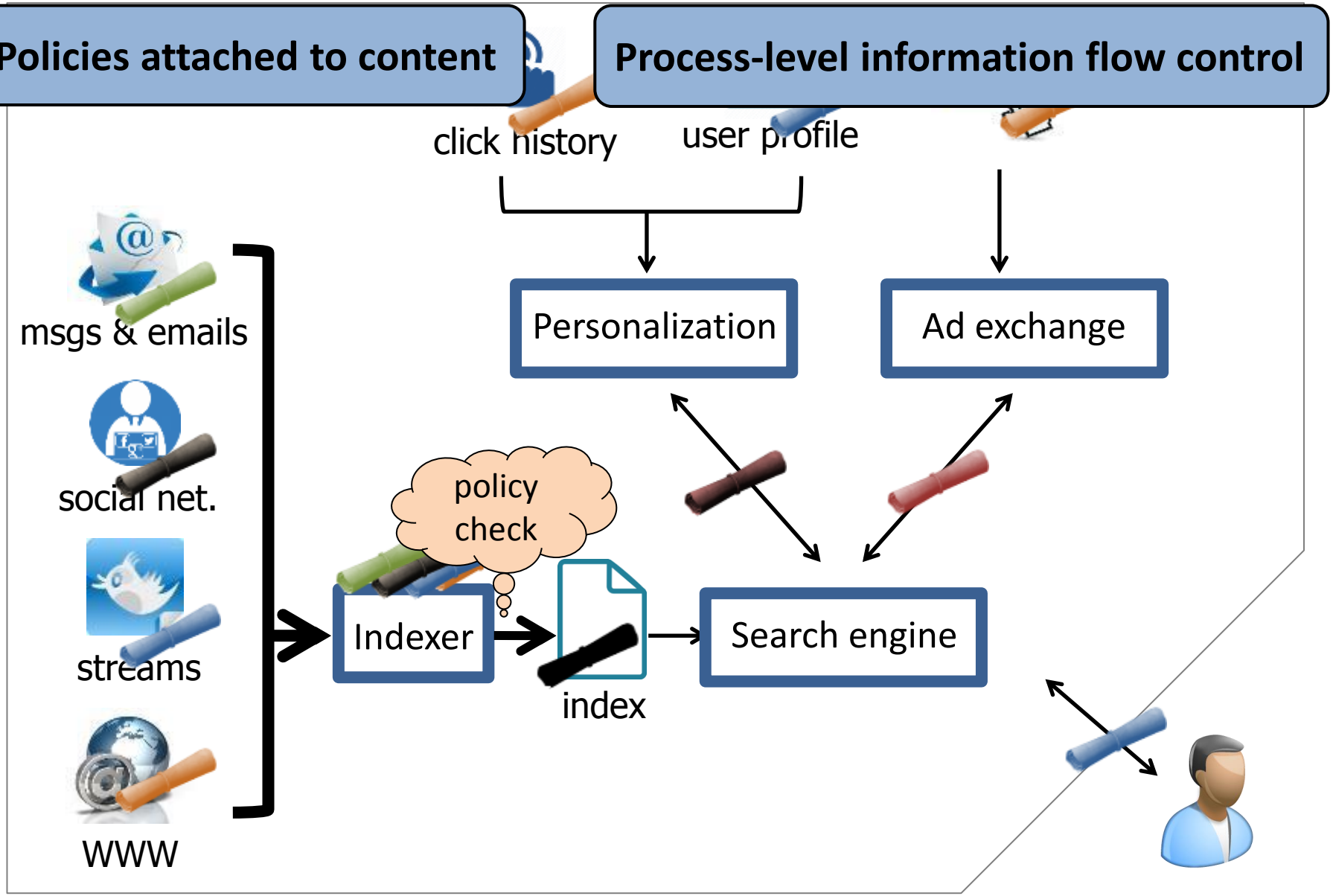
Thoth: overview



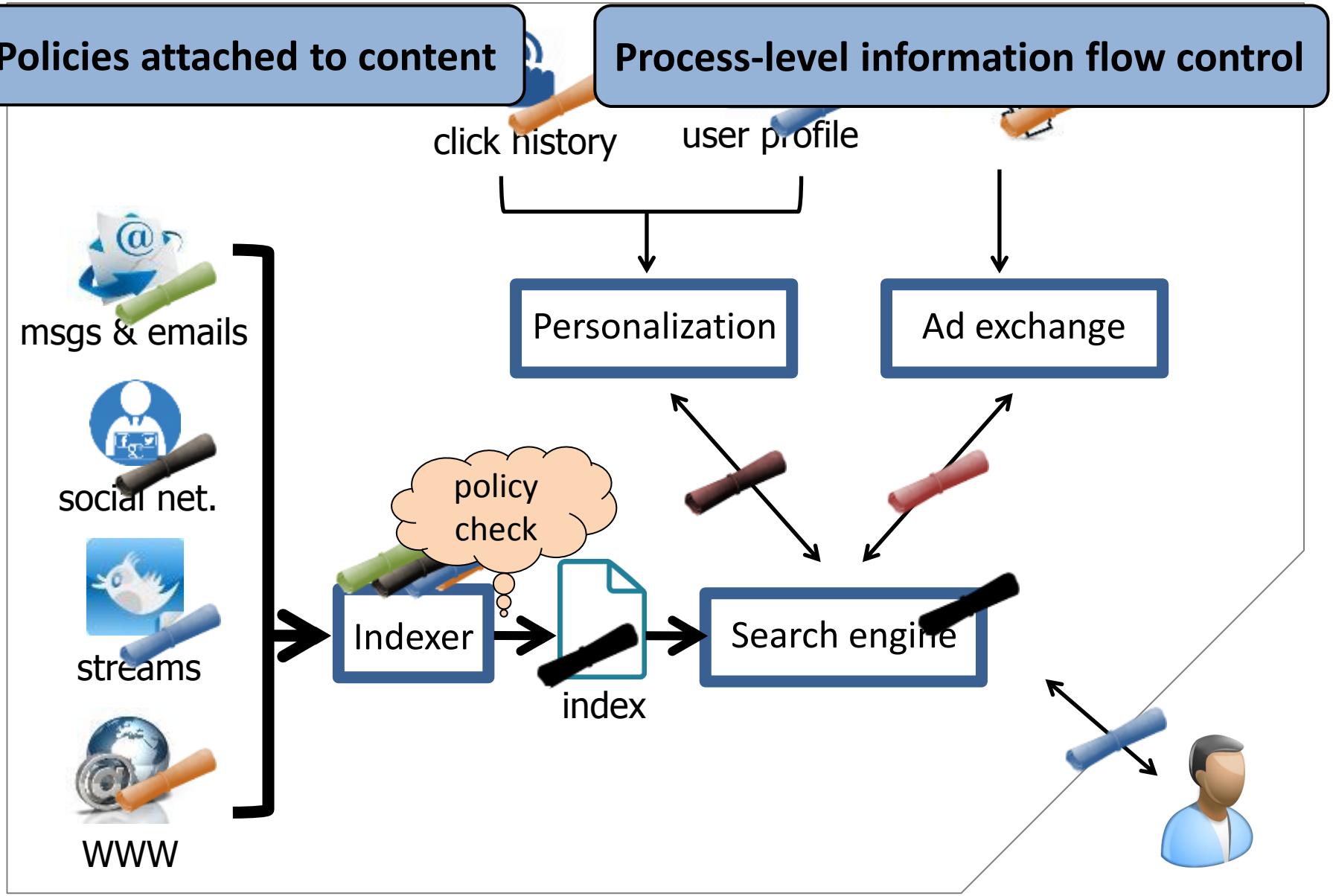
Thoth: overview



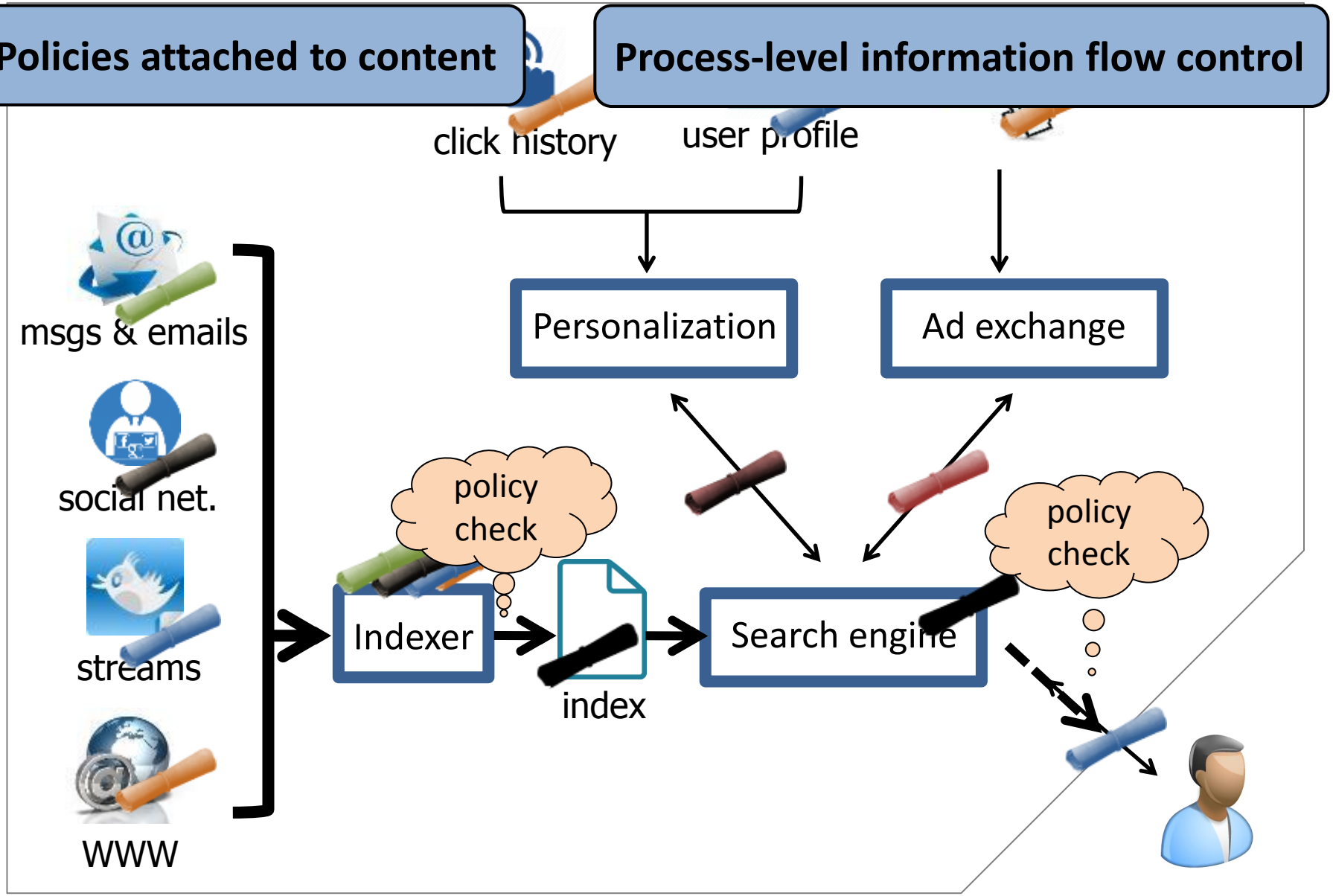
Thoth: overview



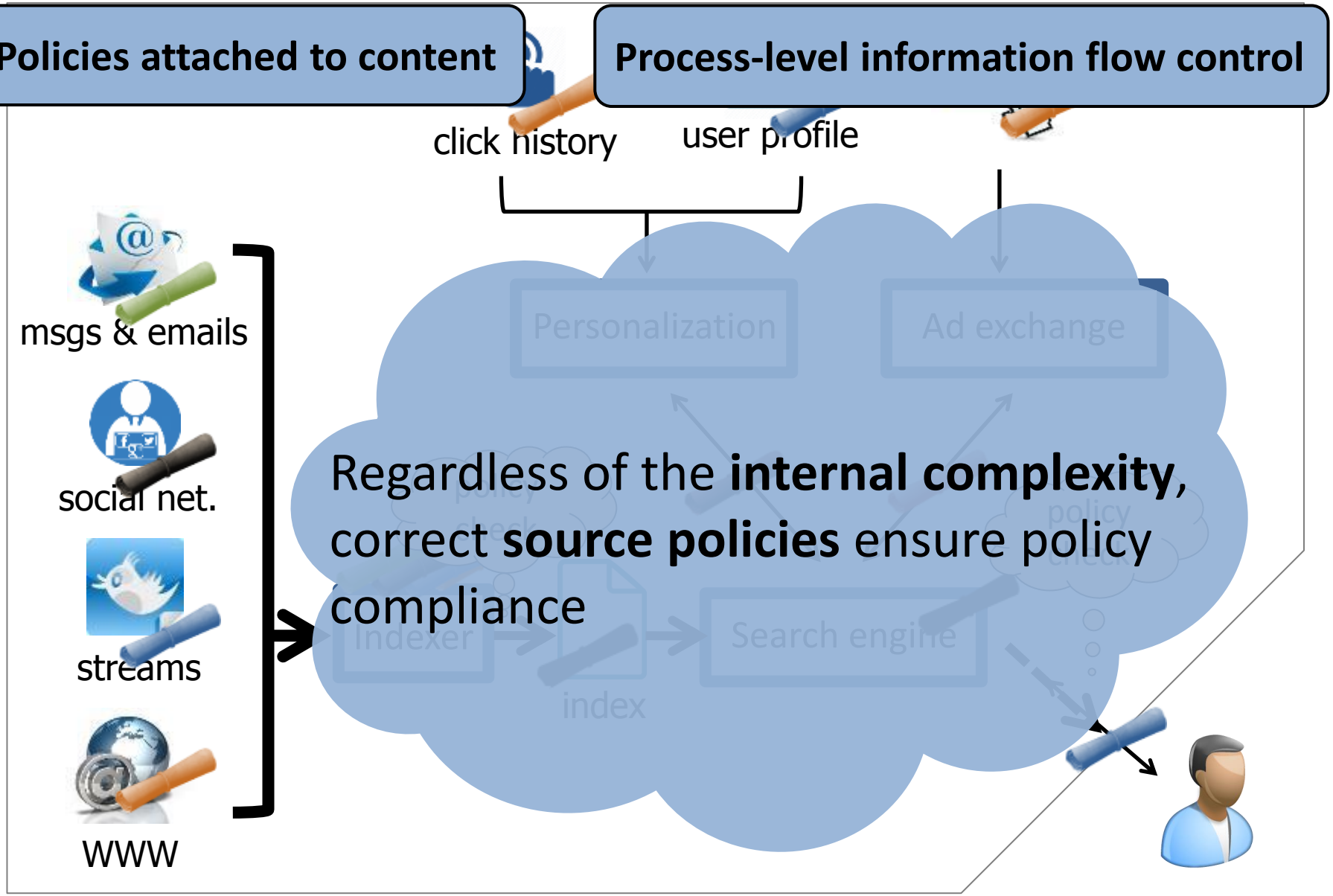
Thoth: overview



Thoth: overview



Thoth: overview



Thoth: threat model

- Thoth, OS, storage are trusted (but not applications)
- Covert and side channels are not a concern

Pragmatic:

- Provider is interested in policy compliance
- Goal is preventing inadvertent policy violations

Guarantees:

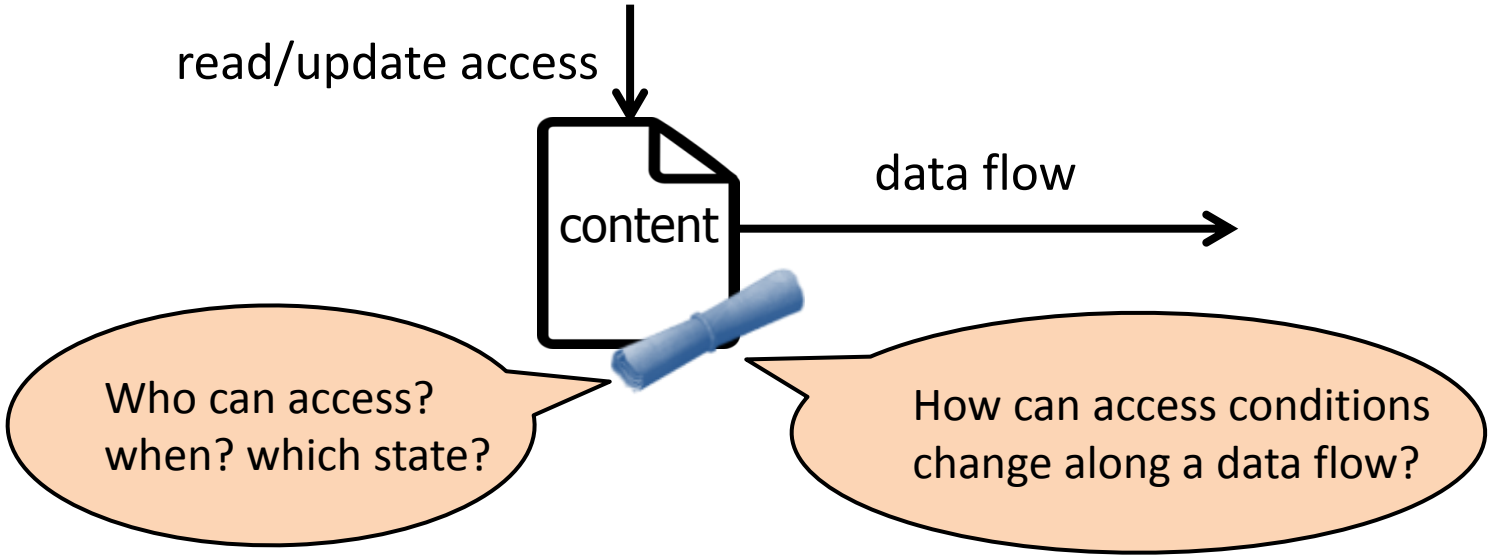
Application bugs and misconfigurations will not violate source policies

Thoth policy language overview

Declarative *data flow* policy language

Thoth policies

- can express confidentiality, integrity, declassification
- are attached to source content



Example: client access control

Alice's post accessible by herself, friends, friends of friends

read :-

sessionIs(Alice) /* Alice */

/* Alice's friends:
x can access when in Alice's friend list */

OR

sessionIs(x) **AND** "friend_list$_{Alice}$, offset$_x$" says x

/* Alice's friends of friends:
y can access when in x's friend list,
and x is in Alice's friend list */

OR

sessionIs(y) **AND** "friend_list$_x$, offset$_y$" says y **AND** "friend_list$_{Alice}$, offset$_y$" says x

"Only authorized users, as specified by Alice, can access the post."

Example: client access control

Alice's post accessible by herself, friends, friends of friends

read :-

sessionIs(Alice) /* Alice */

/* Alice's friends:
x can access when in Alice's friend list */

OR

sessionIs(x) **AND** "friend_list$_{Alice}$, offset$_x$" says x

/* Alice's friends of friends:
y can access when in x's friend list,
and x is in Alice's friend list */

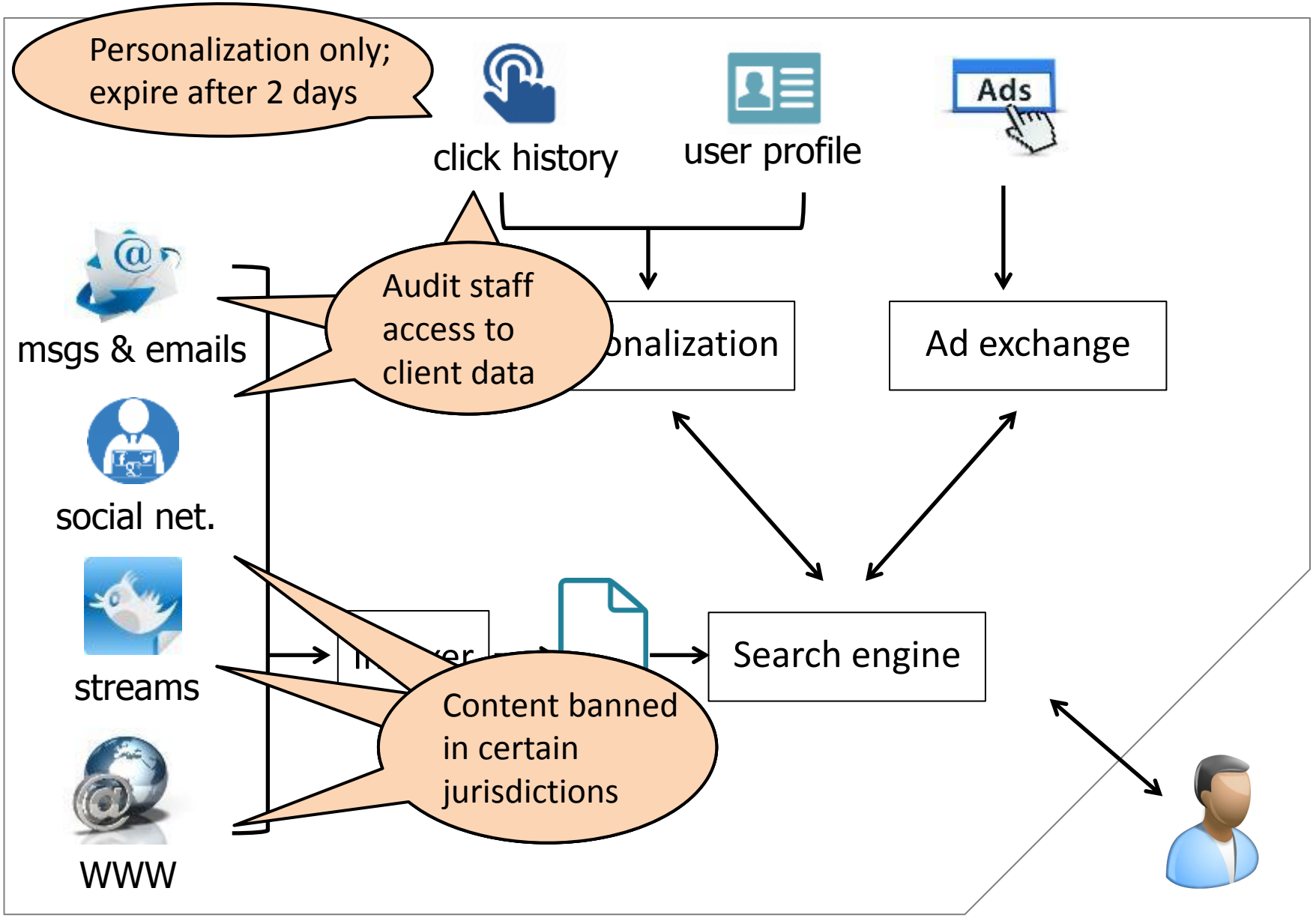
Applications are responsible for finding the entries in the friend lists

OR

sessionIs(y) **AND** "friend_list$_x$, offset$_y$" says y **AND** "friend_list$_{Alice}$, offset$_y$" says x

"Only authorized users, as specified by Alice, can access the post."

More example policies



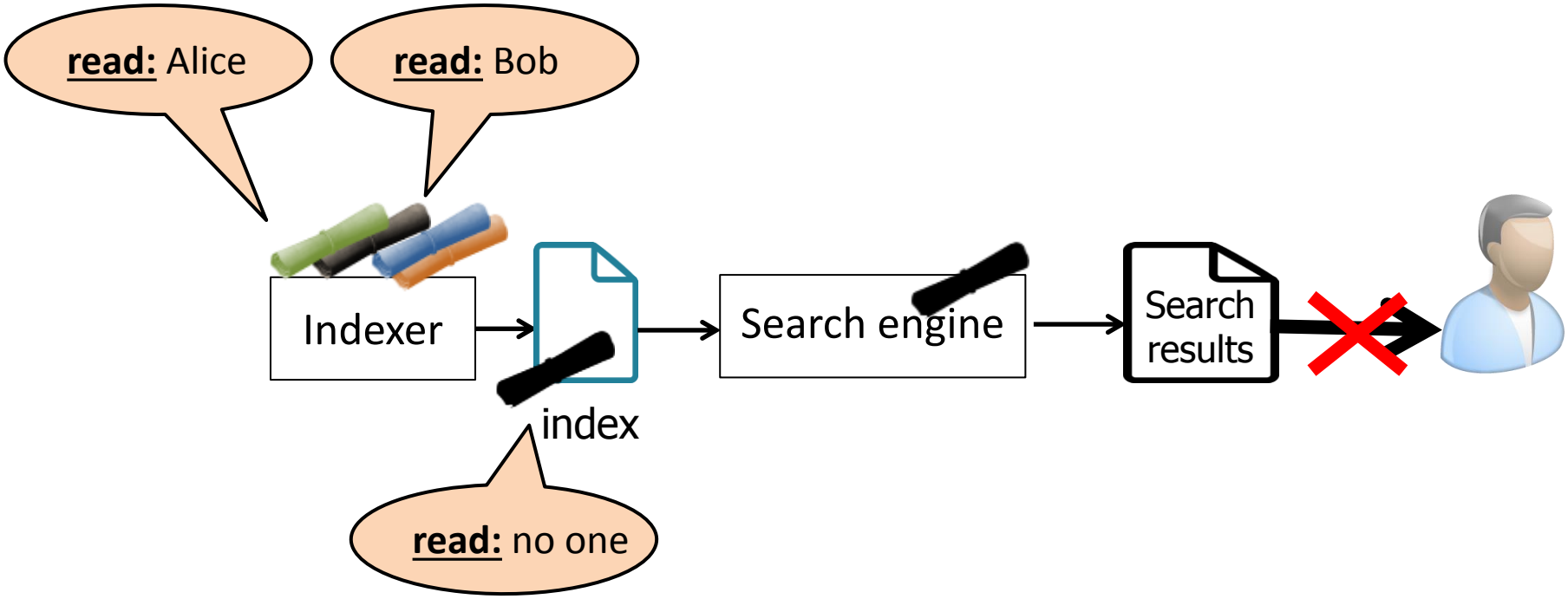
Thoth: Key ideas

- **Process-level** information flow control
  - Language independent, efficient
  - Good match for distributed computing frameworks
- **Policies-as-taint**
  - Enables taint reduction via policy comparison, partial evaluation
- **Policy-specified** declassification
  - No trust in application code for declassification
- **Typed declassification**

Typed declassification

Allows declassification for data of a specific type

Example: *Declassifying search results*



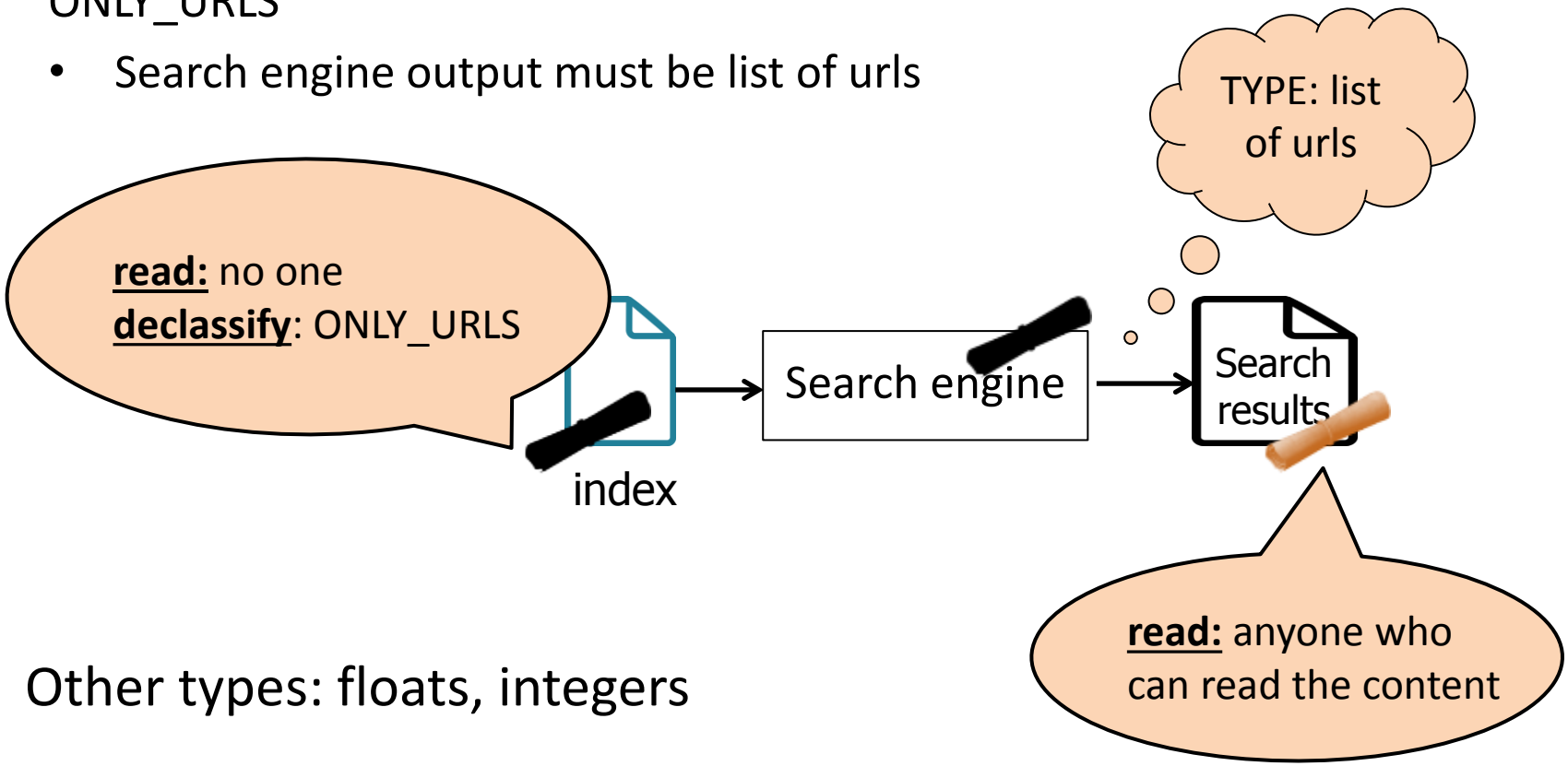
Typed declassification

Allows declassification for data of a specific type

Example: *Declassifying search results*

ONLY_URLS

- Search engine output must be list of urls



Other types: floats, integers

End-to-end policy enforcement

Algorithm: Process *p* performs I/O on content with policy *pol*

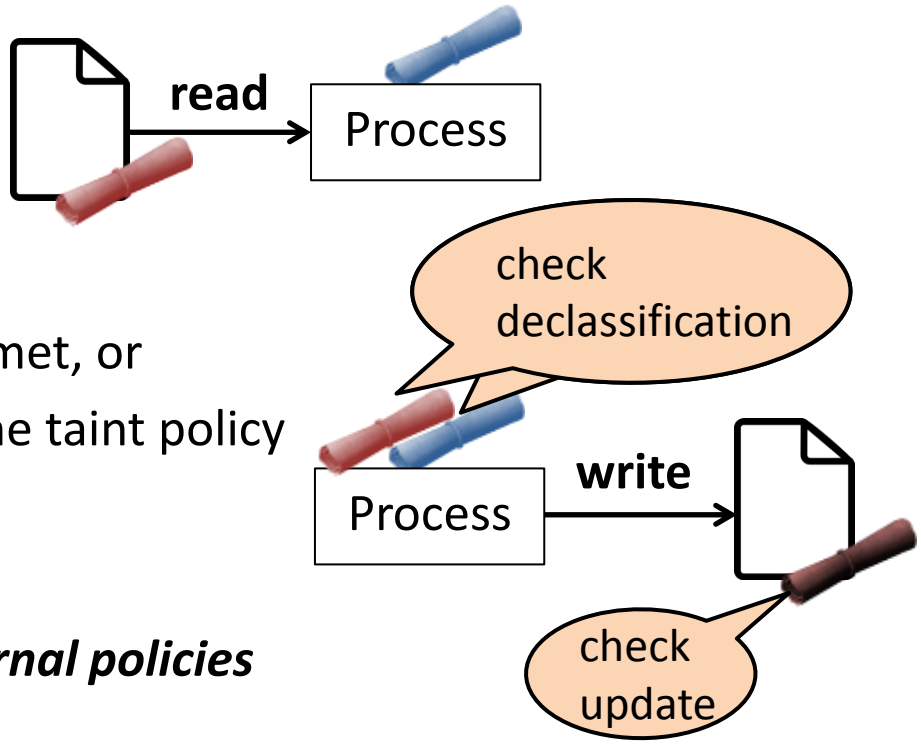
If p is external (access control)

- **read**: check *pol*.read
- **write**: check *pol*.update

If p is internal (flow control)

- **read**: *pol* is added to *p*.taint
- **write**: check *pol.*update;

  for each policy in the taint; either
  - declassification conditions are met, or
  - *pol* is at least as restrictive as the taint policy



Ensures compliance regardless of internal policies

Outline

Motivation

- Data retrieval systems, policies
- Challenges and goals

Thoth: policy compliance layer

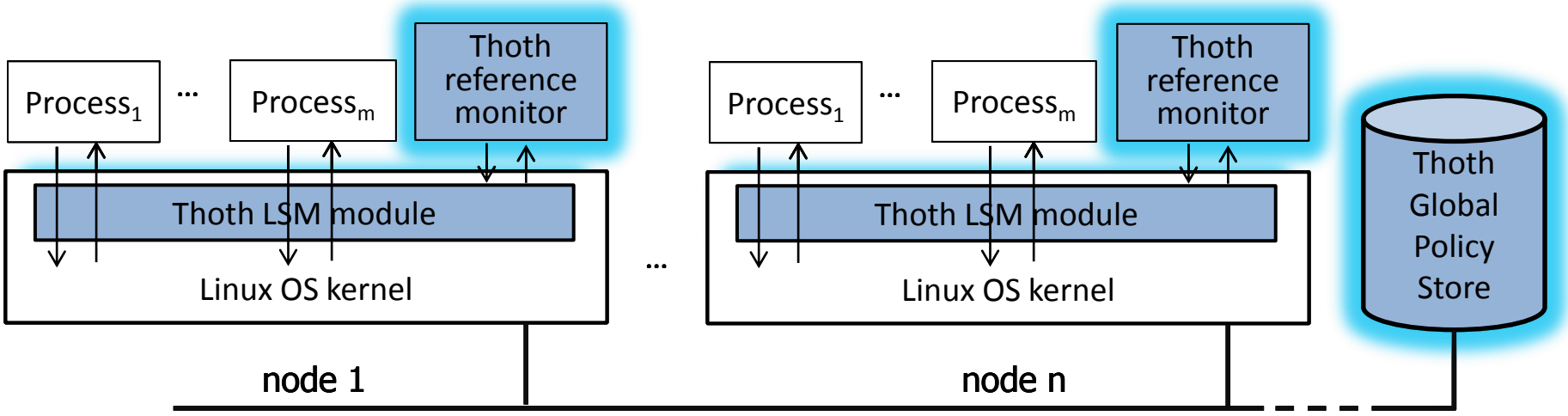
- Overview, threat model
- Policy language and examples
- Typed declassification
- Runtime enforcement

Prototype and evaluation

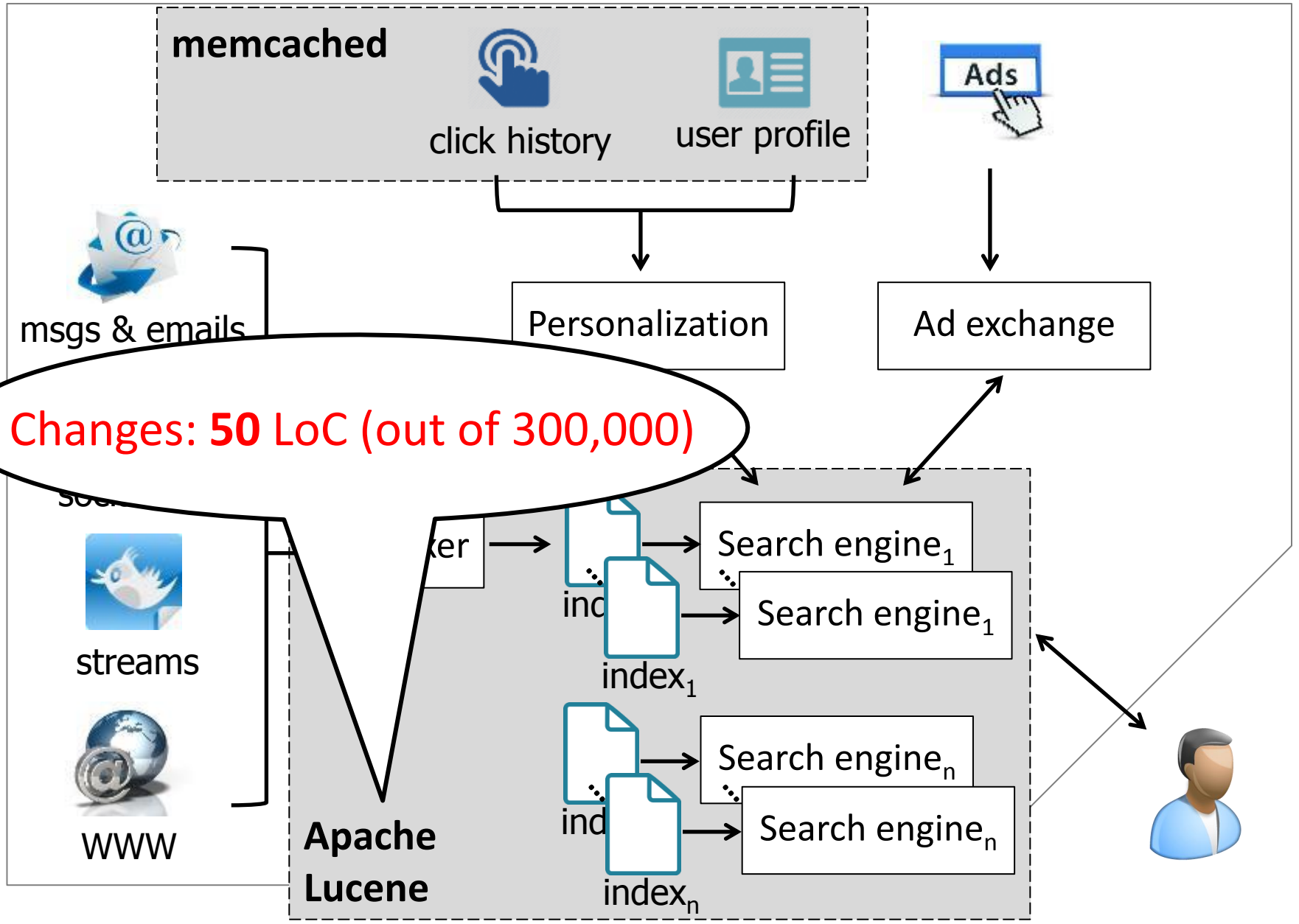
Thoth Implementation

Per-node enforcement

- Linux security module (LSM) (3,500 LoC)
  - intercepts system calls (e.g., read, write, ..)
  - exposes Thoth API (e.g., set_policy)
- Reference monitor (19,000 LoC + OpenSSL)
  - authenticates users
  - evaluates policies
- Global policy store



Prototype search engine



Performance evaluation

Setup

- 2-shard index, each shard hosted by a server
- No replication (2 servers), 2x replication (4 servers)

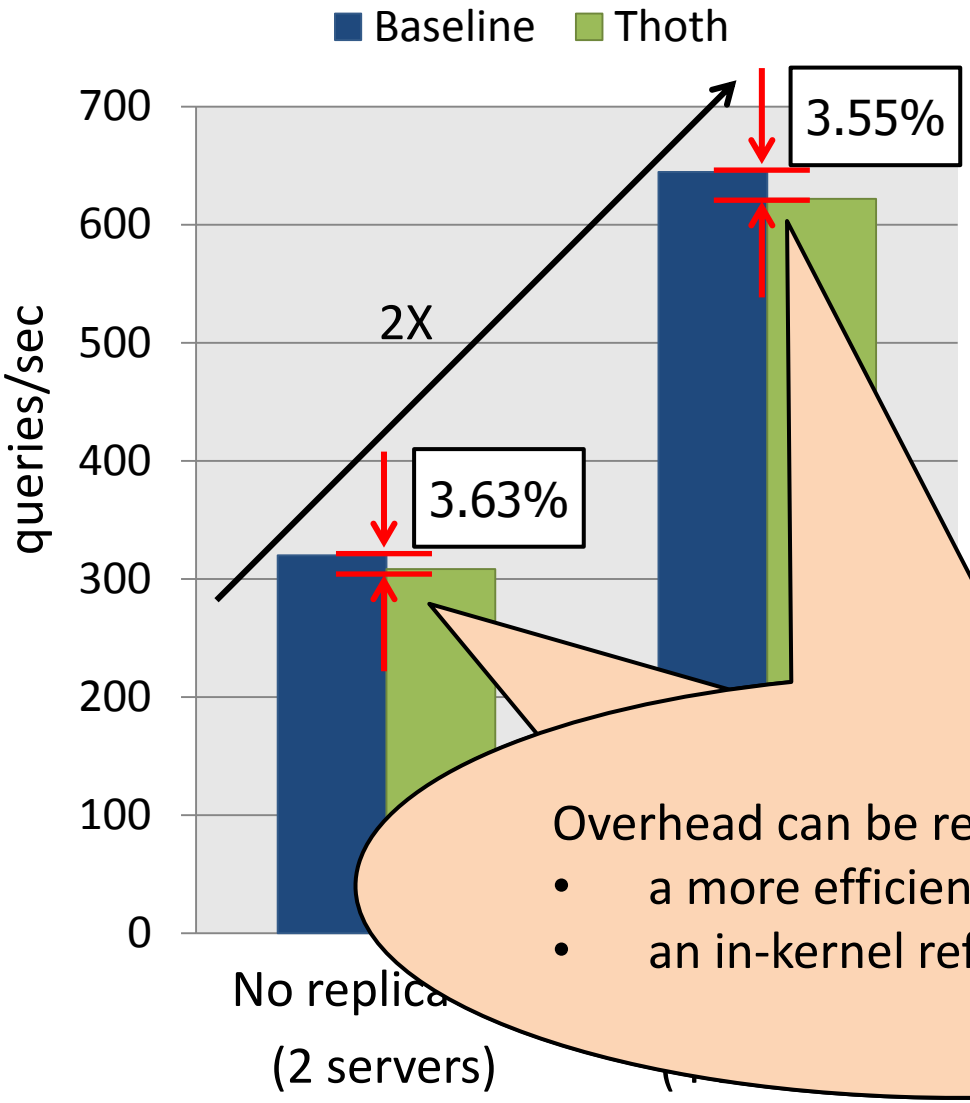
Dataset

- Wikipedia English articles; 15M documents
- Policies
  - 30% private; 50% public, 20% friends-only
  - Non-public documents allow staff access subject to logging
  - All documents have censorship clause

Workload

- Queries based on Wikipedia popularity access trace

Search throughput



Concurrent queries; clients saturate the search engine.

In-memory index and documents to fully expose Thoth overheads.

overhead:

- I/O interception + IPC to reference monitor
- policy evaluation

Overhead can be reduced significantly with

- a more efficient I/O interception
- an in-kernel reference monitor

Search latency

	Average (ms)	Overhead
Baseline	47	
Thoth	53.7	14.2%

Enforcing policy adds **6.7 milliseconds**

- More performance numbers (indexing, microbenchmarks)
- Security evaluation

Please see the paper!

Policy compliance with Thoth: Contributions

- Declarative policies; directly attached to content
- Kernel-level compliance independent of application code
- Correct source policies ensure compliance regardless of the system internal complexity

Summary

- Declarative policies attached to conduits
  - Confidentiality, integrity, declassification
- Policy compliance despite application bugs or misconfigurations
  - Process-level IFC, policies-as-taint
  - Policy-specified/Typed declassification
- Efficient policy compliance
  - Low runtime overhead
  - Minimal application code changes
- Demonstrated utility with a distributed search engine

Questions?