# Securing Workflows

## On the Use of Microservices and Metagraphs to Prevent Data Exposures


Loïc Miller

April 22, 2022

University of Strasbourg, France


Supervisors:

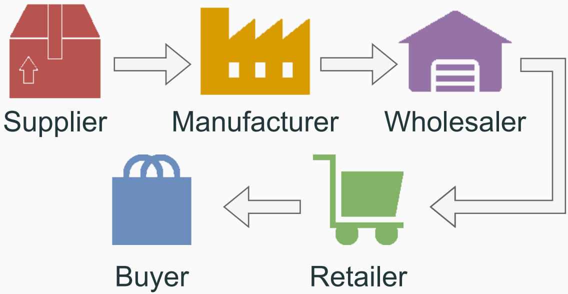
| | |
|---|---|
| Pascal | Mérindol |
| Antoine | Gallais |
| Cristel | Pelsser |

Jury:

| | |
|---|---|
| Gregory | Blanc |
| Etienne | Rivière |
| Géraldine | Texier |
| Sébastien | Tixeuil |

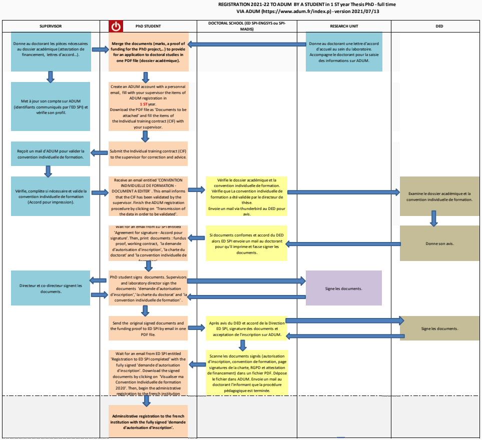
# Businesses and operations

Workflows are used **everywhere** and by **everyone**.



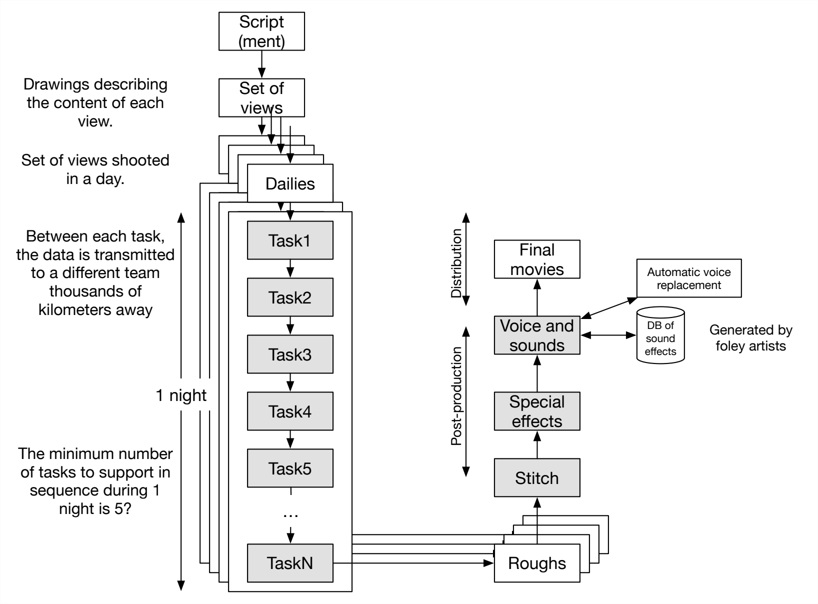

Supply chain, customer orders, ticketing systems, etc.

# Businesses and operations - Sometimes convoluted



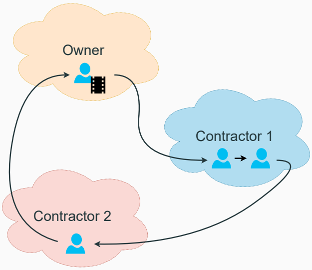

They can be complex.

# Businesses and operations - Sometimes straightforward



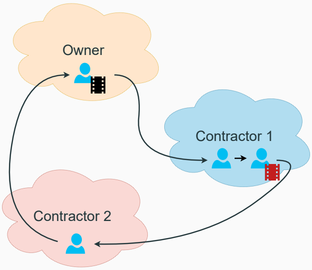
## Workflows

- **Sequence of tasks** processing a set of data.



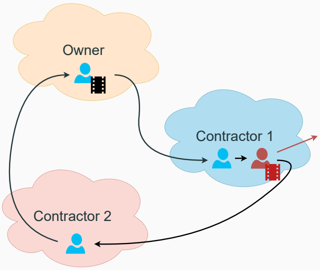
# Workflows

- **Sequence of tasks** processing a set of data.
- They involve other organizations, resulting in **multi-party workflows**.



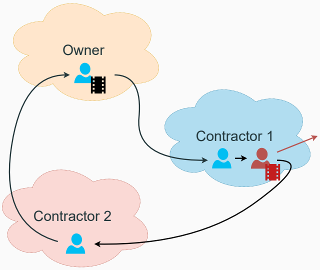
# Workflows

- **Sequence of tasks** processing a set of data.
- They involve other organizations, resulting in **multi-party workflows**.
- Complications in terms of communication and **security**.



# Workflows

- **Sequence of tasks** processing a set of data.
- They involve other organizations, resulting in **multi-party workflows**.
- Complications in terms of communication and **security**.

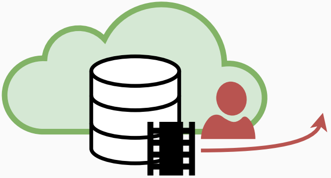



In the movie industry, data is often stored **unencrypted** in the cloud.

# Data exposures

**Sensitive data** is accessed by an **unauthorized party**.

Breach



Leak

# Data breaches

**Exploit** flaws in the security of the system.

Breach

[1] Jonathan Stempel and Jim Finkle. *Yahoo says all three billion accounts hacked in 2013 data theft*. 2017

# Data breaches

**Exploit** flaws in the security of the system.

- At rest[1] or in transport.

Breach

[1] Jonathan Stempel and Jim Finkle. *Yahoo says all three billion accounts hacked in 2013 data theft*. 2017

# Data breaches

**Exploit** flaws in the security of the system.

- At rest[1] or in transport.
- 2013 Yahoo data theft.

Breach

[1] Jonathan Stempel and Jim Finkle. *Yahoo says all three billion accounts hacked in 2013 data theft*. 2017

# Data breaches

**Exploit** flaws in the security of the system.

- At rest[1] or in transport.
- 2013 Yahoo data theft.
- **88%** of cloud breaches due to **human error**.

Breach

[1] Jonathan Stempel and Jim Finkle. *Yahoo says all three billion accounts hacked in 2013 data theft*. 2017

# Data leaks

Leak due to **processing**.

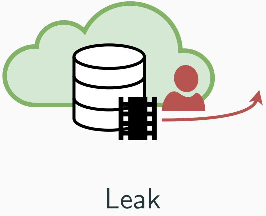
Leak

[2]Brian Krebs. *First American Financial Corp. Leaked Hundreds of Millions of Title Insurance Records*. 2019

# Data leaks

Leak due to **processing**.

- Mistake[2] or malicious.

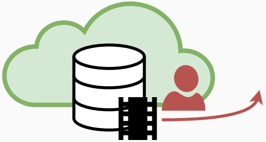



[2] Brian Krebs. *First American Financial Corp. Leaked Hundreds of Millions of Title Insurance Records*. 2019

# Data leaks

Leak due to **processing**.

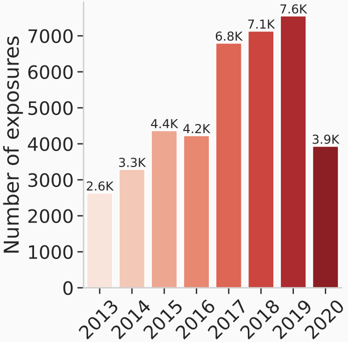
- Mistake[2] or malicious.
- 2019 First American Corp. leak.



Leak

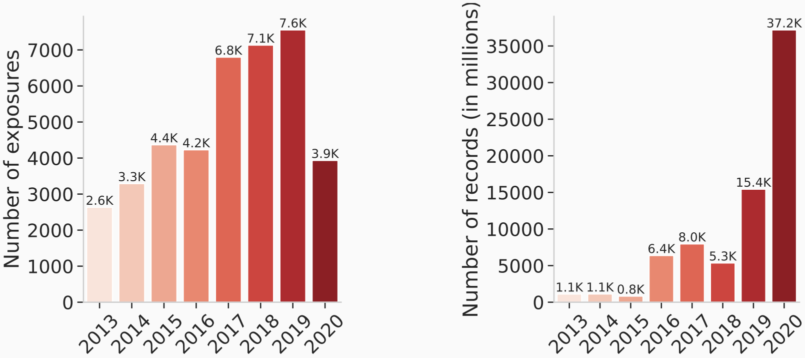
[2]Brian Krebs. *First American Financial Corp. Leaked Hundreds of Millions of Title Insurance Records*. 2019

# Exposures are trending up[3]




[3] Risk Based Security. *Data Breach Quickview 2020 Year End Report*. 2021

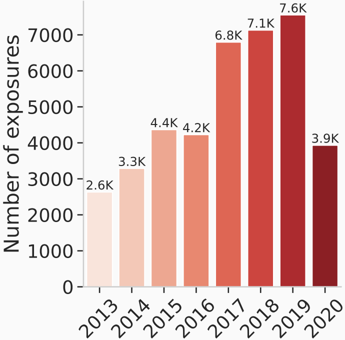
# Exposures are trending up[3]

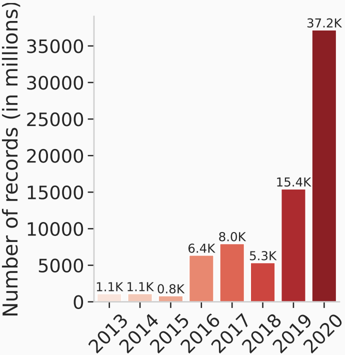



Record = **collection** of related fields.

[3] Risk Based Security. *Data Breach Quickview 2020 Year End Report*. 2021

# Exposures are trending up[3]




**82%** of compromised records from **five leaks.**

[3] Risk Based Security. *Data Breach Quickview 2020 Year End Report*. 2021

# Overview

1. Workflows are used **everywhere** and by **everyone**.

# Overview

1. Workflows are used **everywhere** and by **everyone**.
2. Exposures are **widespread**, outcomes of **critical** vulnerabilities, and happening **more**.

# Overview

1. Workflows are used **everywhere** and by **everyone**.
2. Exposures are **widespread**, outcomes of **critical** vulnerabilities, and happening **more**.
3. The shift to the cloud has brought **new security risks**.

# Research statement

**Enforce secure multi-party workflows and prevent data exposures**

# Research questions

- **RQ1:** How can we use microservices to enable multi-party workflow?

## Research questions

- **RQ1:** How can we use microservices to enable multi-party workflow?
- **RQ2:** How do we verify a policy specification corresponds to its implementation?

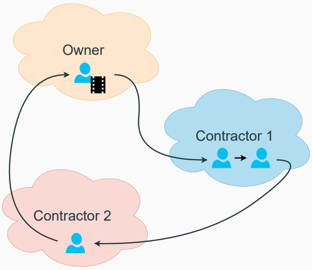
## Research questions

- **RQ1:** How can we use microservices to enable multi-party workflow?
- **RQ2:** How do we verify a policy specification corresponds to its implementation?
- **RQ3:** How do we verify a policy specification contains no redundancies?

# A Secure Infrastructure to Prevent Data Exposures

# Workflows

- Workflow is a **sequence of tasks** processed by a set of actors.
- **Owner** of the data interacts with **contractors** to realize task.
- Actors have **agents**: employee or automated service.

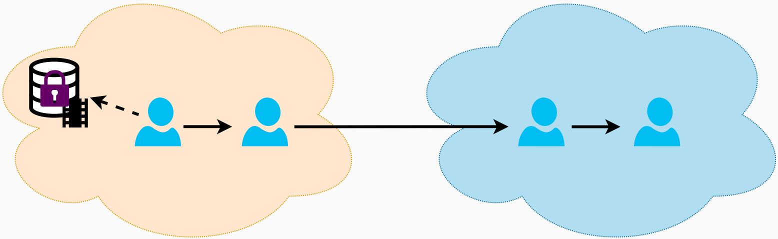


# Objectives

**How can we enforce workflows and prevent data exposures?**

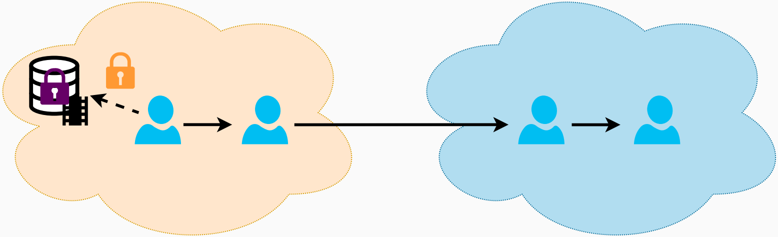
# Achieved properties

- Data security **at rest**: stored **encrypted**,



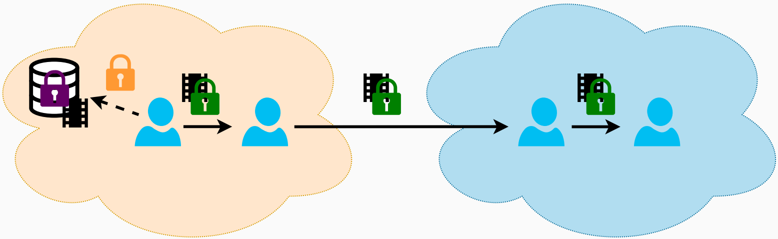
# Achieved properties

- Data security **at rest**: stored **encrypted**, access restricted by **isolation** and **policy**.



# Achieved properties

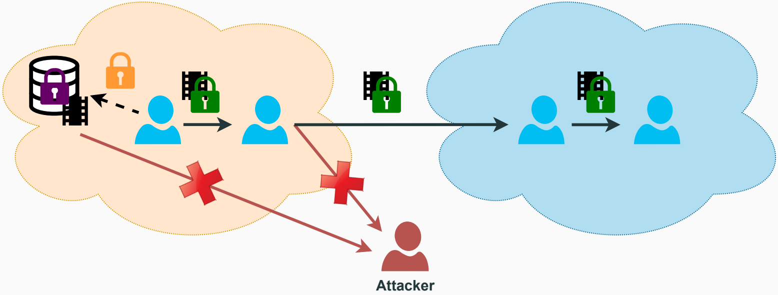
- Data security **at rest**: stored **encrypted**, access restricted by **isolation** and **policy**.
- Data security **in transport**: exchanged **encrypted**, with integrity and **authentication** checks.



# Achieved properties

- Data security **at rest**: stored **encrypted**, access restricted by **isolation** and **policy**.
- Data security **in transport**: exchanged **encrypted**, with integrity and **authentication** checks.

The data cannot be **leaked** in both cases.

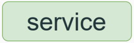
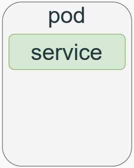


# Building block security properties

| Service |
| --- |
| service |
| Isolation |

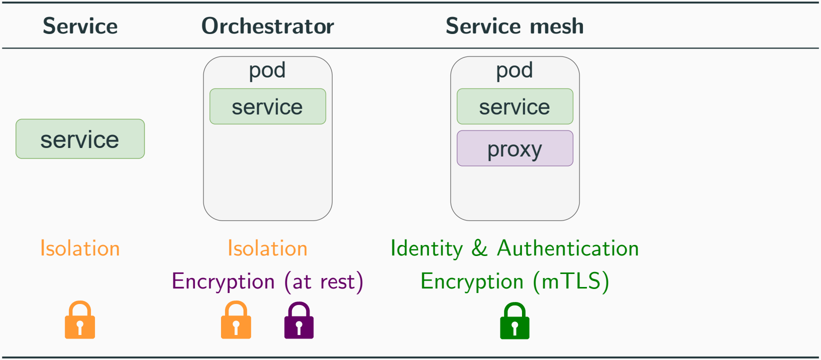
Encrypted storage, encrypted communications, policy enforcement.

# Building block security properties

| Service | Orchestrator |
|---|---|
|  |  |
| Isolation | Isolation |
| | Encryption (at rest) |

Encrypted storage, encrypted communications, policy enforcement.

# Building block security properties




Encrypted storage, encrypted communications, policy enforcement.

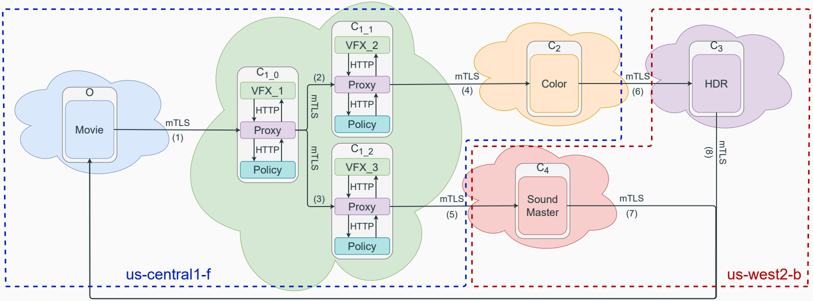
# Building block security properties

| Service | Orchestrator | Service mesh | Policy engine |
|---|---|---|---|
| service | pod: service | pod: service, proxy | pod: service, proxy, policy |
| Isolation | Isolation<br>Encryption (at rest) | Identity & Authentication<br>Encryption (mTLS) | Authorization |

Encrypted storage, encrypted communications, policy enforcement.

# Proof of Concept deployed on Google Cloud Platform

Post-production movie workflow.




- One Kubernetes cluster per actor.
- One n1-standard-v2 per cluster (2 vCPUs, 7.5 GB of memory), except the owner which has two.

# Evaluating security overhead

Pod startup time and Request duration.

# Effect of policy engine on pod startup time

- Independent-samples t-test
- Two deployments: one with policy engine and one without.
- 130 observations per pod ($N = 1820$).

Time increased by **2 seconds on average (32.72%)**.

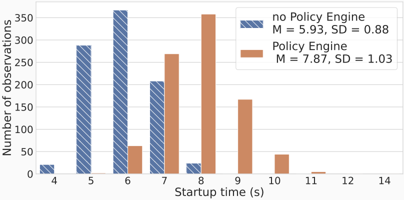
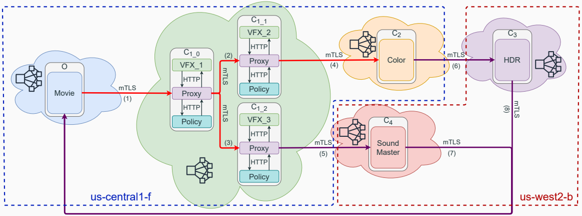


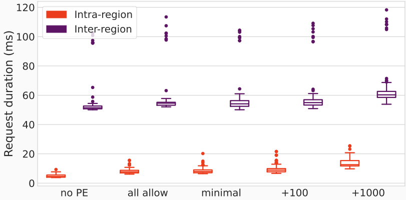

**Figure 1:** Startup time distribution

# Effect of policy size on request duration




We analyze **intra-region** and **inter-region** communications.

- +5 – 10ms on average.
- Low impact inter-region.



# Conclusion: 1st axis

- Infrastructure to secure communications in a workflow.

## Conclusion: 1st axis

- Infrastructure to secure communications in a workflow.
- Proof of concept: Code, data and guidance available.

## Conclusion: 1st axis

- Infrastructure to secure communications in a workflow.
- Proof of concept: Code, data and guidance available.
- We verified communications and security.

# Conclusion: 1st axis

- Infrastructure to secure communications in a workflow.
- Proof of concept: Code, data and guidance available.
- We verified communications and security.
- Performance analysis: Acceptable tradeoff.

## Assumption used so far

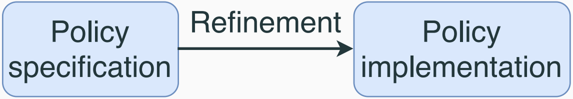
**The policy is optimal and error-free.**

## Assumption used so far

**The policy is optimal ~~and error-free~~.**

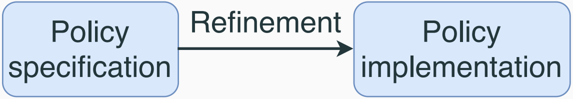
# Motivation

**Access Control** is an essential building block of security. Generally managed by a policy administrator.



## Motivation

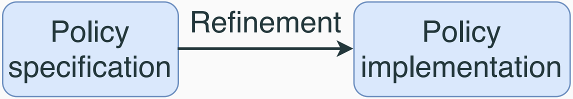
**Access Control** is an essential building block of security. Generally managed by a policy administrator.




**Prone to errors**:

# Motivation

**Access Control** is an essential building block of security. Generally managed by a policy administrator.

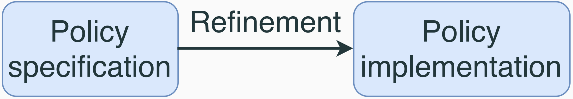



**Prone to errors**:

- Attackers.

# Motivation

**Access Control** is an essential building block of security. Generally managed by a policy administrator.

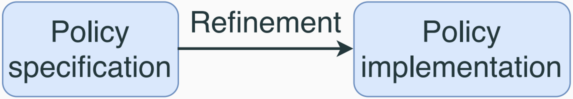



**Prone to errors**:

- Attackers.
- Distributed deployments.

## Motivation

**Access Control** is an essential building block of security. Generally managed by a policy administrator.




**Prone to errors**:

- Attackers.
- Distributed deployments.
- Refinement: Semi-automatic or automatic tools.

## Objective: Policy verification

- Verify the implementation matches the specification
- Pinpoint errors

## Why metagraphs?

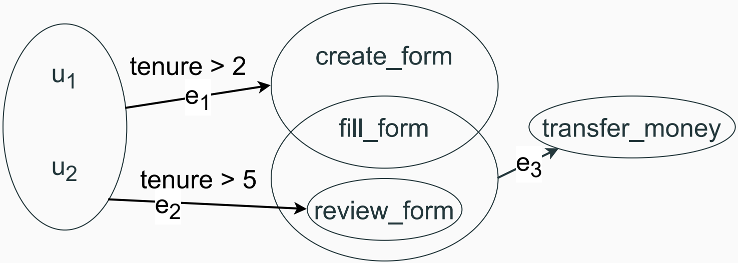
- Existing works dealing with policy verification use SAT solvers [2], decision diagrams [3] or graphs [10].

| | SAT solvers | Decision diagrams | Graphs | Metagraphs |
|---|---|---|---|---|
| Natural policy modeling | ■ | ◪ | ◪ | ■ |
| Visual representation | □ | ◪ | ◪ | ■ |

- Properties **specific to metagraphs** for detecting conflicts and redundancies[4].

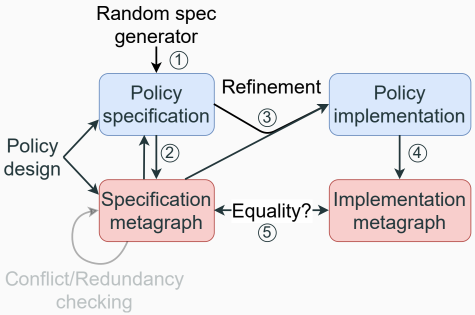
[4] Dinesha Ranathunga, Matthew Roughan, and Hung Nguyen. “Verifiable Policy-Defined Networking using Metagraphs”. In: *IEEE Transactions on Dependable and Secure Computing* (2020).

# The metagraph: a collection of directed set-to-set mappings [1]




Employees ($u_1$, $u_2$) and tasks (*create_form*, *fill_form*, *review_form*, *transfer_money*) are put into relation by the edges ($e_1, e_2, e_3$) between sets of elements.

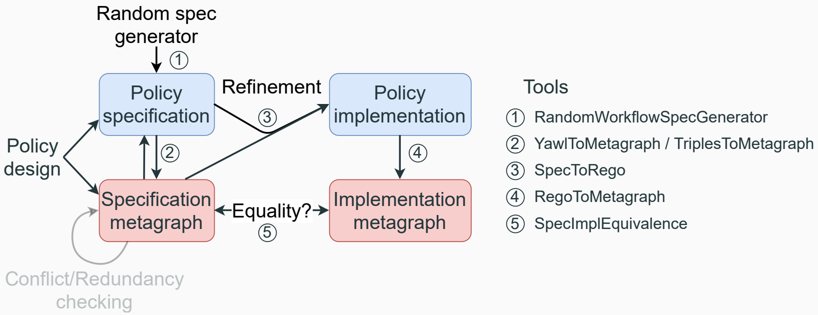
# Policy verification procedure




Tools

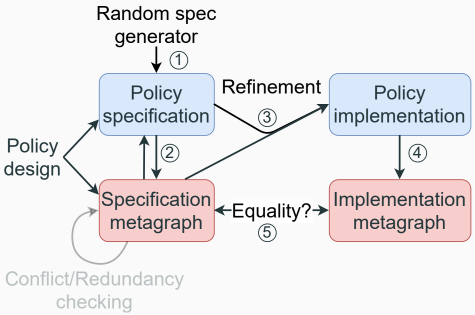
① RandomWorkflowSpecGenerator
② YawlToMetagraph / TriplesToMetagraph
③ SpecToRego
④ RegoToMetagraph
⑤ SpecImplEquivalence

# Policy verification procedure




Policy specification: YAWL, or metagraph-like format.

# Policy verification procedure



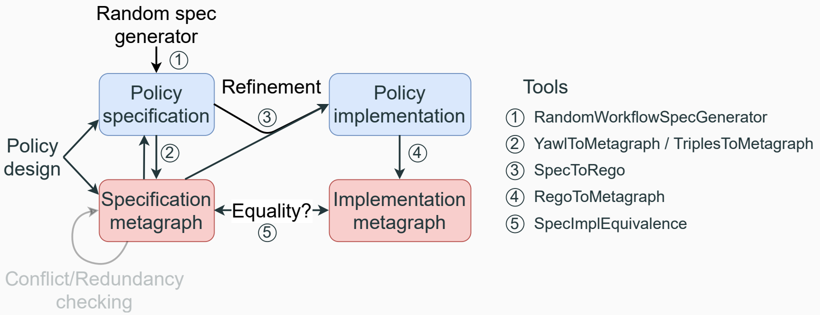

Tools

① RandomWorkflowSpecGenerator
② YawlToMetagraph / TriplesToMetagraph
③ SpecToRego
④ RegoToMetagraph
⑤ SpecImplEquivalence

Policy specification: YAWL, or metagraph-like format.

Policy implementation: Rego.

# Policy verification procedure

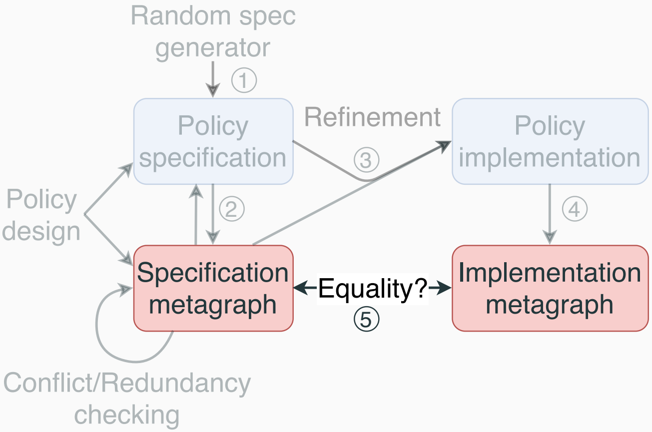



Policy specification: YAWL, or metagraph-like format.

Policy implementation: Rego.

We can pinpoint errors in the policy.

## Performance analysis ⑤



# Performance analysis 5

We measure the time required to compare two metagraphs.

# Performance analysis ⑤

**We measure the time required to compare two metagraphs.**

- **Random** policies to get more robust results.

# Performance analysis ⑤

**We measure the time required to compare two metagraphs.**

- **Random** policies to get more robust results.
- **Number of elements in the policy**: 10, 20, 30, 50 or 100.

# Performance analysis ⑤

We measure the time required to compare two metagraphs.

- **Random** policies to get more robust results.
- **Number of elements in the policy**: 10, 20, 30, 50 or 100.
- **Policy size**: 2 or 4 propositions per edge.
  $\rightarrow$ 300 policy specifications ($5 \times 2 \times 30$)

# Performance analysis ⑤

**We measure the time required to compare two metagraphs.**

- **Random** policies to get more robust results.
- **Number of elements in the policy**: 10, 20, 30, 50 or 100.
- **Policy size**: 2 or 4 propositions per edge.
  $\rightarrow$ 300 policy specifications ($5 \times 2 \times 30$)
- **Translation error rate**: 0.0, 0.2 and 0.4.
  $\rightarrow$ 27,000 policy implementations ($300 \times 3 \times 30$)

# Performance analysis (5)

**We measure the time required to compare two metagraphs.**

- **Random** policies to get more robust results.
- **Number of elements in the policy**: 10, 20, 30, 50 or 100.
- **Policy size**: 2 or 4 propositions per edge.
  $\rightarrow$ 300 policy specifications ($5 \times 2 \times 30$)
- **Translation error rate**: 0.0, 0.2 and 0.4.
  $\rightarrow$ 27,000 policy implementations ($300 \times 3 \times 30$)
- 30 measures per implementation.
  $\rightarrow$ 810,000 measures ($27000 \times 30$)

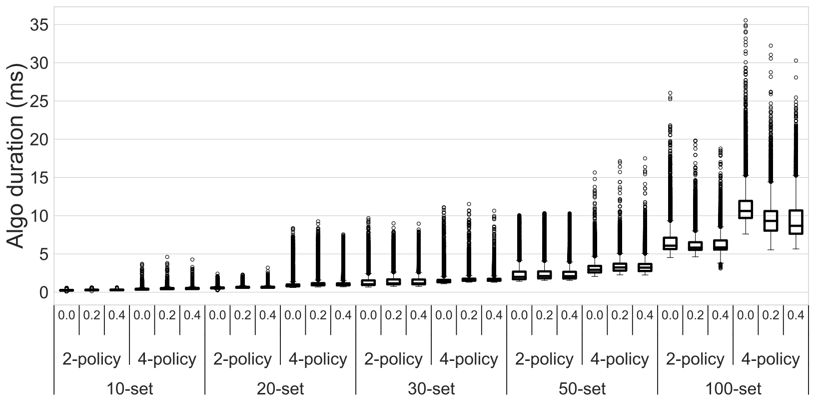
# Performance analysis ⑤

**We measure the time required to compare two metagraphs.**

- **Random** policies to get more robust results.
- **Number of elements in the policy**: 10, 20, 30, 50 or 100.
- **Policy size**: 2 or 4 propositions per edge.
  $\rightarrow$ 300 policy specifications ($5 \times 2 \times 30$)
- **Translation error rate**: 0.0, 0.2 and 0.4.
  $\rightarrow$ 27,000 policy implementations ($300 \times 3 \times 30$)
- 30 measures per implementation.
  $\rightarrow$ 810,000 measures ($27000 \times 30$)

Rego policy files between 305 and 24729 lines of code, **in line** with observed policies.

# Time increases with number of elements and policy size




- Verification times between 0 and 12 ms on average.
- Error rate has a negligible effect (correlation of 0.01).

# Conclusion: 2nd axis

- **New policy verification method using metagraphs.**

[5] Code, data and guidance at https://github.com/loicmiller/policy-verification

## Conclusion: 2nd axis

- **New policy verification method using metagraphs.**
- Motivated the use of metagraphs to represent and verify policies.

[5] Code, data and guidance at https://github.com/loicmiller/policy-verification

# Conclusion: 2nd axis

- **New policy verification method using metagraphs.**
- Motivated the use of metagraphs to represent and verify policies.
- Developed **suite of tools**[5]**:**
  - RandomPolicySpecGenerator
  - YawlToMetagraph / SpecToRego
  - RegoToMetagraph
  - SpecImplEquivalence

[5]Code, data and guidance at https://github.com/loicmiller/policy-verification

# Conclusion: 2nd axis

- **New policy verification method using metagraphs.**
- Motivated the use of metagraphs to represent and verify policies.
- Developed **suite of tools**[5]**:**
  - RandomPolicySpecGenerator
  - YawlToMetagraph / SpecToRego
  - RegoToMetagraph
  - SpecImplEquivalence
- Evaluated our method: verification times **between 0 and 12 ms** on average.

[5] Code, data and guidance at https://github.com/loicmiller/policy-verification

## Assumption used so far

The policy is optimal ~~and error-free~~.

# Assumption used so far

**~~The policy is optimal and error-free.~~**

# Goal: Identify redundancies in a (security) policy.

Elements which do not change the behavior of the policy if removed.

## Goal: Identify redundancies in a (security) policy.

Elements which do not change the behavior of the policy if removed.

**Motivation**: Speed, reduce clutter, reduce errors.

# Goal: Identify redundancies in a (security) policy.

Elements which do not change the behavior of the policy if removed.

**Motivation**: Speed, reduce clutter, reduce errors.

Metagraphs have already been used to detect redundancies [9]...

## Goal: Identify redundancies in a (security) policy.

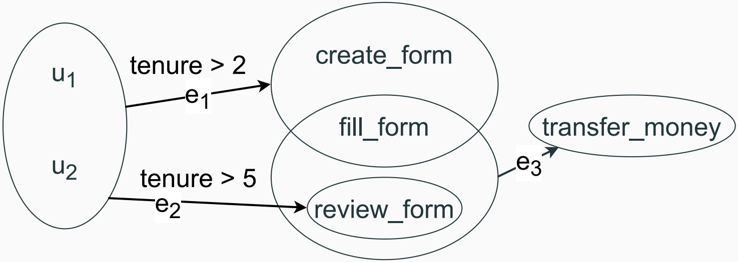
Elements which do not change the behavior of the policy if removed.

**Motivation**: Speed, reduce clutter, reduce errors.

Metagraphs have already been used to detect redundancies [9]...

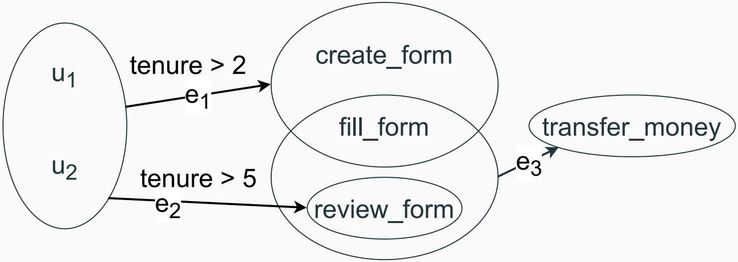
...but the current solution has shortcomings.

# Metapaths are not simple paths




$M_1(\{u_1, u_2\}, \{transfer\_money\}) = \{e_1, e_2, e_3\}$ is a **metapath**.

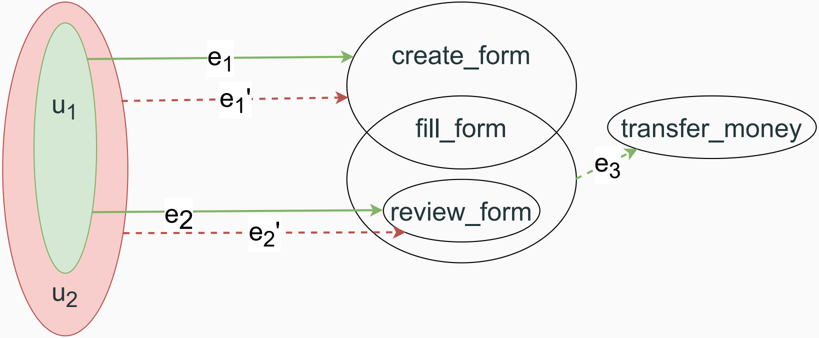
# Metapaths are not simple paths




$M_1(\{u_1, u_2\}, \{transfer\_money\}) = \{e_1, e_2, e_3\}$ is a **metapath**.

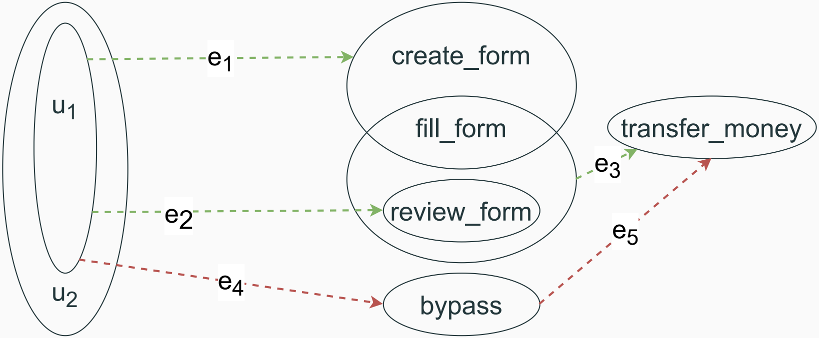
A metapath is **dominant** if it is both **input-dominant** and **edge-dominant**.

# Input dominance - Minimality of input




$M_1(\{u_1, u_2\}, \{transfer\_money\}) = \{e_1', e_2', e_3\}$ is not input-dominant because $M_2(\{u_1\}, \{transfer\_money\}) = \{e_1, e_2, e_3\}$ is a metapath.

## Edge dominance - Minimality of edges




$M_1(\{u_1\}, \{transfer\_money\}) = \{e_1, e_2, e_3, e_4, e_5\}$ is not edge-dominant because $M_2(\{u_1\}, \{transfer\_money\}) = \{e_1, e_2, e_3\}$ is a metapath.

# Dominant metapaths identify minimal access.

Elements not on any dominant metapath are redundant.

Rationale: In every possible access, we can do without the redundancy.

# Dominant metapaths identify minimal access.

Elements not on any dominant metapath are redundant.

Rationale: In every possible access, we can do without the redundancy.

> *"...simply check all feasible metapaths in a policy metagraph for edge and input dominance, if either fails, the policy includes redundancies" - Ranathunga et al. [9].*

# Dominant metapaths identify minimal access.

Elements not on any dominant metapath are redundant.

Rationale: In every possible access, we can do without the redundancy.

> *"...simply check all feasible metapaths in a policy metagraph for edge and input dominance, if either fails, the policy includes redundancies" - Ranathunga et al. [9].*

Great! Problem solved, right?

## In reality...

- Checking all metapaths takes too much time.

# In reality...

- Checking all metapaths takes too much time.
- Even worse, just finding all metapaths takes too much time.

# Finding all metapaths takes too much time

Algorithm is based on computing the **transitive closure** of $A^*$, the adjacency matrix - $(n^3)^m$.

# Finding all metapaths takes too much time

Algorithm is based on computing the **transitive closure** of $A^*$, the adjacency matrix - $(n^3)^m$.

- Equivalent to finding **all simple paths** between **all pairs** of elements.

# Finding all metapaths takes too much time

Algorithm is based on computing the **transitive closure** of $A^*$, the adjacency matrix - $(n^3)^m$.

- Equivalent to finding **all simple paths** between **all pairs** of elements.
- Does not find all metapaths.

# Finding all metapaths takes too much time

Algorithm is based on computing the **transitive closure** of $A^*$, the adjacency matrix - $(n^3)^m$.

- Equivalent to finding **all simple paths** between **all pairs** of elements.
- Does not find all metapaths.
- The redundant metapaths found are **not minimal**.

# Finding all metapaths takes too much time

Algorithm is based on computing the **transitive closure** of $A^*$, the adjacency matrix - $(n^3)^m$.

- Equivalent to finding **all simple paths** between **all pairs** of elements.
- Does not find all metapaths.
- The redundant metapaths found are **not minimal**.

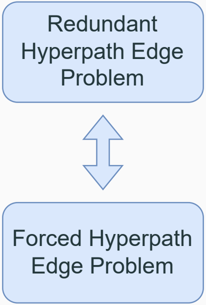
Implementing their method, it took **1 hour** to process metagraphs of **13 elements at most**.

# Alternatives?

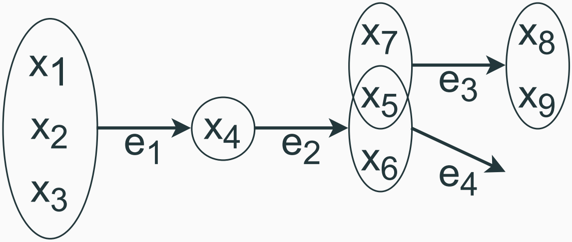
- No simple algorithm.
- Can it be done?
- NP-Hard? **Yes.**

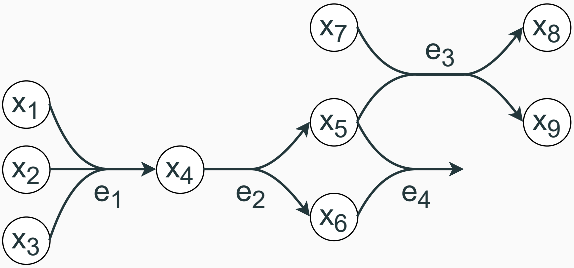
# Alternatives?

- No simple algorithm.
- Can it be done?
- NP-Hard? **Yes.**

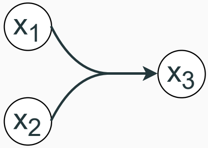


# Hypergraphs, a structure related to metagraphs.

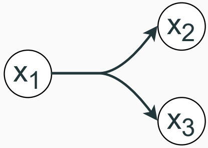


# Types of hypergraphs (B, F, BF)

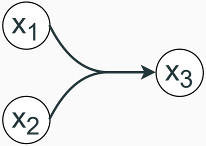



B-edge

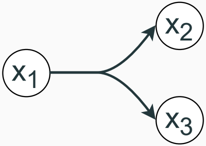



F-edge

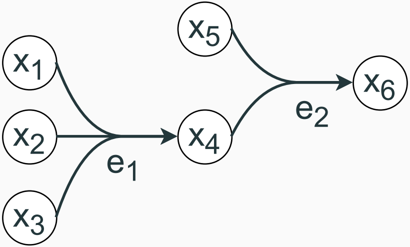
# Types of hypergraphs (B, F, BF)




B-edge

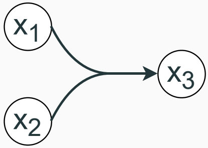



F-edge

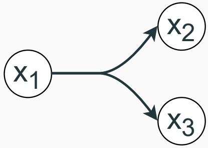



B-hypergraph

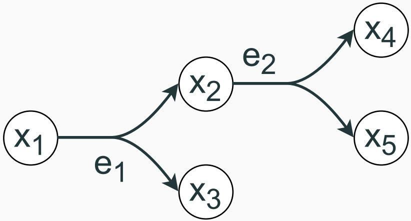
# Types of hypergraphs (B, F, BF)




B-edge

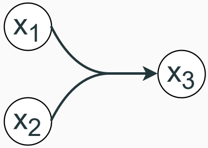



F-edge

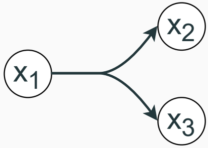



F-hypergraph

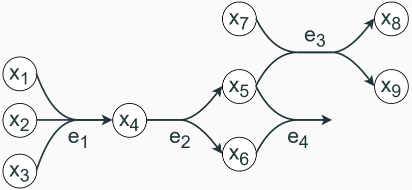
# Types of hypergraphs (B, F, BF)




B-edge

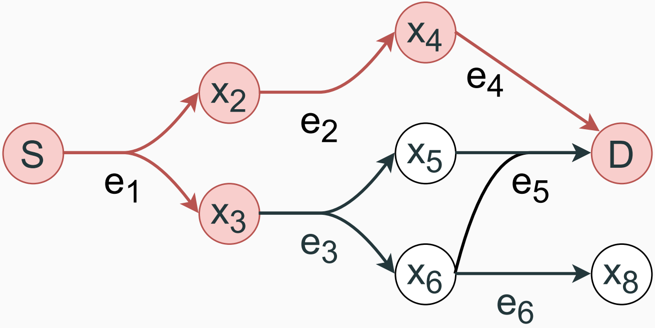



F-edge



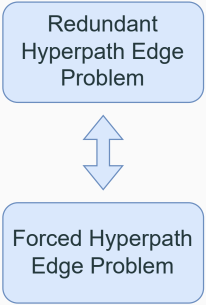

BF-hypergraph

# Hyperpaths

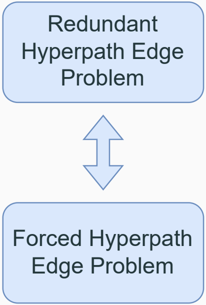



- Minimal sub-hypergraph $\mathcal{H}'$.
- Invertex of new edge must already be in hyperpath.

# Proof that finding redundancies is NP-Hard

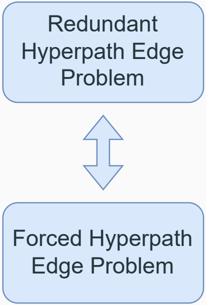


# Proof that finding redundancies is NP-Hard




- Find all redundant edges in $\mathcal{H}$.

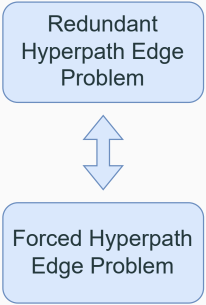
## Proof that finding redundancies is NP-Hard




- Find all redundant edges in $\mathcal{H}$.

- Is there an input-dominant hyperpath in $\mathcal{H}$ using $e$.

# Proof that finding redundancies is NP-Hard



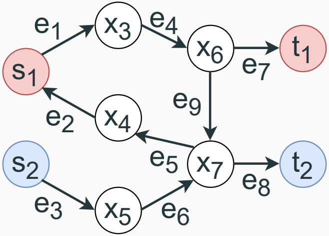

- Find all redundant edges in $\mathcal{H}$.
- Is there an input-dominant hyperpath in $\mathcal{H}$ using $e$.

An input-dominant hyperpath using $e$ means $e$ is not redundant.

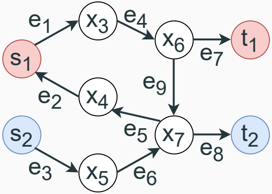
# Proving the FHEP is NP-Complete with simple graphs

The **Forced Path Edge Problem**: simple graph version of the FHEP.

Reduction from **2-VDPP**, a known NP-Hard problem.



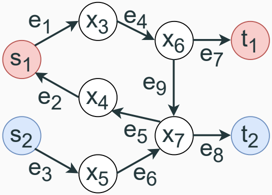
# Proving the FHEP is NP-Complete with simple graphs



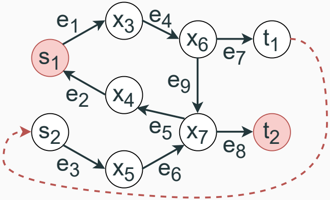

Disjoint paths (2-VDPP)

Suppose we have an instance of 2-VDPP.

# Proving the FHEP is NP-Complete with simple graphs



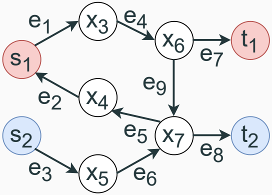

Disjoint paths (2-VDPP)



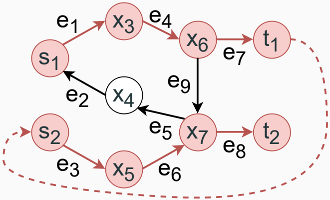

$G'$ construction (FPEP)

**Construction** $G'$ with added forced edge.

# Proving the FHEP is NP-Complete with simple graphs



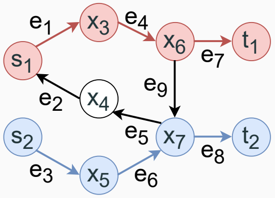

Disjoint paths (2-VDPP)



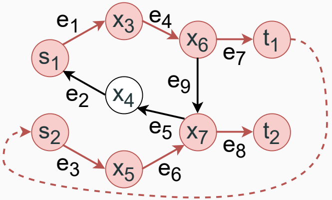

$G'$ construction (FPEP)

A solution to FPEP is a simple path from $s_1$ to $t_2$ via $e'$.

# Proving the FHEP is NP-Complete with simple graphs



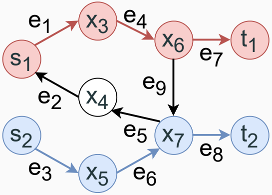

Disjoint paths (2-VDPP)



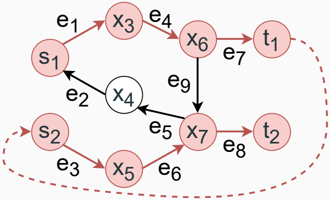

$G'$ construction (FPEP)

A solution to FPEP is a solution to 2-VDPP.

# Proving the FHEP is NP-Complete with simple graphs



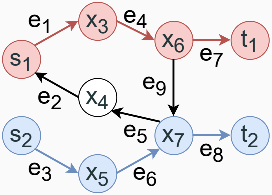

Disjoint paths (2-VDPP)



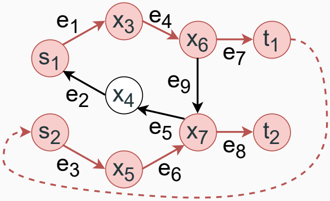

$G'$ construction (FPEP)

**The Forced Path Edge Problem is NP-Complete.**

# Proving the FHEP is NP-Complete with simple graphs




Disjoint paths (2-VDPP)




$G'$ construction (FPEP)

**The Forced Path Edge Problem is NP-Complete.**

**Corollary: the FHEP is NP-Complete.**

## Complexity summary

| | | | Redundancy |
|---|---|---|---|
| Forced Edge | Cyclic | B | NP-Hard [13] |
| | | F | NP-Hard [13] |
| | | BF | NP-Hard [13] |
| | Acyclic | B | P (linear) [13] |
| | | F | ? |
| | | BF | ? |

## Complexity summary

| | | | Redundancy |
|---|---|---|---|
| Forced Edge | Cyclic | B | NP-Hard [13] |
| | | F | NP-Hard [13] |
| | | BF | NP-Hard [13] |
| | Acyclic | B | P (linear) [13] |
| | | F | ? |
| | | BF | ? |

## Complexity summary

| | | | Redundancy |
|---|---|---|---|
| Forced Edge | Cyclic | B | NP-Hard [13] |
| | | F | NP-Hard [13] |
| | | BF | NP-Hard [13] |
| | Acyclic | B | P (linear) [13] |
| | | F | ? |
| | | BF | ? |

# Complexity summary

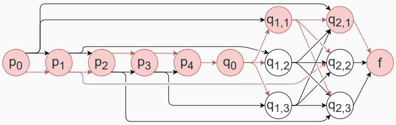
| | | | Redundancy |
|---|---|---|---|
| Forced Edge | Cyclic | B | NP-Hard [13] |
| | | F | NP-Hard [13] |
| | | BF | NP-Hard [13] |
| | Acyclic | B | P (linear) [13] |
| | | F | NP-Hard [8] |
| | | BF | NP-Hard [8] |

# Acyclic F-hypergraph proof

Reduction from 3-SAT.

$$(v_1 \vee v_2 \vee \neg v_4) \wedge$$
$$(v_1 \vee \neg v_2 \vee \neg v_3)$$

3-SAT instance




Our construction.

The FHEP in an acyclic F-hypergraph is NP-Complete.

## Trying to get a correct result faster

- Correct result by enumeration (1 hour / 6 elements).

## Trying to get a correct result faster

- Correct result by enumeration (1 hour / 6 elements).
- SAT formulation.

## Trying to get a correct result faster

- Correct result by enumeration (1 hour / 6 elements).
- SAT formulation.

What aspects of metapaths can we exploit to be faster?

# Trying to get a correct result faster

- Correct result by enumeration (1 hour / 6 elements).
- SAT formulation.

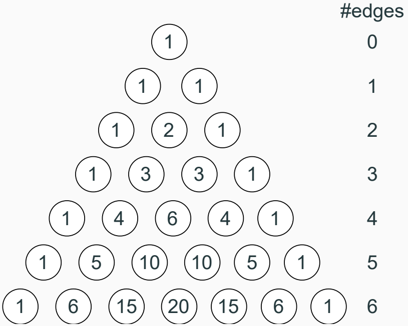
What aspects of metapaths can we exploit to be faster?

**Dominance!**

- We only need **dominant metapaths** to compute the solution, not all of them.
- A dominant metapath is **minimal**, no need to test **supersets**.
- Testing if a metapath is dominant is **polynomial**.

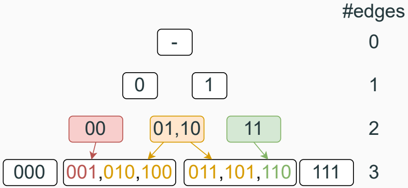
# Using Pascal's triangle

- Build iteratively from the top.
- Only add set if not dominant.
- This guarantees we test only when necessary.

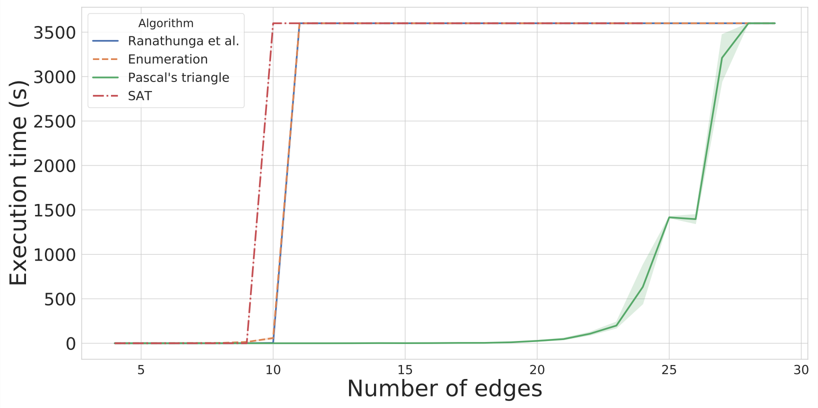


# Using Pascal's triangle

- Build iteratively from the top.
- Only add set if not dominant.
- This guarantees we test only when necessary.



# Performance results




- SAT almost instant for generated instances.
- Pascal's triangle method up to 28 edges.

## Conclusion: 3rd axis

- Finding redundancies is NP-Hard.
- Roadblocks in SAT formulation.
- Efficient algorithm using Pascal's triangle.

## Conclusion

- Microservices to enable leak-free multi-party workflows.

# Conclusion

- Microservices to enable leak-free multi-party workflows.
- Metagraphs are a useful model for policies.

# Conclusion

- Microservices to enable leak-free multi-party workflows.
- Metagraphs are a useful model for policies.
- Policy verification to check implementations.

# Conclusion

- Microservices to enable leak-free multi-party workflows.
- Metagraphs are a useful model for policies.
- Policy verification to check implementations.
- Policy analysis to check specifications.

# Contributions of this thesis

This thesis therefore focuses on the prevention of data exposures, in workflows in particular.

| # | Contribution | Tool | Repository (`github.com/`) |
|---|---|---|---|
| 1 | Secure infrastructure design [6, 5] | Proof of Concept | `loicmiller/secure-workflow` |
| 2 | Policy verification [7, 5] | Policy verification | `loicmiller/policy-verification` |
| | | MGToolkit for Python 3 | `loicmiller/mgtoolkit` |
| 3 | Policy redundancy elimination [8] | Redundancy elimination | `loicmiller/policy-analysis` |
| | | SAT formulation | `loicmiller/fhep-sat-formulation` |

All code, data, results and figures are publicly available.


- Miller et al. "Towards Secure and Leak-Free Workflows Using Microservice Isolation". In: HPSR (2021).
- Miller et al. "Verification of Cloud Security Policies". In: HPSR (2021).
- Miller et al. "Securing Workflows Using Microservices and Metagraphs". In: Electronics (2021).
- Gil Pons et al. "Finding (s,d)-Hypernetworks in F-Hypergraphs is NP-Hard". In: arXiv (2022).

# Future Works

# Short term goals

- Improved SAT generation (De Morgan's Law).
- Explore related complexity issues.

## Midterm goals

- Explore security properties (separation of duties).
- Explore impact of workflow patterns (cancellation).

# Long term goals

- Constitution of a policy benchmark dataset.
- Distributed policy (least privilege).

## Distributed policy

- Split a single policy across distributed elements?
- Verify correctness? Least privilege?

- Policy composition (algebras).
- Who specifies what? Multiple languages?

**Thank you!**

[1] Amit Basu and Robert W Blanning. *Metagraphs and their applications*. Vol. 15. Springer Science & Business Media, 2007.

[2] Padmalochan Bera, Soumya Kanti Ghosh, and Pallab Dasgupta. "Policy based security analysis in enterprise networks: A formal approach". In: *IEEE Transactions on Network and Service Management* 7.4 (2010), pp. 231–243.

[3] Mohamed G Gouda and Alex X Liu. "Structured firewall design". In: *Computer networks* 51.4 (2007), pp. 1106–1120.

[4] Brian Krebs. *First American Financial Corp. Leaked Hundreds of Millions of Title Insurance Records*. 2019. URL: https://krebsonsecurity.com/2019/05/first-american-financial-corp-leaked-hundreds-of-millions-of-title-insurance-records/.

[5] Loïc Miller et al. "Securing Workflows Using Microservices and Metagraphs". In: *Electronics* 10.24 (2021), p. 3087.

[6] Loïc Miller et al. "Towards Secure and Leak-Free Workflows Using Microservice Isolation". In: *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. IEEE. 2021, pp. 1–5. DOI: 10.1109/HPSR52026.2021.9481820.

[7] Loïc Miller et al. "Verification of Cloud Security Policies". In: *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*. IEEE. 2021, pp. 1–5. DOI: 10.1109/HPSR52026.2021.9481870.

[8] Reynaldo Gil Pons, Max Ward, and Loïc Miller. *Finding (s,d)-Hypernetworks in F-Hypergraphs is NP-Hard*. 2022. arXiv: 2201.04799 [cs.DM].

[9] Dinesha Ranathunga, Matthew Roughan, and Hung Nguyen. "Verifiable Policy-Defined Networking using Metagraphs". In: *IEEE Transactions on Dependable and Secure Computing* (2020).

[10] Dinesha Ranathunga et al. "Malachite: Firewall policy comparison". In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE. 2016, pp. 310–317.

[11] Risk Based Security. *Data Breach Quickview 2020 Year End Report*. 2021. URL: https://pages.riskbasedsecurity.com/en/en/2020-yearend-data-breach-quickview-report.

[12] Jonathan Stempel and Jim Finkle. *Yahoo says all three billion accounts hacked in 2013 data theft*. 2017. URL: https://www.reuters.com/article/us-yahoo-cyber/yahoo-says-all-three-billion-accounts-hacked-in-2013-data-theft-idUSKCN1C8201.

[13] Antonio P Volpentesta. “Hypernetworks in a directed hypergraph”. In: *European Journal of Operational Research* 188.2 (2008), pp. 390–405.

# Effect of policy engine on pod startup time

- Independent-samples t-test
- Two deployments: one with policy engine and one without.
- 130 observations per pod ($N = 1820$).

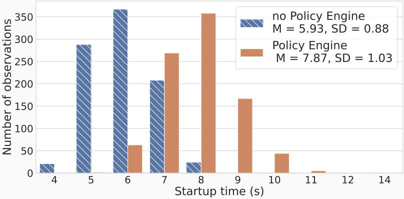
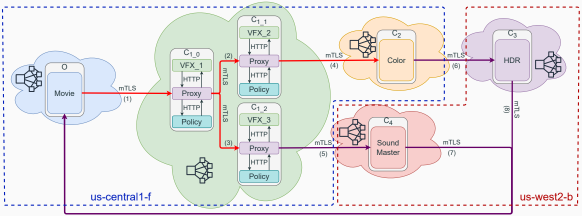



**Figure 2:** Startup time distribution

- $t(1818) = 43.19$, $p < 0.001$
- High effect size: $d = 1.985$
- High statistical power: $1 - \beta = 0.999$

Time increased by **2 seconds on average (32.72%)**.

# Effect of policy size on request duration




We analyze **intra-region** and **inter-region** communications.

One-way between subjects ANOVA.

40 observations per communication per scenario ($N = 1600$).

Policy scenarios: `no opa`, `all allow`, `minimal` ,+100 (+147%), +1000 (+1470%).

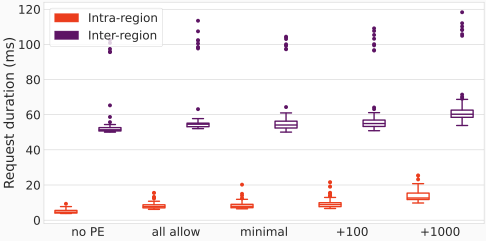
# High (low) impact on intra (inter) region request time

## Intra-region

- $F(4, 795) = 364.05$, $p < 0.001$
- **High** effect size: $\eta_p^2 = 0.65$

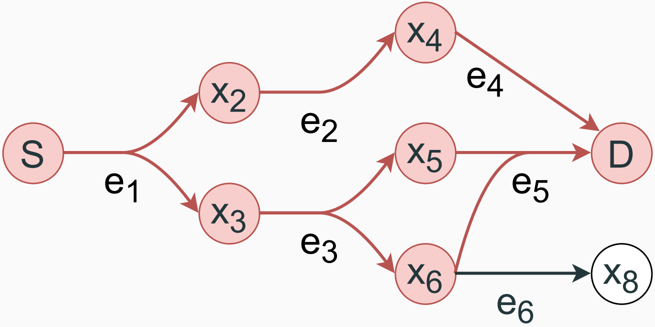
## Inter-region

- $F(4, 795) = 15.23$, $p < 0.001$
- **Low** effect size: $\eta_p^2 = 0.07$




- **Significant difference in request duration between the five scenarios for both types.**

# $(\mathcal{S},\mathcal{D})$-hypernetwork: Sum of all hyperpaths



# Finding (s,d)-Hypernetworks in F-Hypergraphs is NP-Hard

- FHEP reducible to SDHP.
- If FHEP is NP-complete, SDHP is NP-Hard.
- Reduction from 3-SAT (NP-Complete).

# We take an instance of 3-SAT

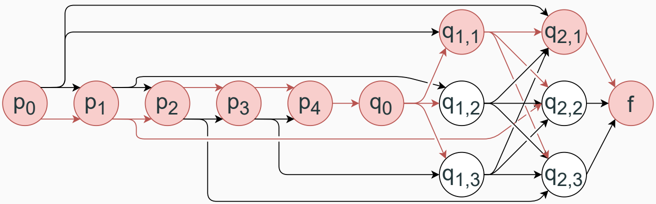
$$(v_1 \lor v_2 \lor \neg v_4) \land (v_1 \lor \neg v_2 \lor \neg v_3)$$

We construct a corresponding acyclic F-hypergraph.

Any forced edge hyperpath corresponds to a solution to 3-SAT instance.

# The construction

$$(v_1 \vee v_2 \vee \neg v_4) \wedge (v_1 \vee \neg v_2 \vee \neg v_3)$$




$p_0$ is the source. $f$ the destination.

$p_i$ for each variable. $q_{i,1}, q_{i,2}, q_{i,3}$ for each clause.

Edge where a variable appears in a clause.

# Complexity summary for finding a hyperpath

| | | | Edge-dom | Input-dom | Dom |
|---|---|---|---|---|---|
| Regular | Cyclic | B | P (linear) | P (linear) | P |
| | | F | P | P | P |
| | | BF | P | P | P |
| | Acyclic | B | P (linear) | P | P |
| | | F | P | P | P |
| | | BF | P | P | P |
| Forced Edge | Cyclic | B | NP-Hard [13] | ? | NP-Hard [13] |
| | | F | NP-Hard [13] | ? | NP-Hard [13] |
| | | BF | NP-Hard [13] | ? | NP-Hard [13] |
| | Acyclic | B | P (linear) [13] | ? | ? |
| | | F | ? | ? | ? |
| | | BF | ? | ? | ? |

# Complexity summary for finding a hyperpath

| | | | Edge-dom | Input-dom | Dom |
|---|---|---|---|---|---|
| Regular | Cyclic | B | P (linear) | P (linear) | P |
| | | F | P | P | P |
| | | BF | P | P | P |
| | Acyclic | B | P (linear) | P | P |
| | | F | P | P | P |
| | | BF | P | P | P |
| Forced Edge | Cyclic | B | NP-Hard [13] | ? | NP-Hard [13] |
| | | F | NP-Hard [13] | ? | NP-Hard [13] |
| | | BF | NP-Hard [13] | ? | NP-Hard [13] |
| | Acyclic | B | P (linear) [13] | ? | ? |
| | | F | NP-Hard [8] | ? | NP-Hard [8] |
| | | BF | NP-Hard [8] | ? | NP-Hard [8] |