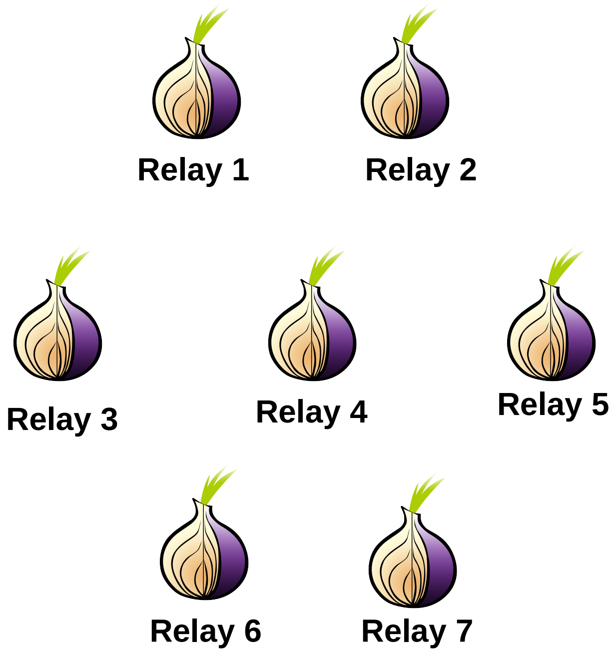
ConsenSGX: Scaling Anonymous Communications Networks with Trusted Execution Environments

Sajin Sasy and Ian Goldberg

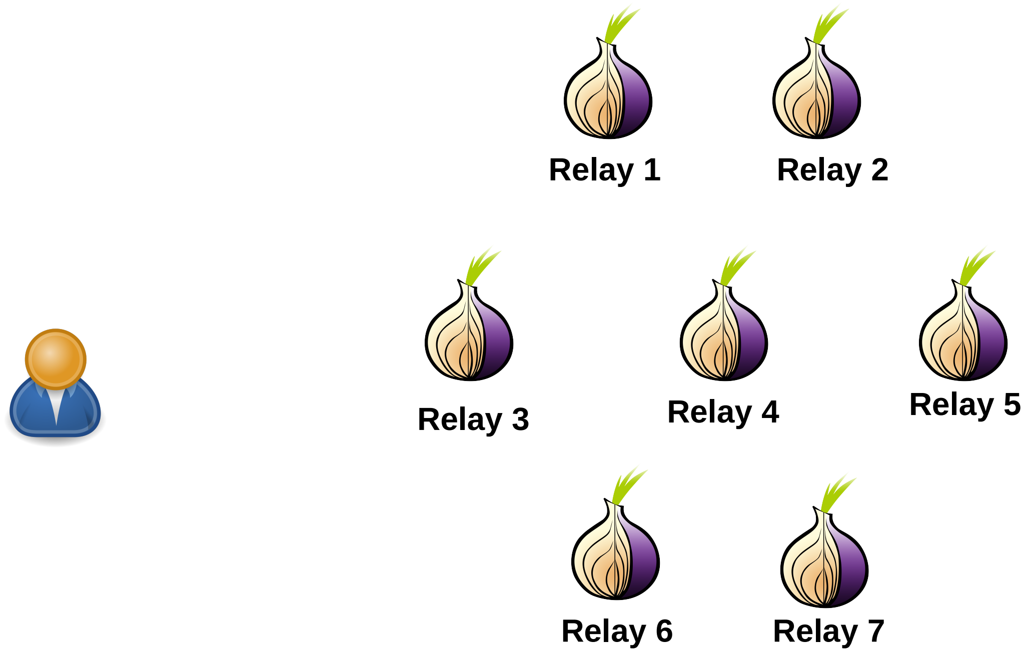
UNIVERSITY OF WATERLOO

CrySP
Cryptography, Security, and Privacy Research Group

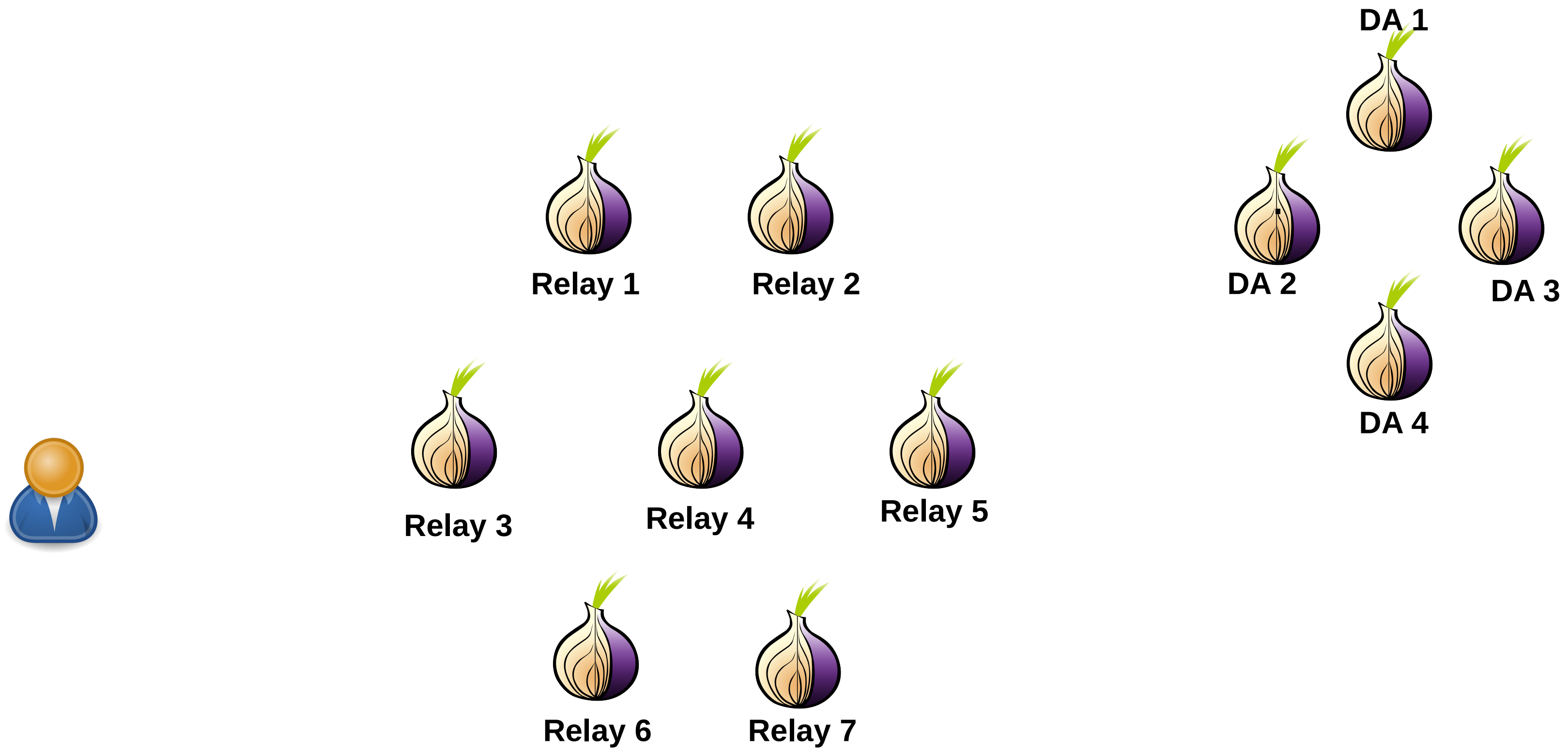
Background



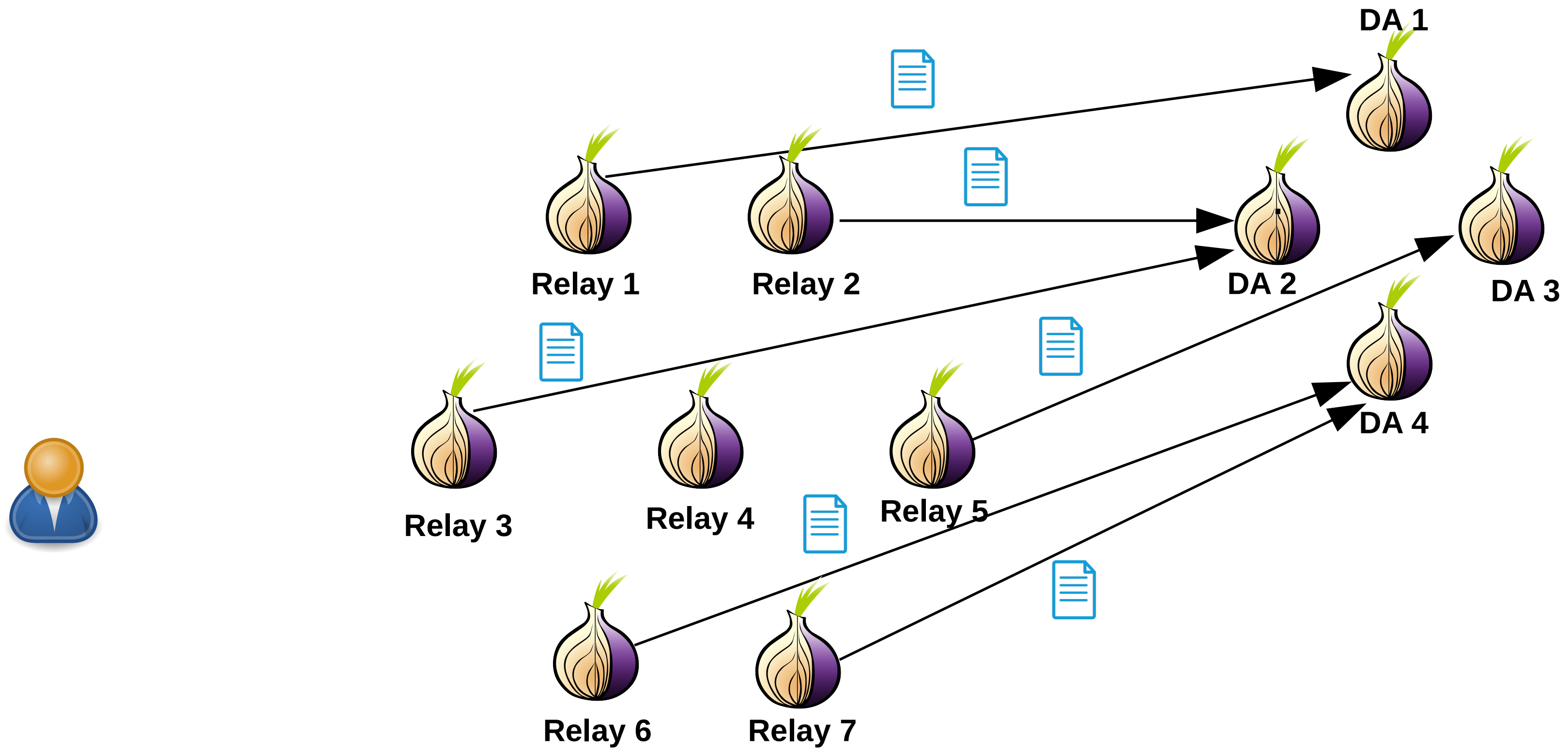
Background



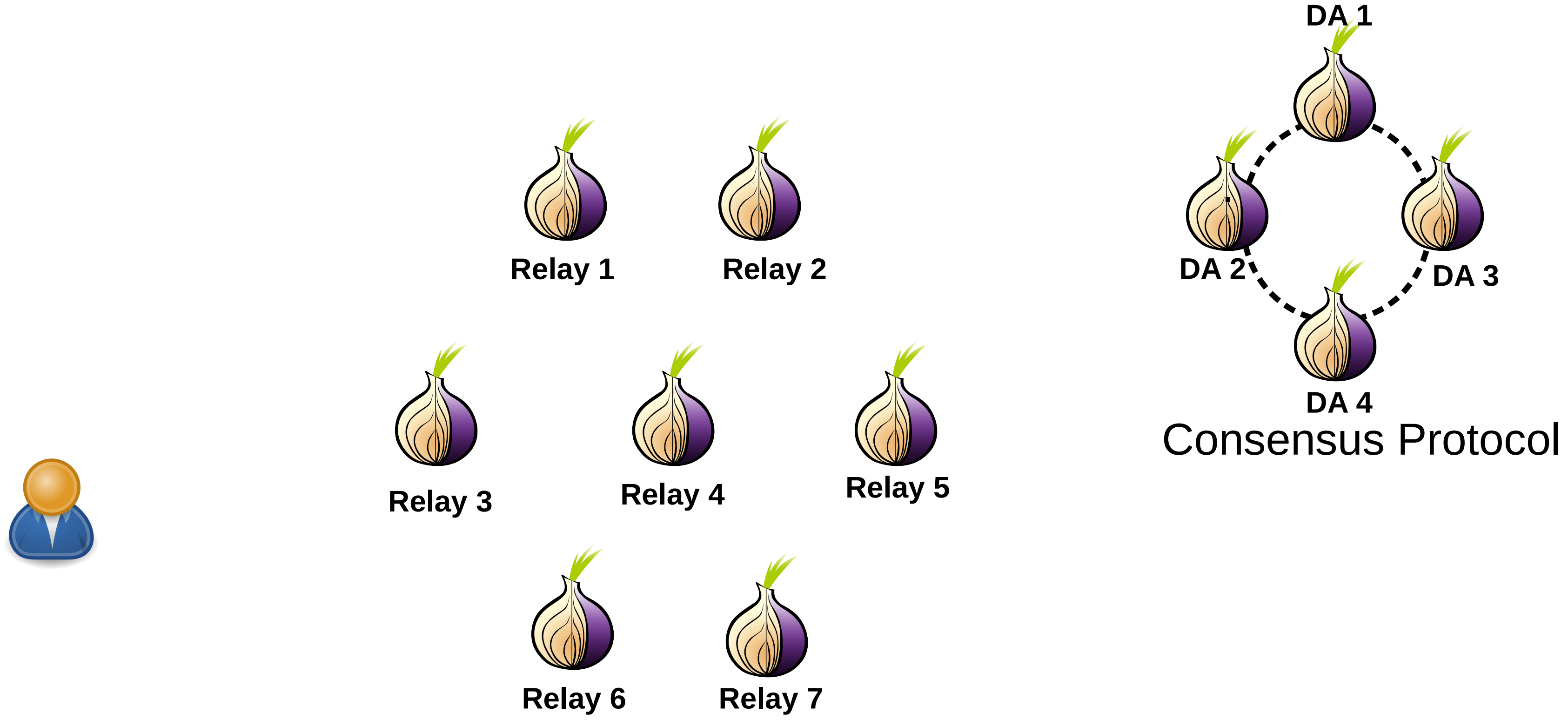
Background


DA 1
Relay 1
Relay 2
DA 2
DA 3
DA 4
Relay 3
Relay 4
Relay 5
Relay 6
Relay 7

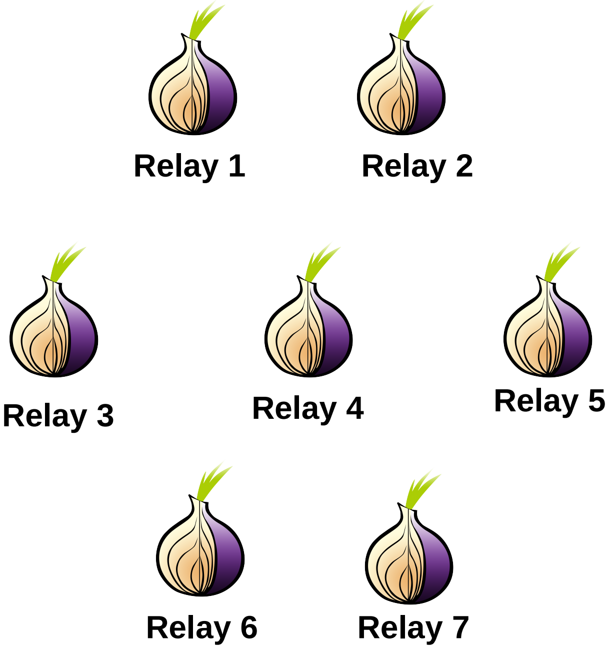
Background

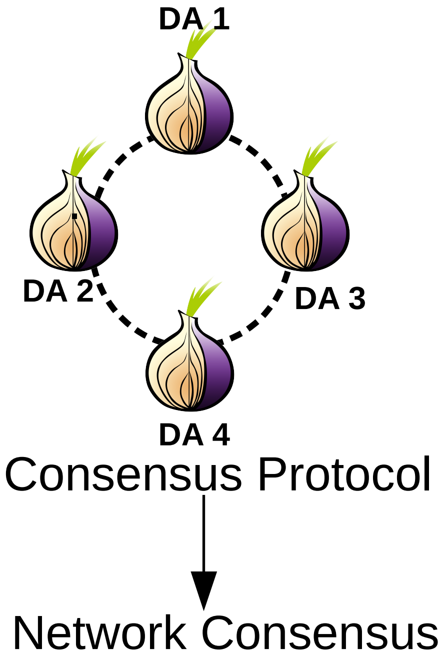

DA 1
Relay 1
Relay 2
DA 2
DA 3
Relay 3
Relay 4
Relay 5
DA 4
Relay 6
Relay 7

Background

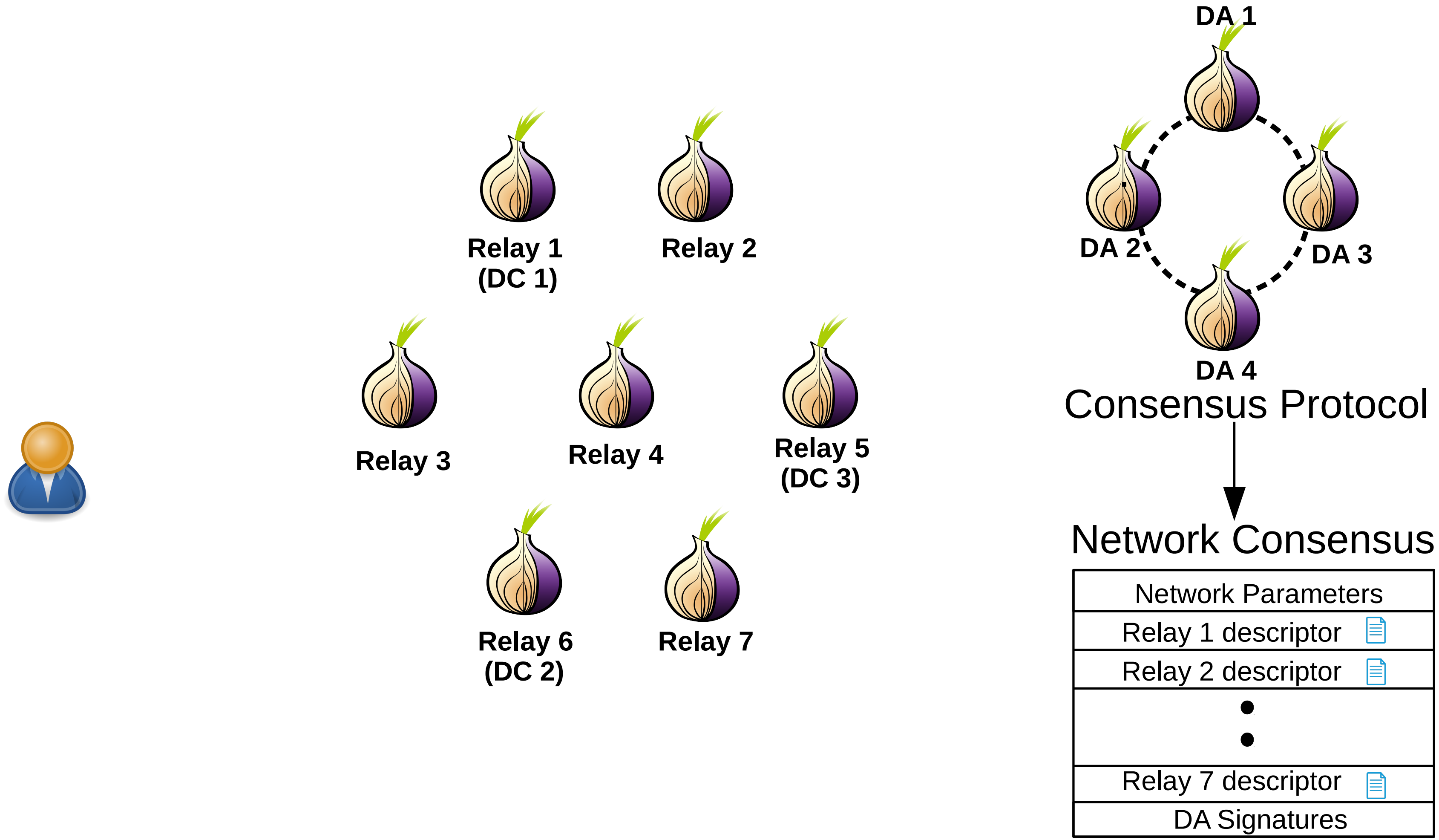

Relay 1
Relay 2
Relay 3
Relay 4
Relay 5
Relay 6
Relay 7
DA 1
DA 2
DA 3
DA 4
Consensus Protocol

Background


Relay 1
Relay 2
Relay 3
Relay 4
Relay 5
Relay 6
Relay 7

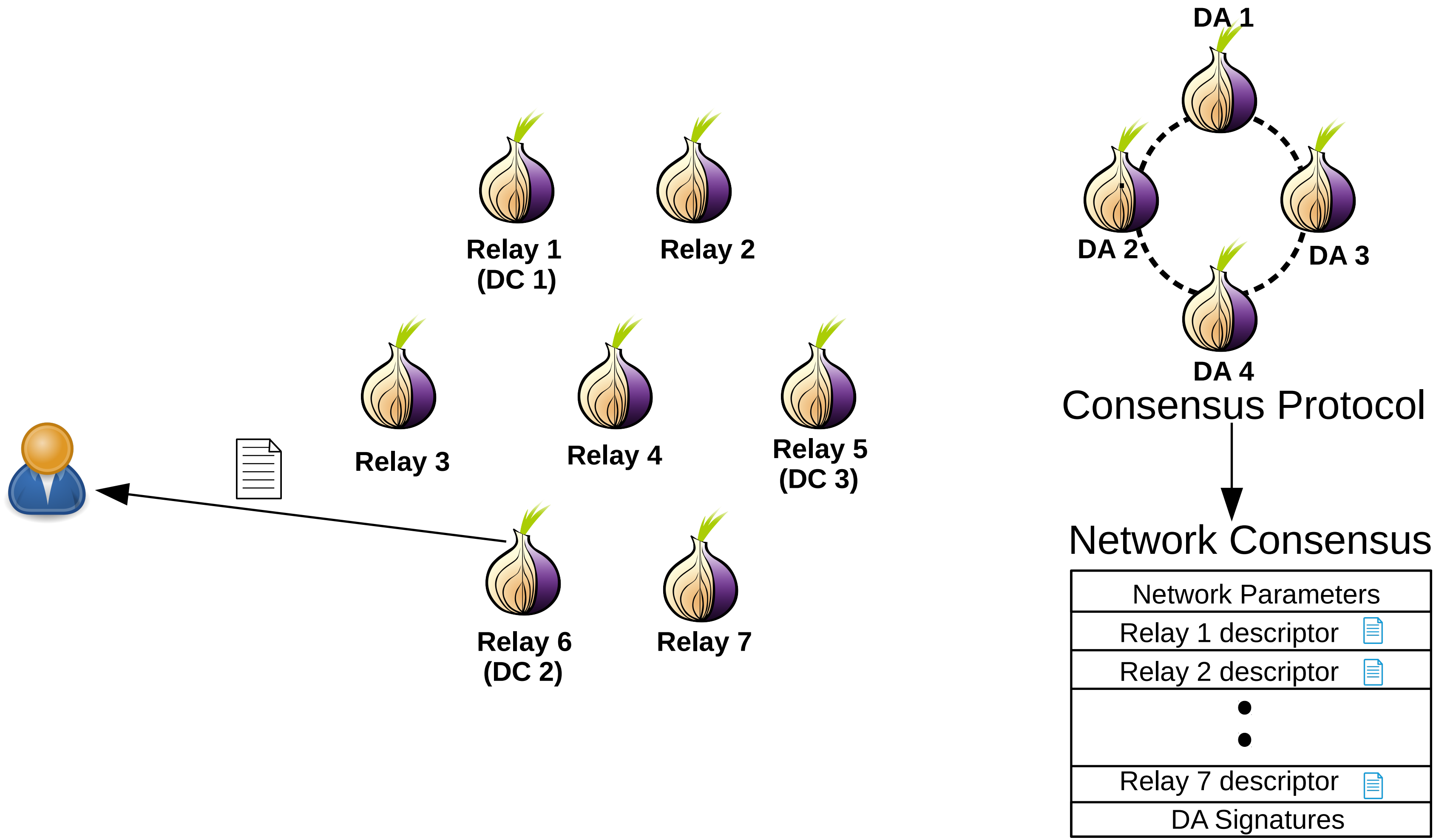

DA 1
DA 2
DA 3
DA 4
Consensus Protocol
Network Consensus

Background

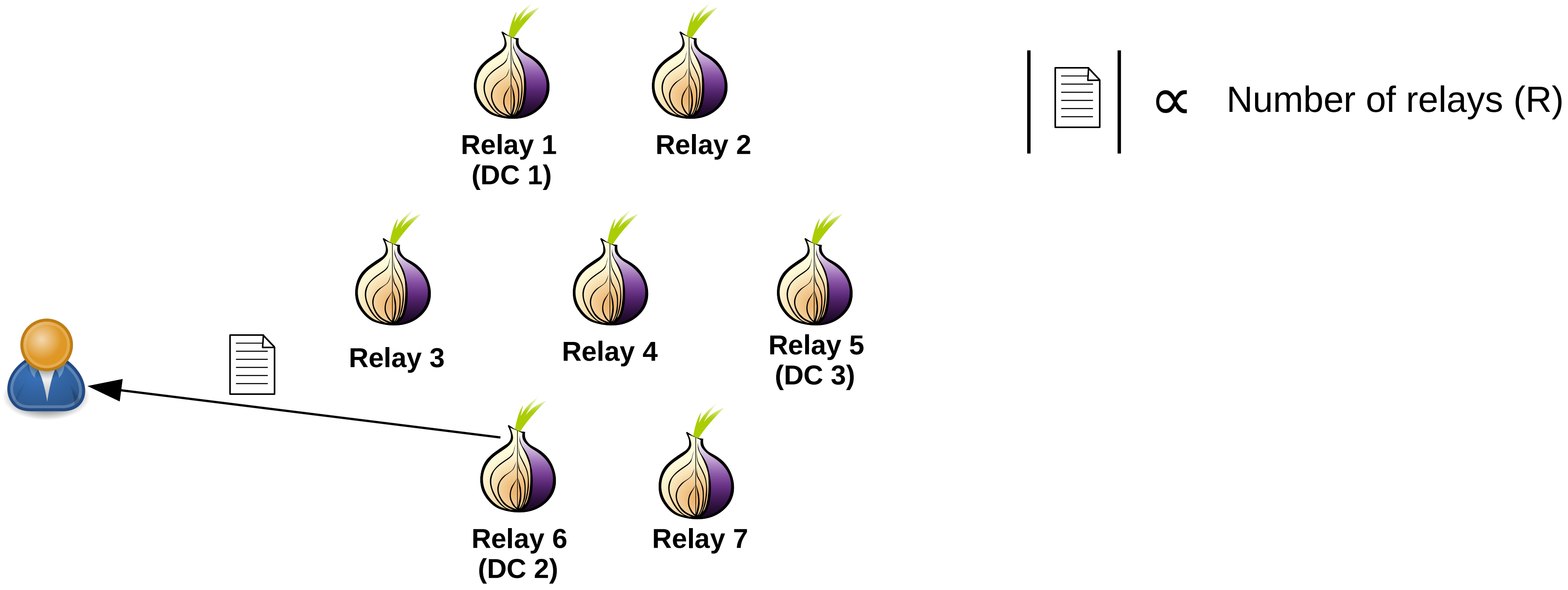


Relay 1
(DC 1)
Relay 2
Relay 3
Relay 4
Relay 5
(DC 3)
Relay 6
(DC 2)
Relay 7
DA 1
DA 2
DA 3
DA 4
Consensus Protocol
Network Consensus
Network Parameters
Relay 1 descriptor
Relay 2 descriptor
Relay 7 descriptor
DA Signatures

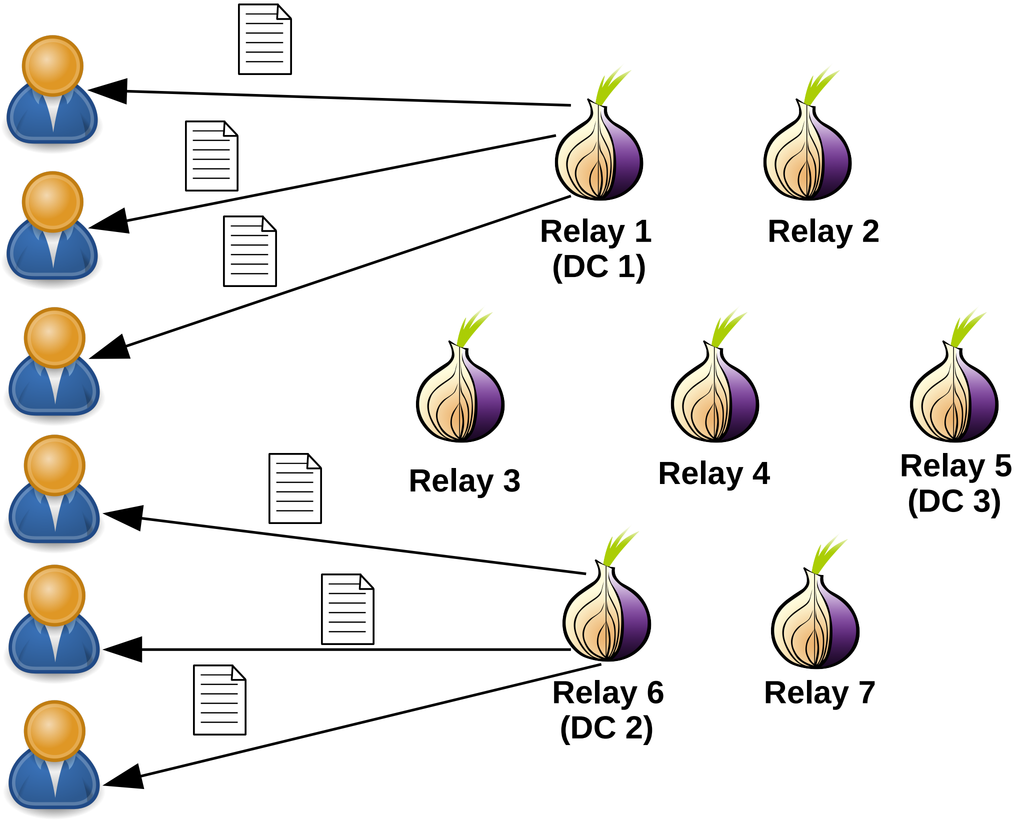
Background

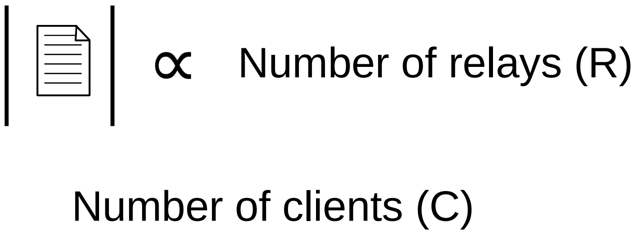

DA 1
DA 2
DA 3
DA 4
Relay 1
(DC 1)
Relay 2
Relay 3
Relay 4
Relay 5
(DC 3)
Relay 6
(DC 2)
Relay 7
Consensus Protocol
Network Consensus
Network Parameters
Relay 1 descriptor
Relay 2 descriptor
Relay 7 descriptor
DA Signatures

Background

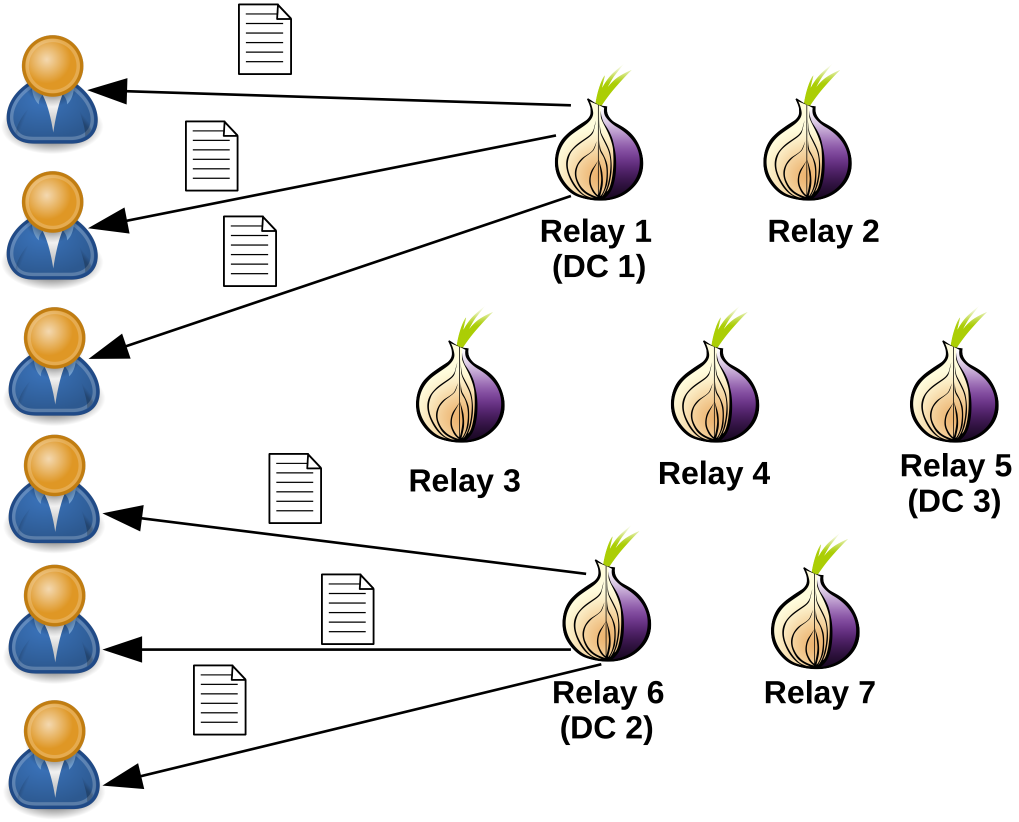

Relay 1
(DC 1)
Relay 2
Relay 3
Relay 4
Relay 5
(DC 3)
Relay 6
(DC 2)
Relay 7
∝ Number of relays (R)

Background


Relay 1
(DC 1)
Relay 2
Relay 3
Relay 4
Relay 5
(DC 3)
Relay 6
(DC 2)
Relay 7


∝ Number of relays (R)
Number of clients (C)

Background



|  | ∝ Number of relays (R)

Number of clients (C)

Total Bandwidth = R * C

Previous Work

1) Peer-to-peer Models:

Previous Work

1) Peer-to-peer Models:

All of which were shown to be susceptible to be different attack vectors.

Previous Work

1) Peer-to-peer Models:
All of which were shown to be susceptible to be different attack vectors.

2) Client-Server Models:
- PIR-Tor :

Previous Work

1) Peer-to-peer Models:
All of which were shown to be susceptible to be different attack vectors.

2) Client-Server Models:

- PIR-Tor :
  - Information-Theoretic PIR (ITPIR)
    Hard to deploy in practice due to non-colluding server assumptions

Previous Work

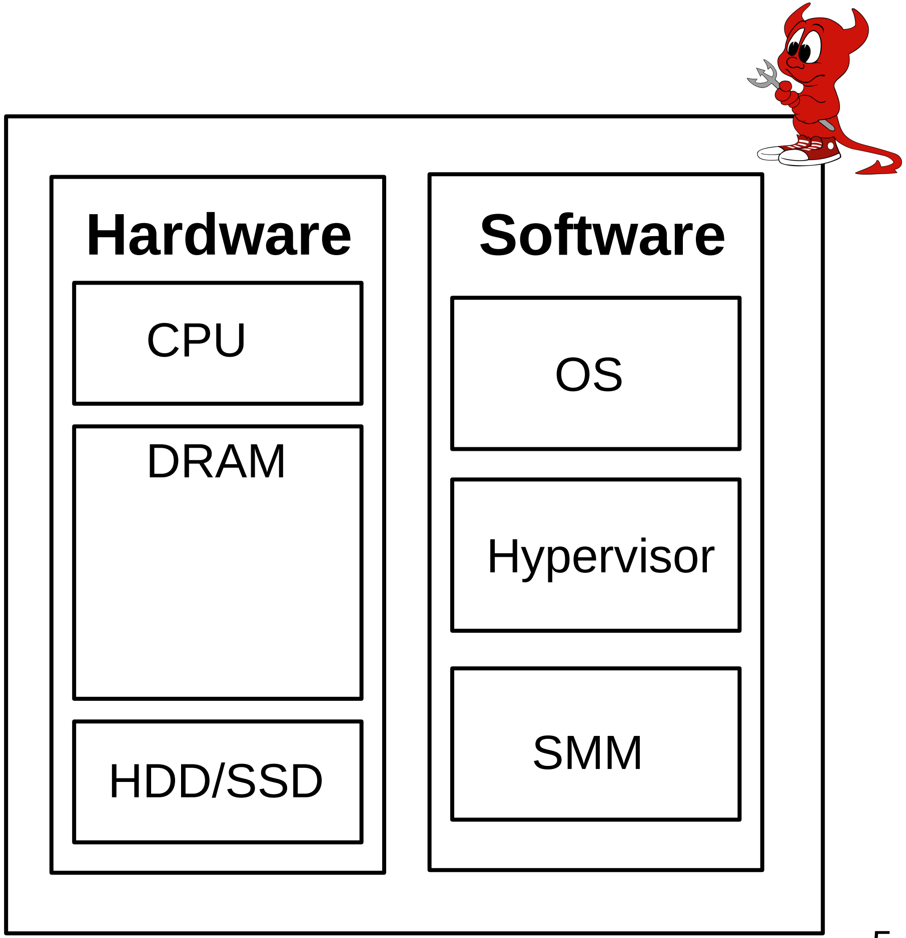
1) Peer-to-peer Models:
All of which were shown to be susceptible to be different attack vectors.

2) Client-Server Models:

- PIR-Tor :
  - Information-Theoretic PIR (ITPIR)
    Hard to deploy in practice due to non-colluding server assumptions
  - Computational PIR (CPIR)
    Uses “A Fast PIR Protocol” by Aguilar-Melchor and Gaborit, which was later shown to be vulnerable to lattice attacks from non-standard assumptions

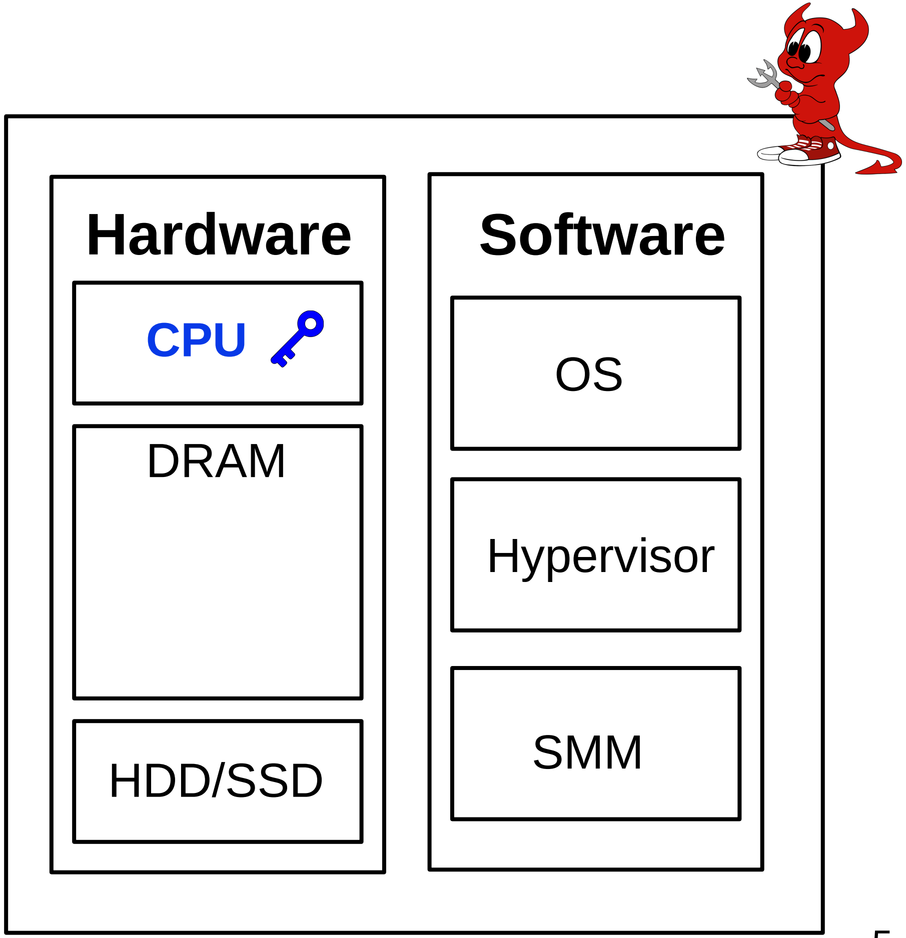
Trusted Execution Environments – Intel SGX

- Extension to the x86 instruction set



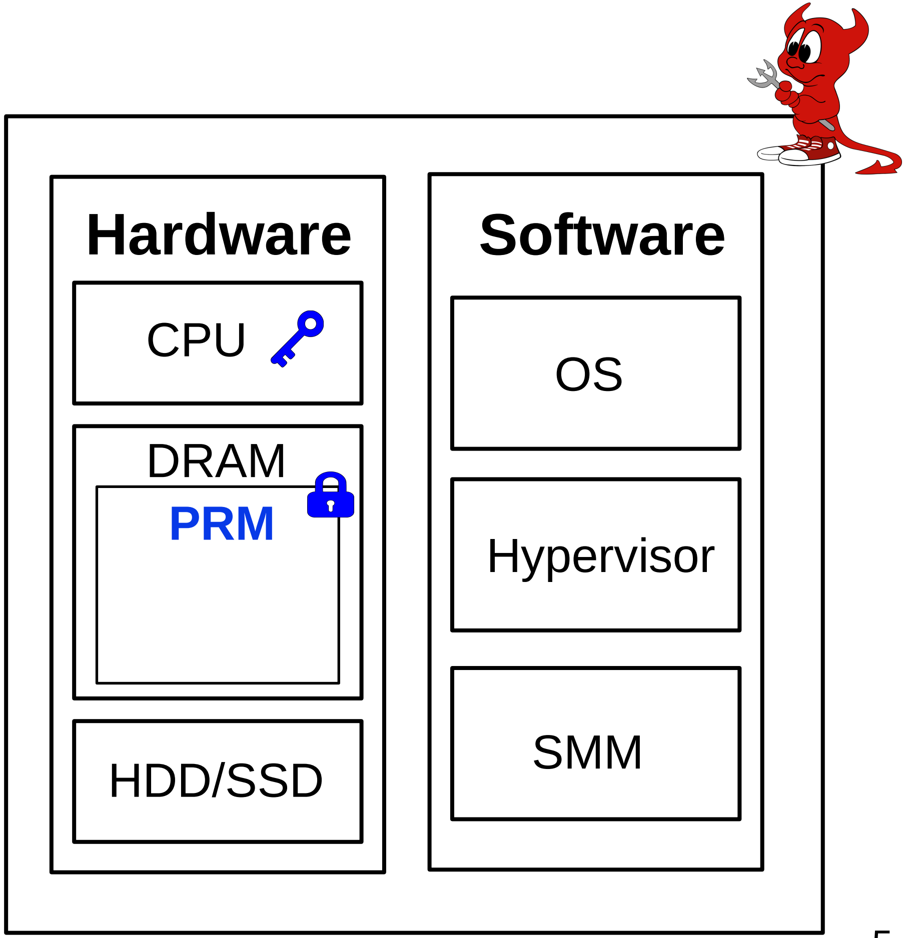
Trusted Execution Environments – Intel SGX

- Extension to the x86 instruction set
- Processor fused with secret keys at manufacture time



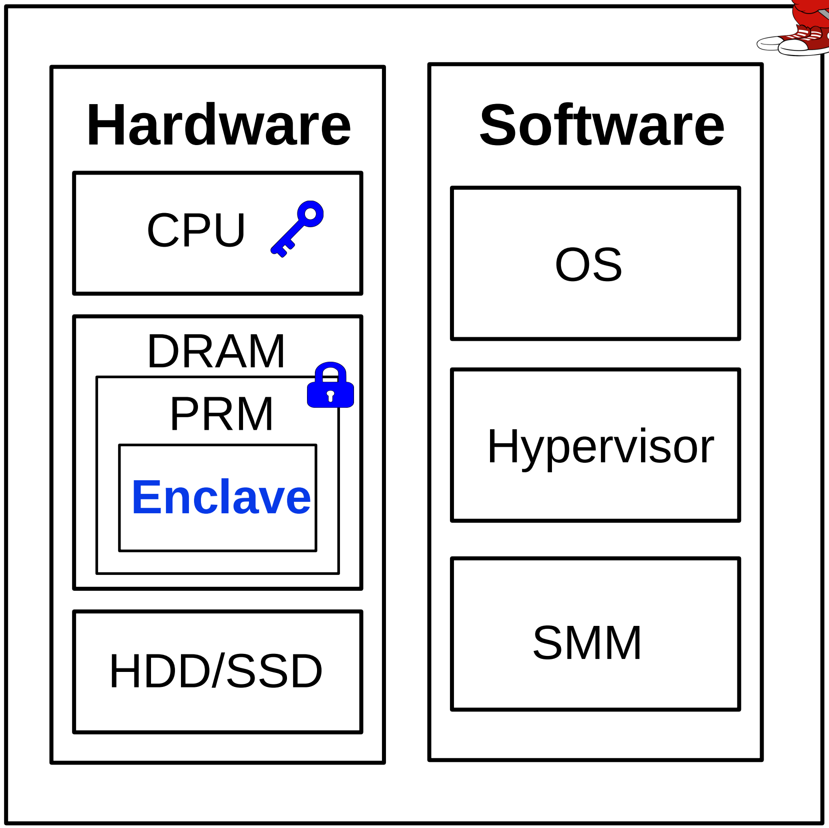
Trusted Execution Environments – Intel SGX

- Extension to the x86 instruction set
- Processor fused with secret keys at manufacture time
- Enabling the processor to set aside Processor Reserved Memory (**PRM**) at boot time



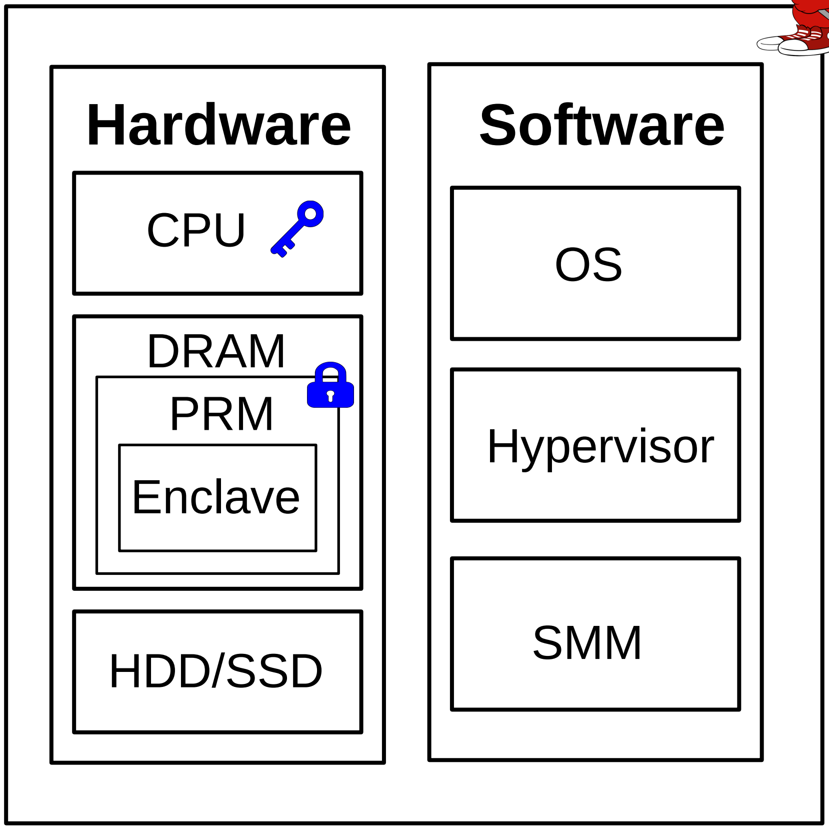
Trusted Execution Environments – Intel SGX

- Extension to the x86 instruction set
- Processor fused with secret keys at manufacture time
- Enabling the processor to set aside Processor Reserved Memory (**PRM**) at boot time
- Able to instantiante secure virtual containers called **enclaves**



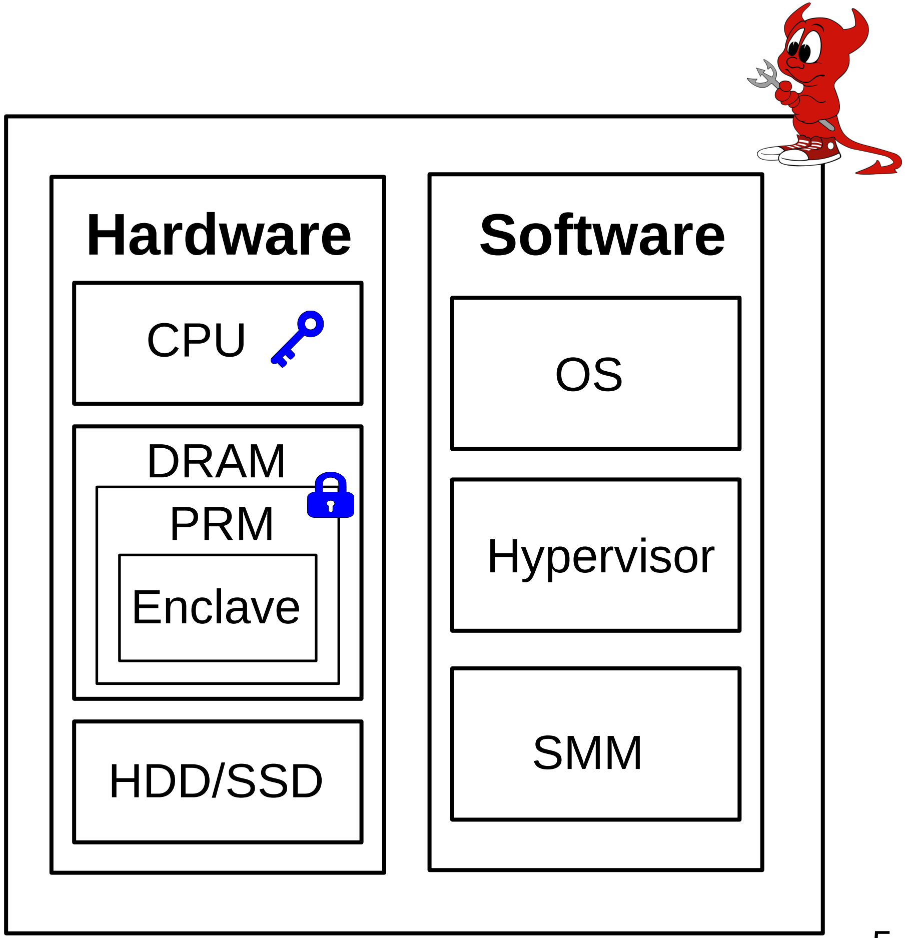
Trusted Execution Environments – Intel SGX

- Extension to the x86 instruction set
- Processor fused with secret keys at manufacture time
- Enabling the processor to set aside Processor Reserved Memory (**PRM**) at boot time
- Able to instantiante secure virtual containers called **enclaves**
- Enclaves can load programs with confidentiality, integrity and freshness guarantees



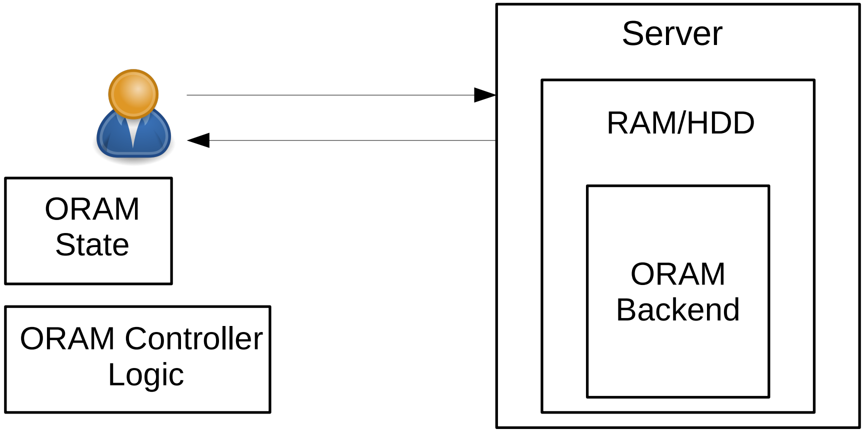
Trusted Execution Environments – Intel SGX

- Extension to the x86 instruction set
- Processor fused with secret keys at manufacture time
- Enabling the processor to set aside Processor Reserved Memory (**PRM**) at boot time
- Able to instantiante secure virtual containers called **enclaves**
- Enclaves can load programs with confidentiality, integrity and freshness guarantees
- Remote Attestation support for users to verify integrity of programs running in an enclave



ORAMs

- Oblivious RAM (ORAM) enables memory accesses with indistinguishable access patterns

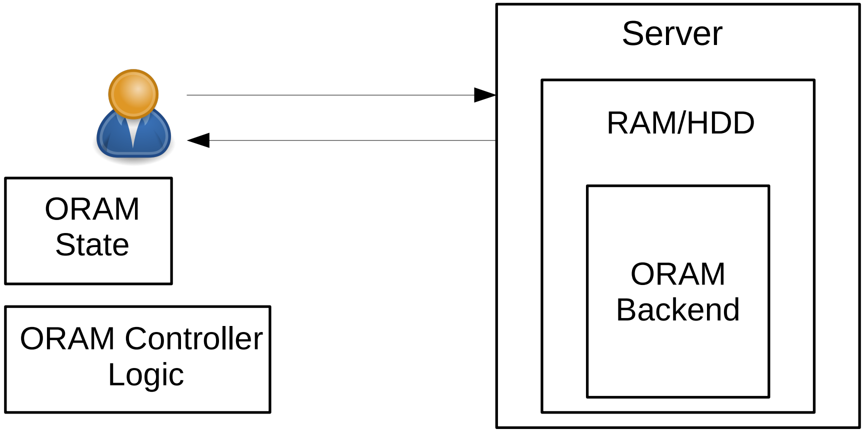


Native ORAM

ORAMs

- Oblivious RAM (ORAM) enables memory accesses with indistinguishable access patterns

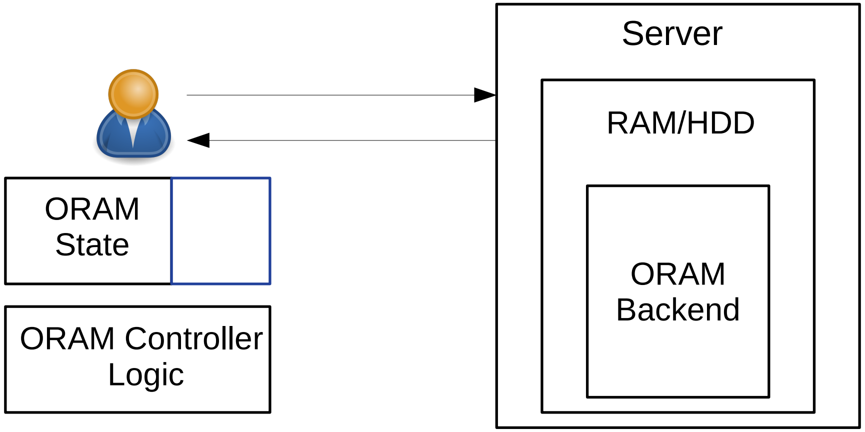
- Typically ORAMs require client-side computations and



Native ORAM

ORAMs

- Oblivious RAM (ORAM) enables memory accesses with indistinguishable access patterns
- Typically ORAMs require client-side computations and
  - large client-side memory

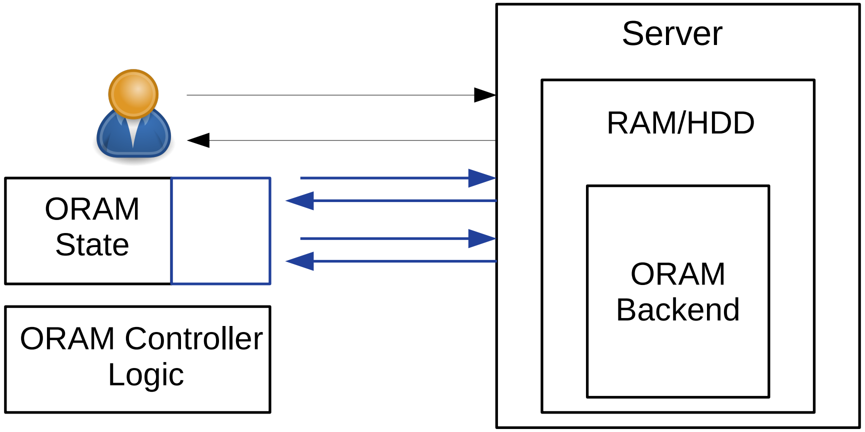


Native ORAM

ORAMs

- Oblivious RAM (ORAM) enables memory accesses with indistinguishable access patterns

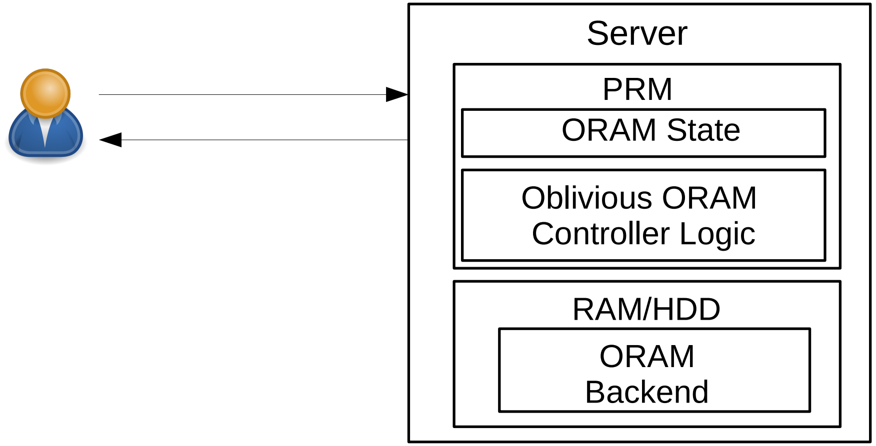
- Typically ORAMs require client-side computations and
  - large client-side memory OR
  - small client-side memory but multiple network roundtrips



Native ORAM

Doubly Oblivious ORAMs

- With the recent TEE advancements, ORAMs have become practically viable

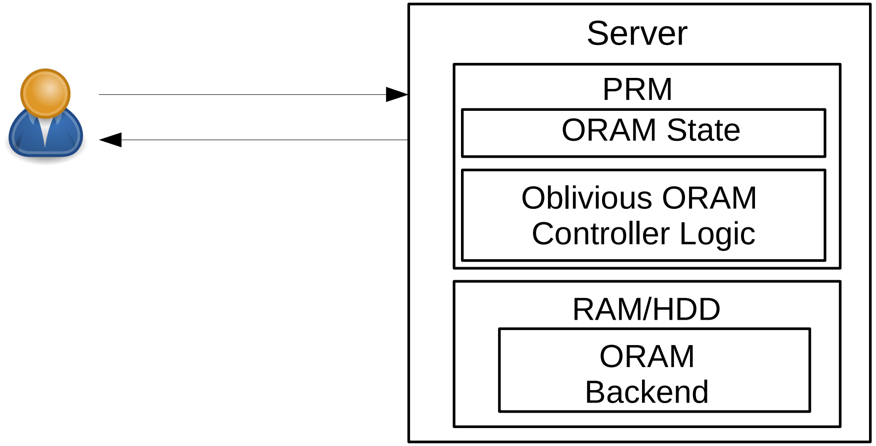


Doubly Oblivious ORAMs (TEE-Supported)

Doubly Oblivious ORAMs

- With the recent TEE advancements, ORAMs have become practically viable

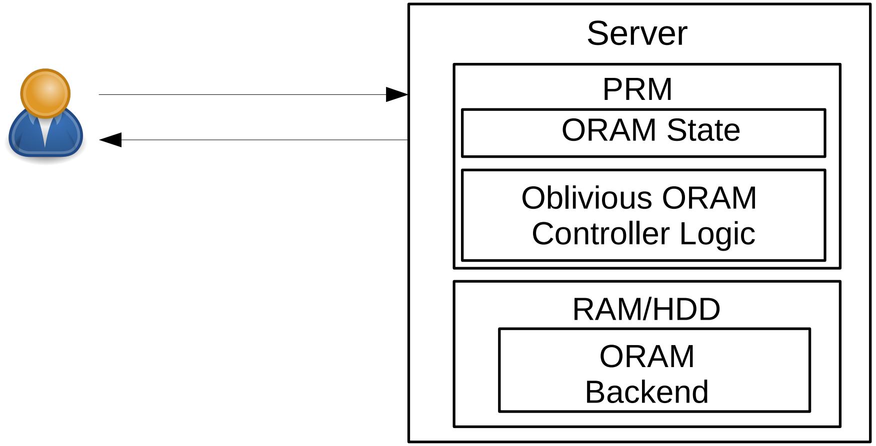
- The ORAM controller logic and state is moved into an enclave on the server side



Doubly Oblivious ORAMs (TEE-Supported)

Doubly Oblivious ORAMs

- With the recent TEE advancements, ORAMs have become practically viable
- The ORAM controller logic and state is moved into an enclave on the server side
- The ORAM controller logic itself is rewritten in an oblivious style to avoid known side-channel attacks against these TEE



Doubly Oblivious ORAMs (TEE-Supported)

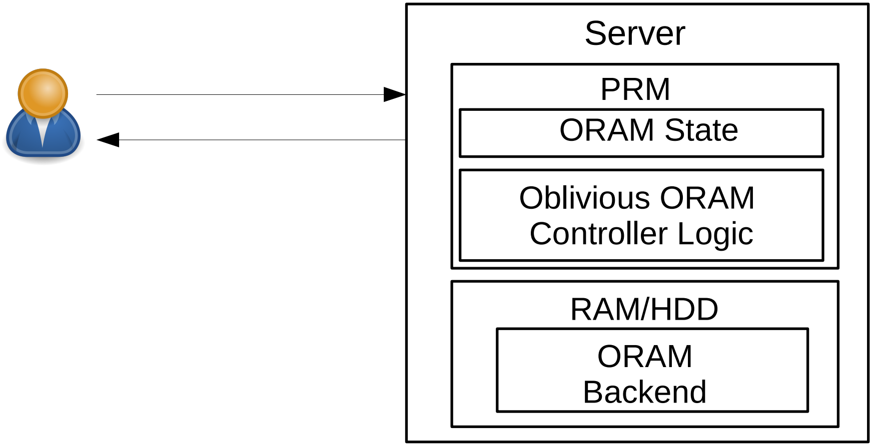
Doubly Oblivious ORAMs

- With the recent TEE advancements, ORAMs have become practically viable

- The ORAM controller logic and state is moved into an enclave on the server side

- The ORAM controller logic itself is rewritten in an oblivious style to avoid known side-channel attacks against these TEE

- Additionally, this also trivially enables single client ORAM protocols to support multiple clients



Doubly Oblivious ORAMs (TEE-Supported)

ConsenSGX

Use TEE-aided ORAM to retrieve a subset of the relay descriptors of a consensus obliviously so that the client can build circuits while not opening up to epistemic attacks

Our solution is:

- Efficient
- Scalable
- Incrementally deployable

ConsenSGX

Deploying an ORAM/PIR scheme has several other challenges in practice

- Indexing relay descriptors
- Selecting optimal block size for these schemes
- Compressing the overhead of individually signing descriptors
- Bootstrapping the scheme

Indexing descriptors

- Files/memory is accessed by an index in ORAM/PIR schemes

Indexing descriptors

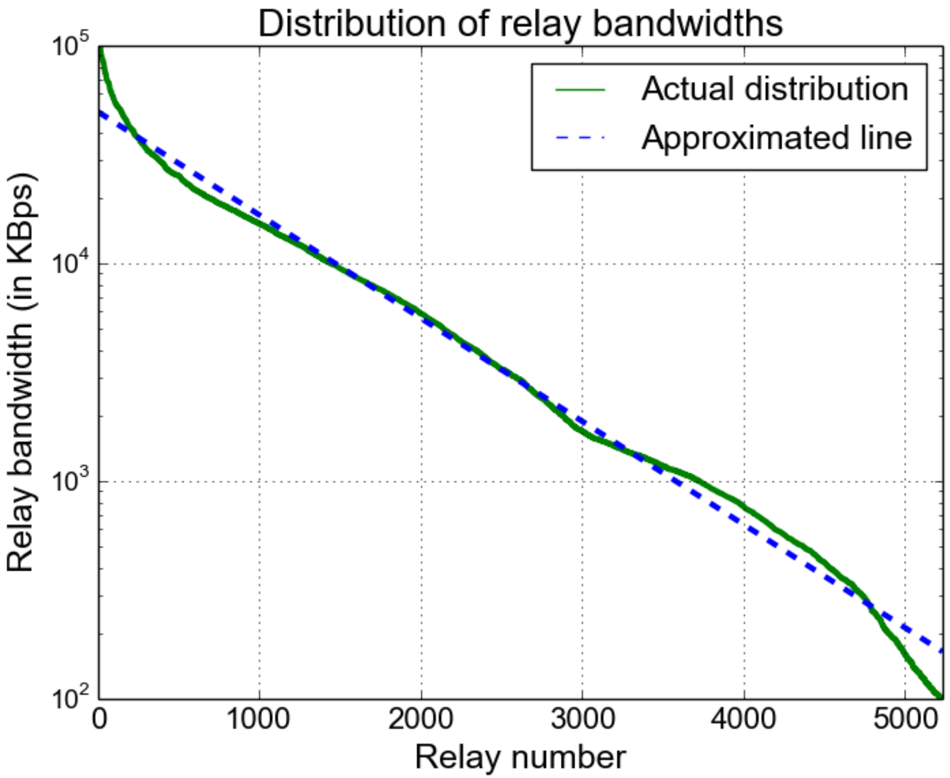
- Files/memory is accessed by an index in ORAM/PIR schemes
- Clients select relays to build a circuit by a bandwidth-weighted sampling mechanism from the full network consensus

Indexing descriptors

- Files/memory is accessed by an index in ORAM/PIR schemes
- Clients select relays to build a circuit by a bandwidth-weighted sampling mechanism from the full network consensus
- Use bandwidth ordering to generate indices for relays in an epoch.

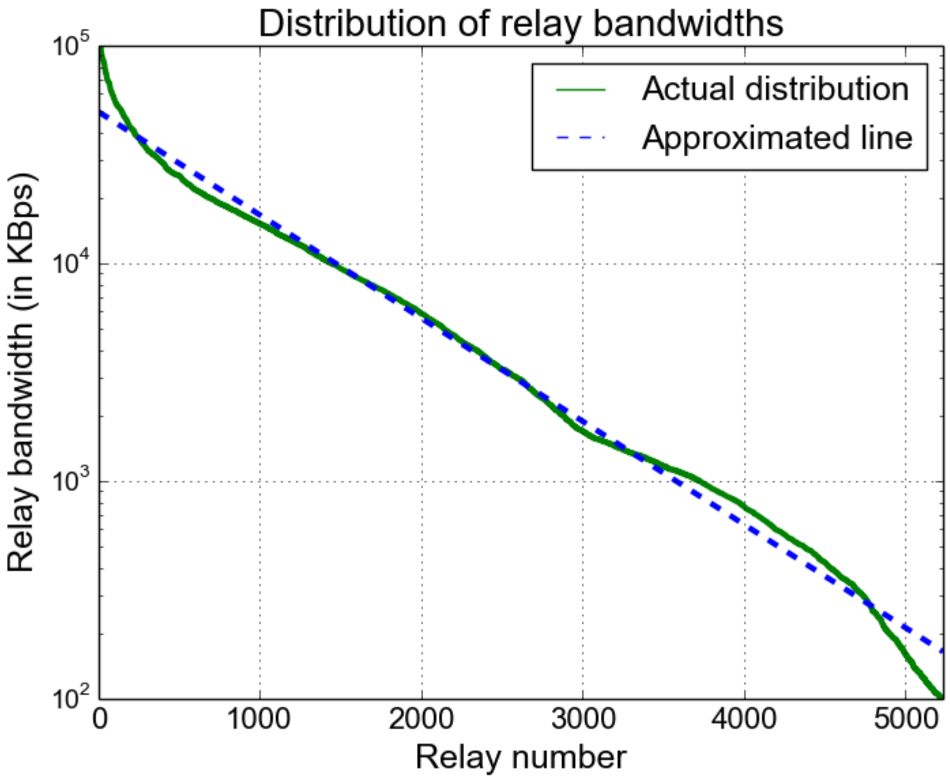
Indexing descriptors

- Files/memory is accessed by an index in ORAM/PIR schemes
- Clients select relays to build a circuit by a bandwidth-weighted sampling mechanism from the full network consensus
- Use bandwidth ordering to generate indices for relays in an epoch.



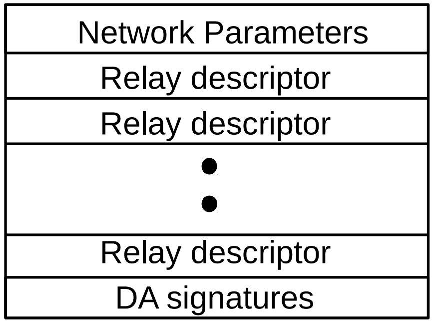
Indexing descriptors

- Files/memory is accessed by an index in ORAM/PIR schemes
- Clients select relays to build a circuit by a bandwidth-weighted sampling mechanism from the full network consensus
- Use bandwidth ordering to generate indices for relays in an epoch.
- The entire bandwidth ditribution of relays can be captured by the slope and intercept of this approximated line.



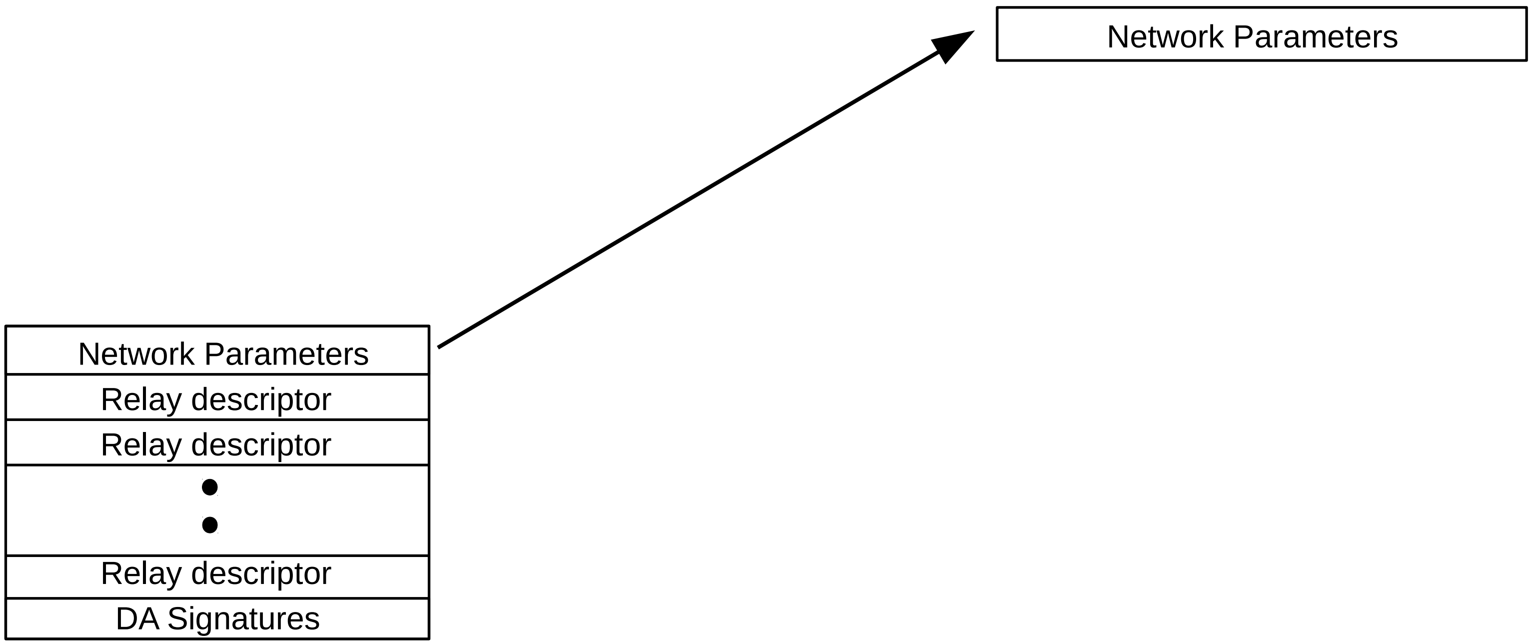
New consensus documents

Two new types of consensus documents:



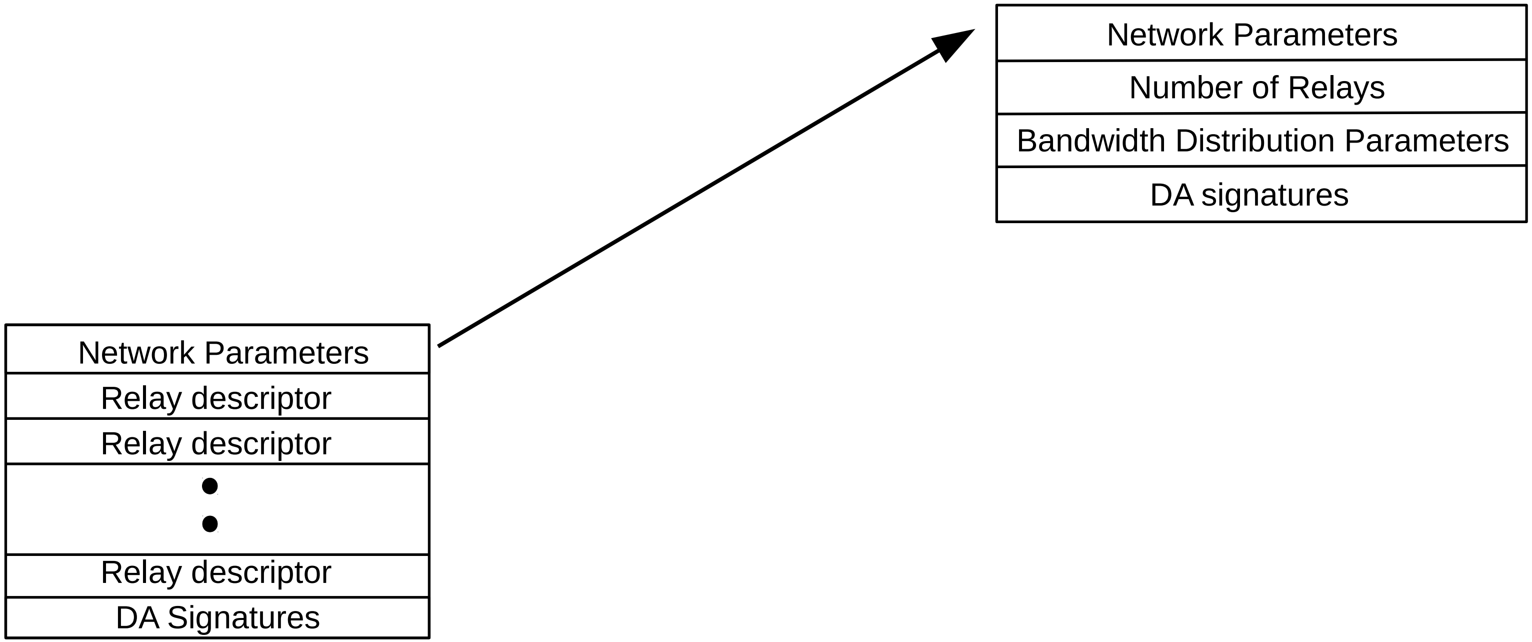
New consensus documents

Two new types of consensus documents:



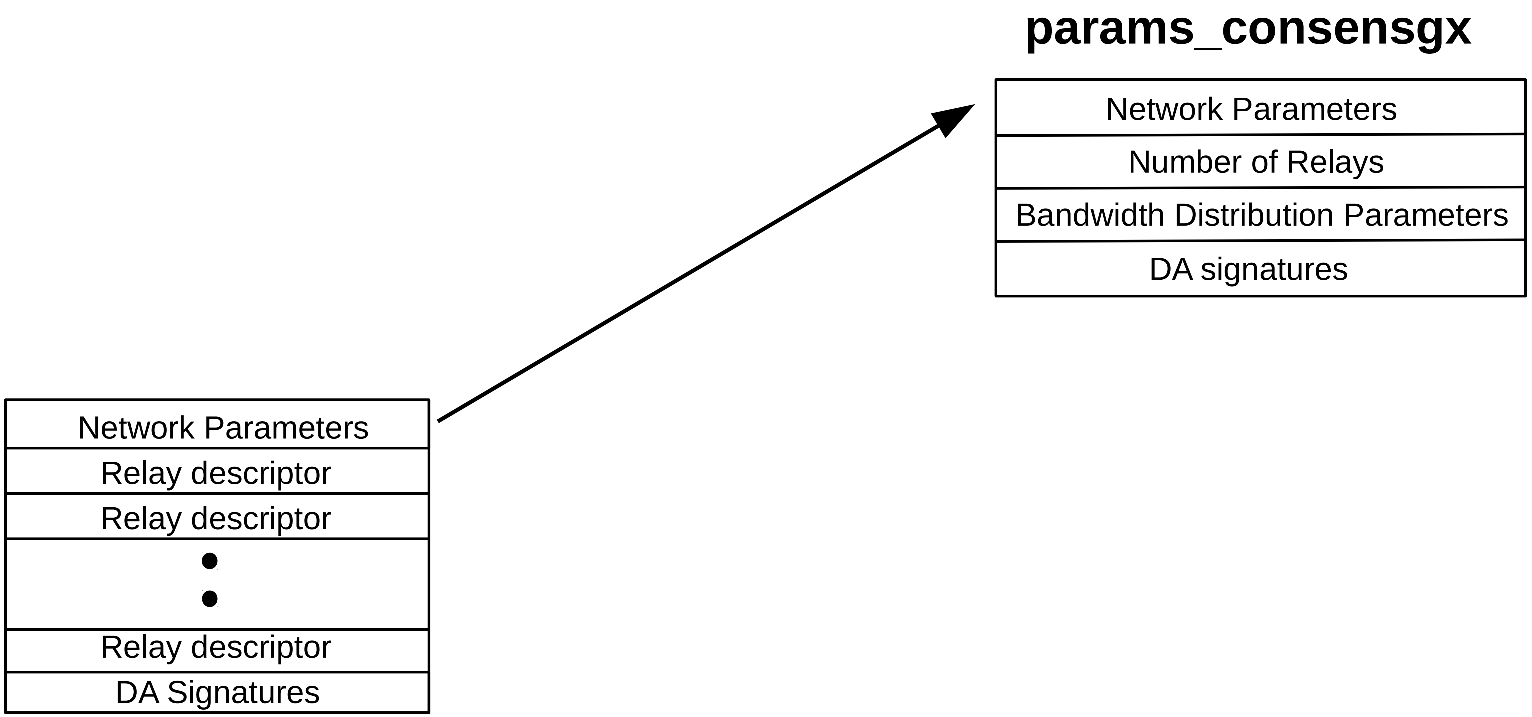
New consensus documents

Two new types of consensus documents:



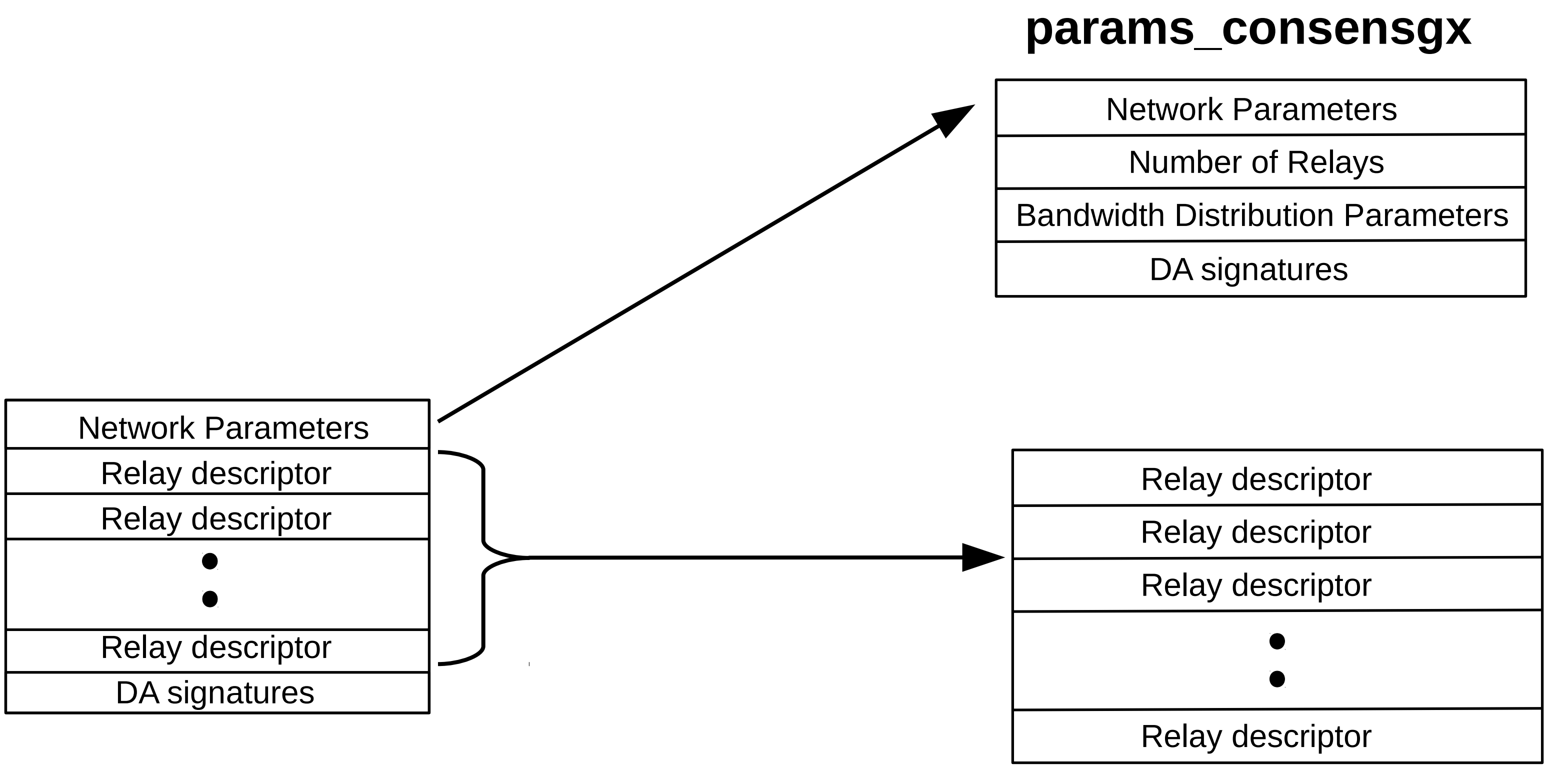
New consensus documents

Two new types of consensus documents:



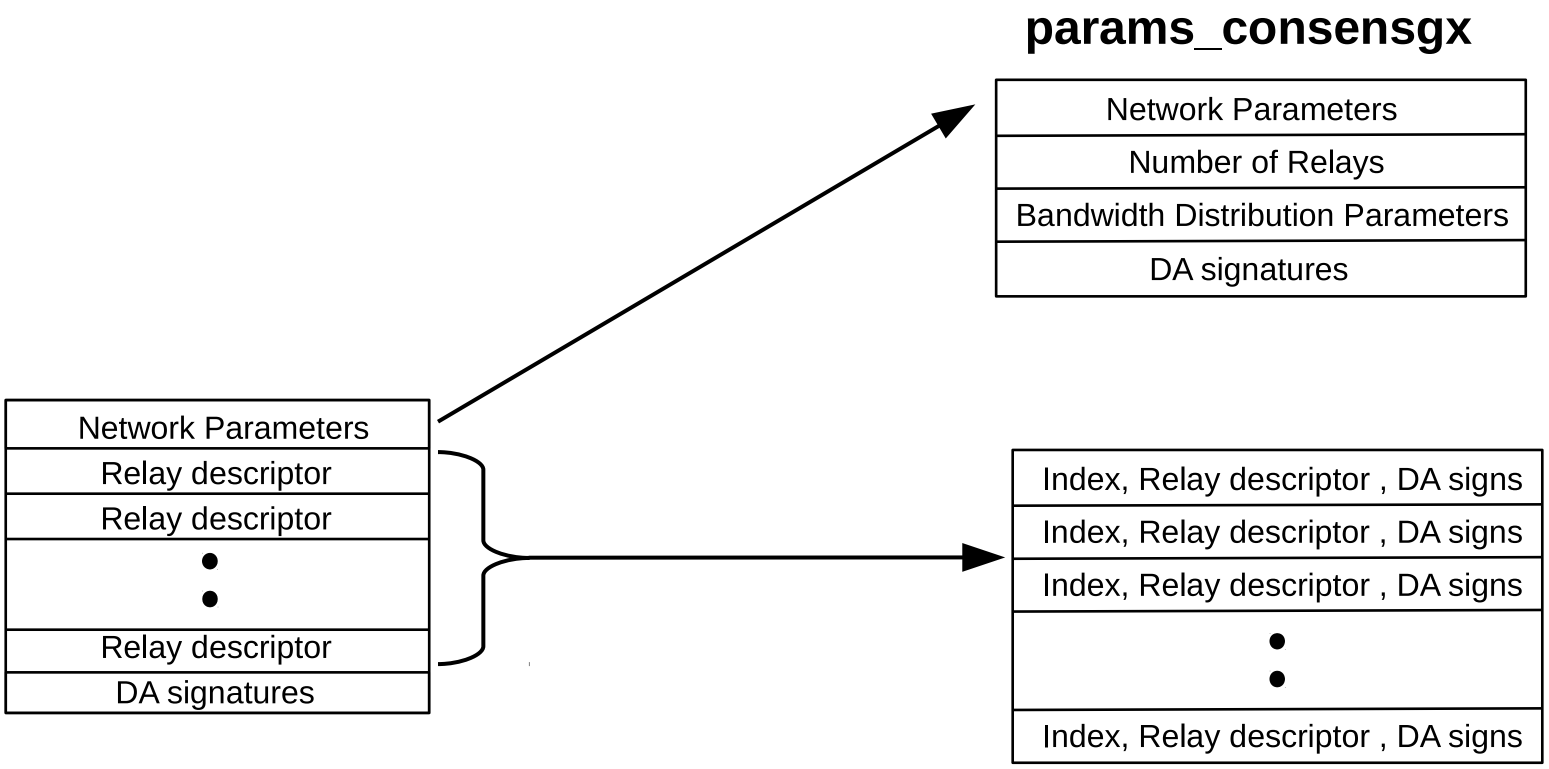
New consensus documents

Two new types of consensus documents:



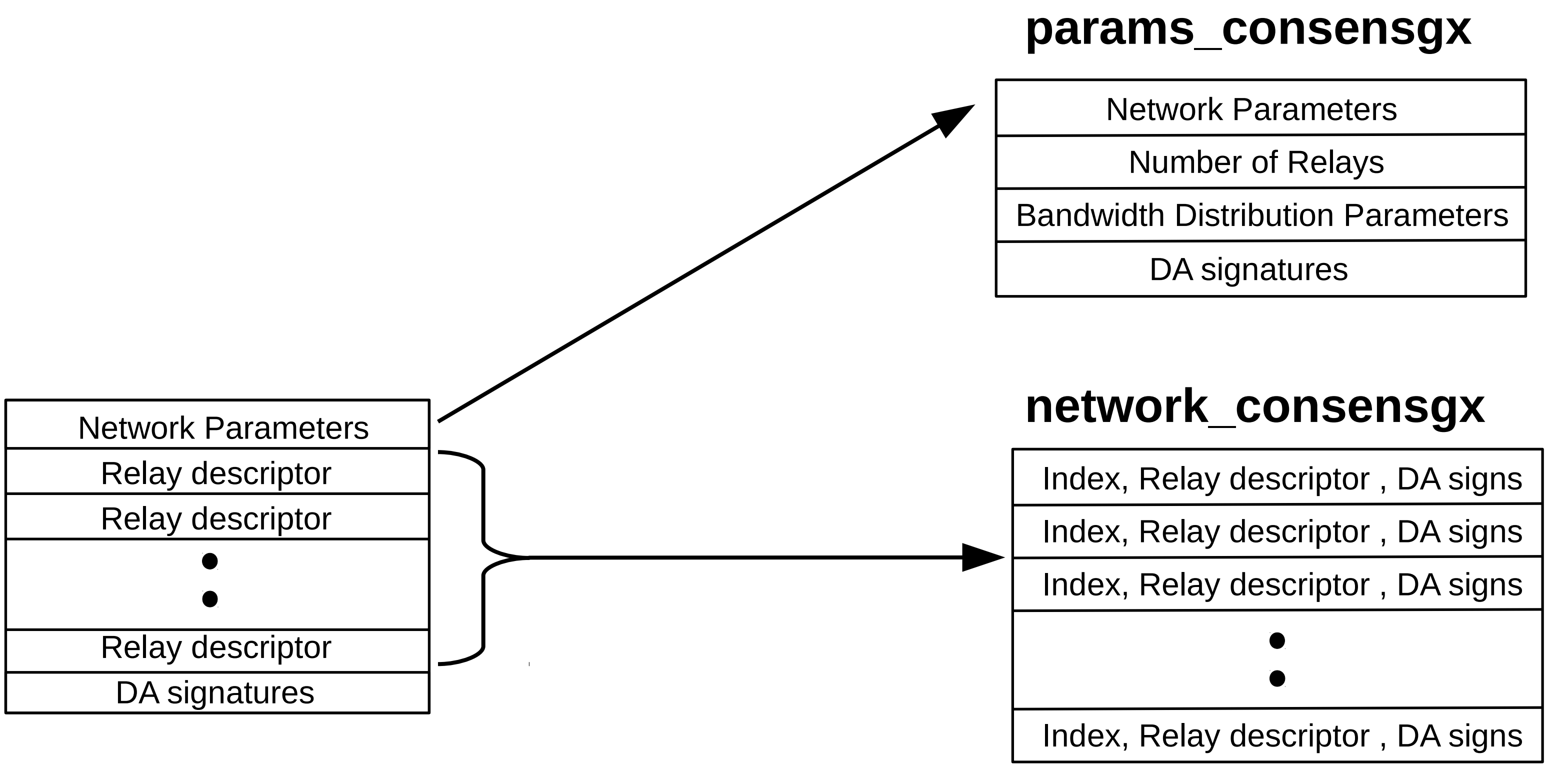
New consensus documents

Two new types of consensus documents:

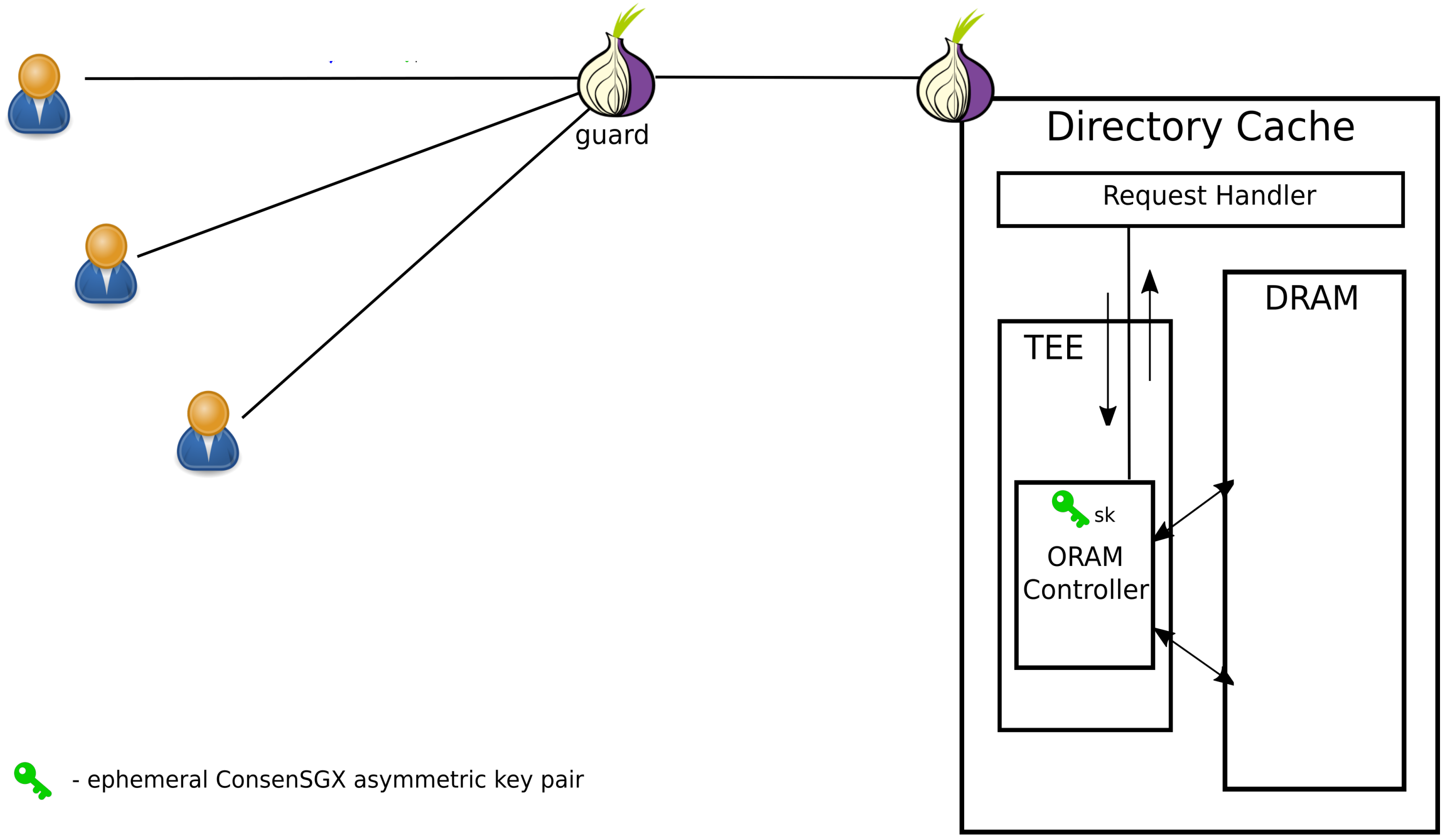


New consensus documents

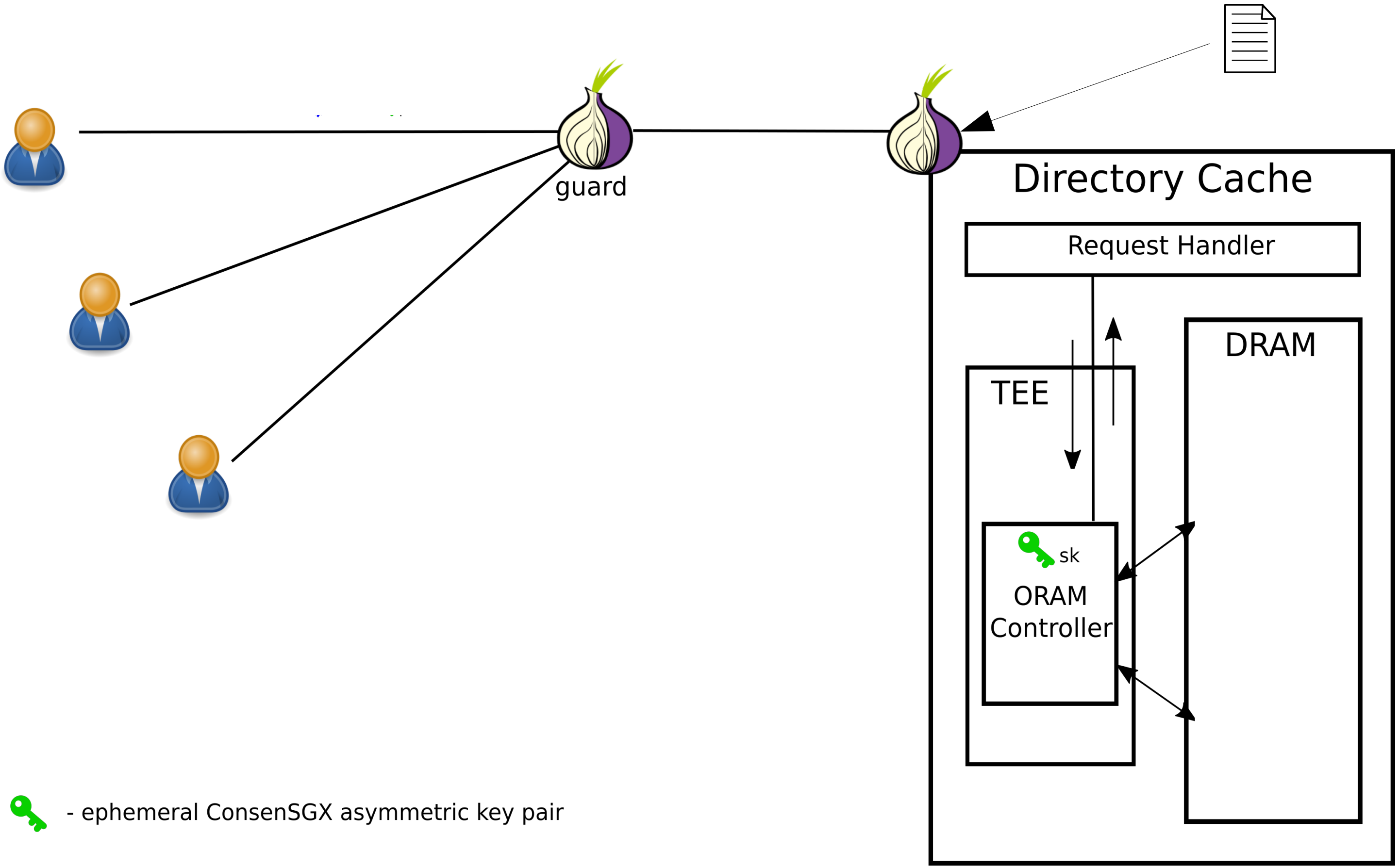
Two new types of consensus documents:



ConsenSGX Architecture

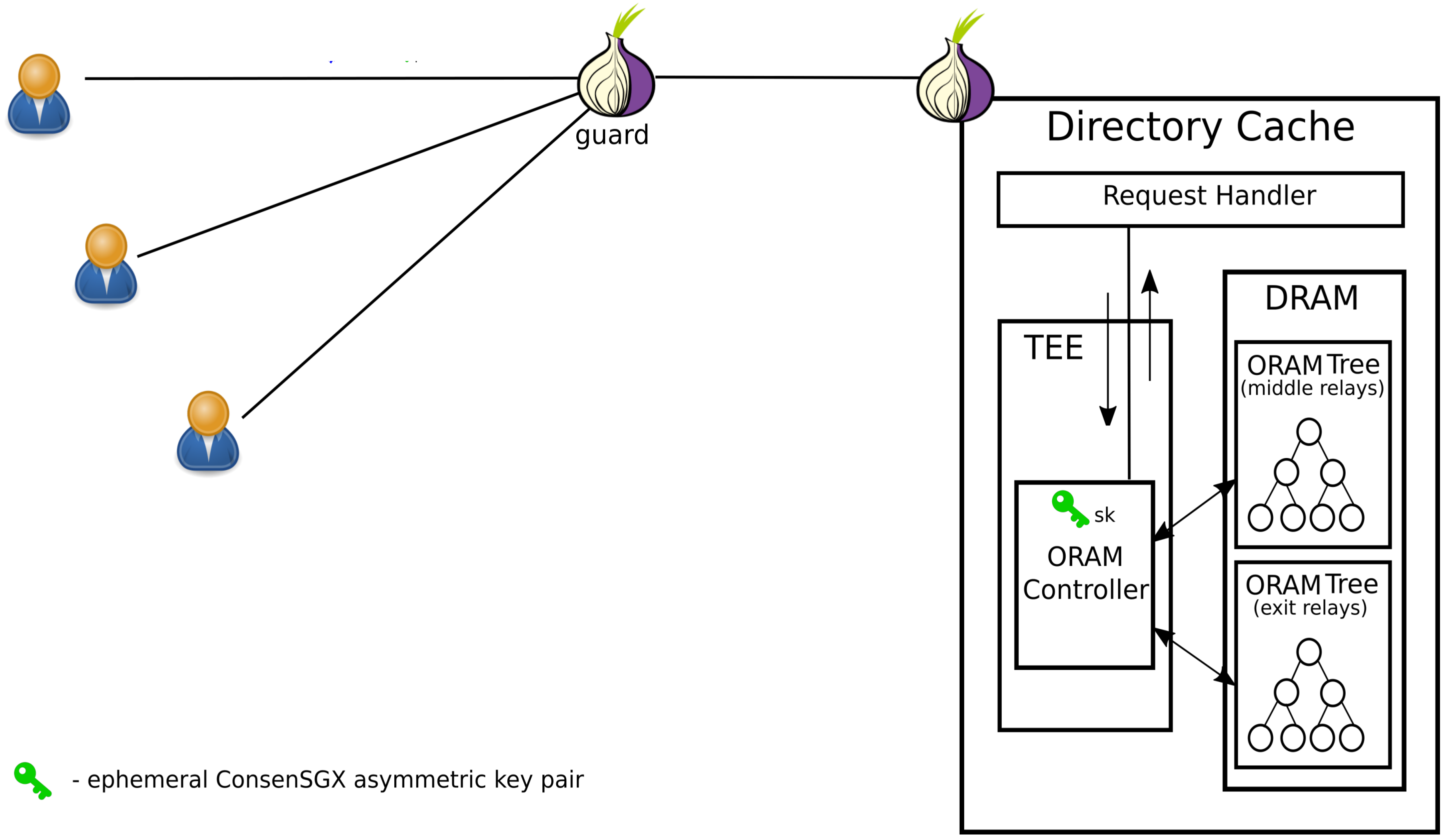


ConsenSGX Architecture

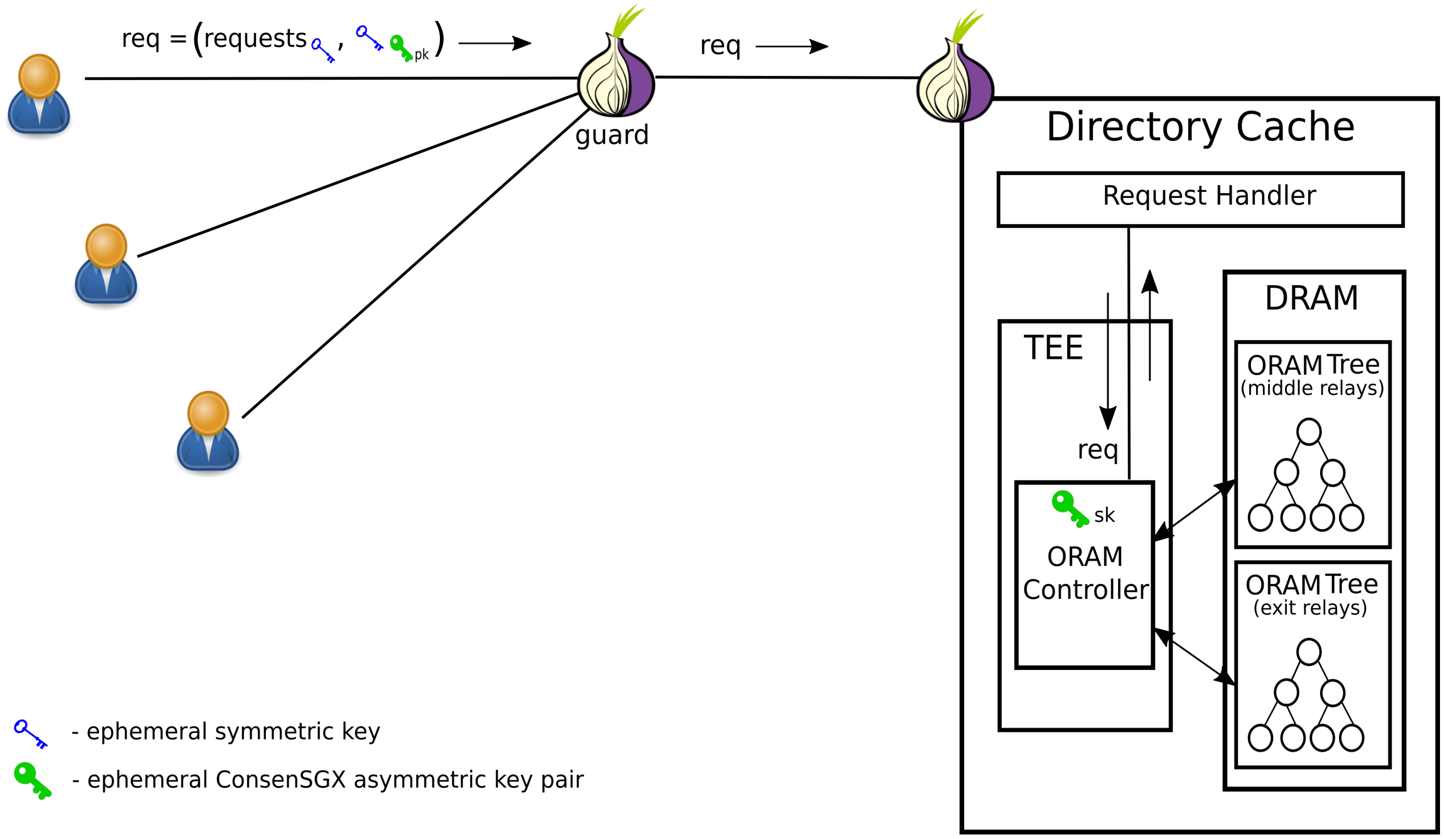


- ephemeral ConsenSGX asymmetric key pair

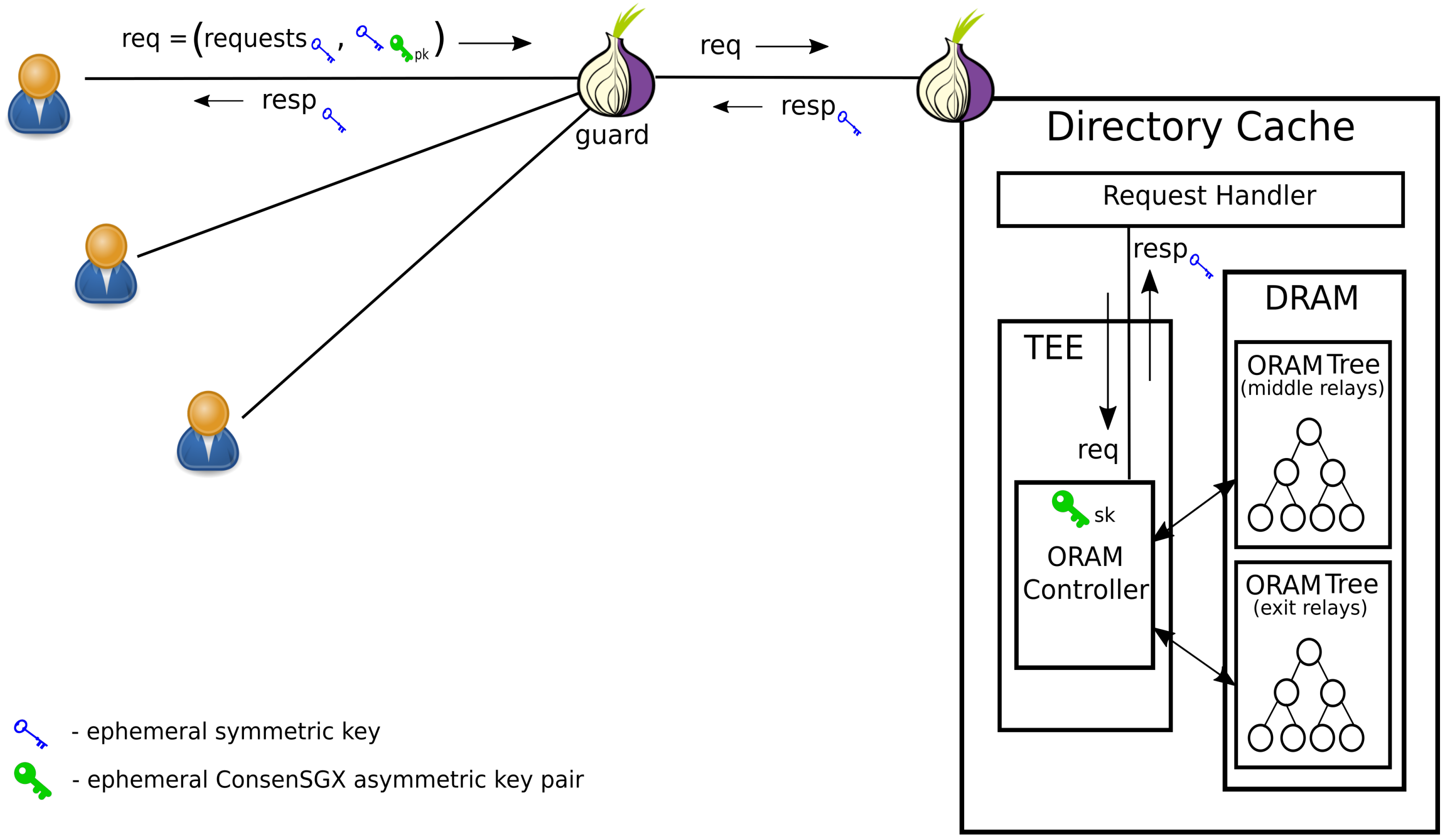
ConsenSGX Architecture



ConsenSGX Architecture



ConsenSGX Architecture



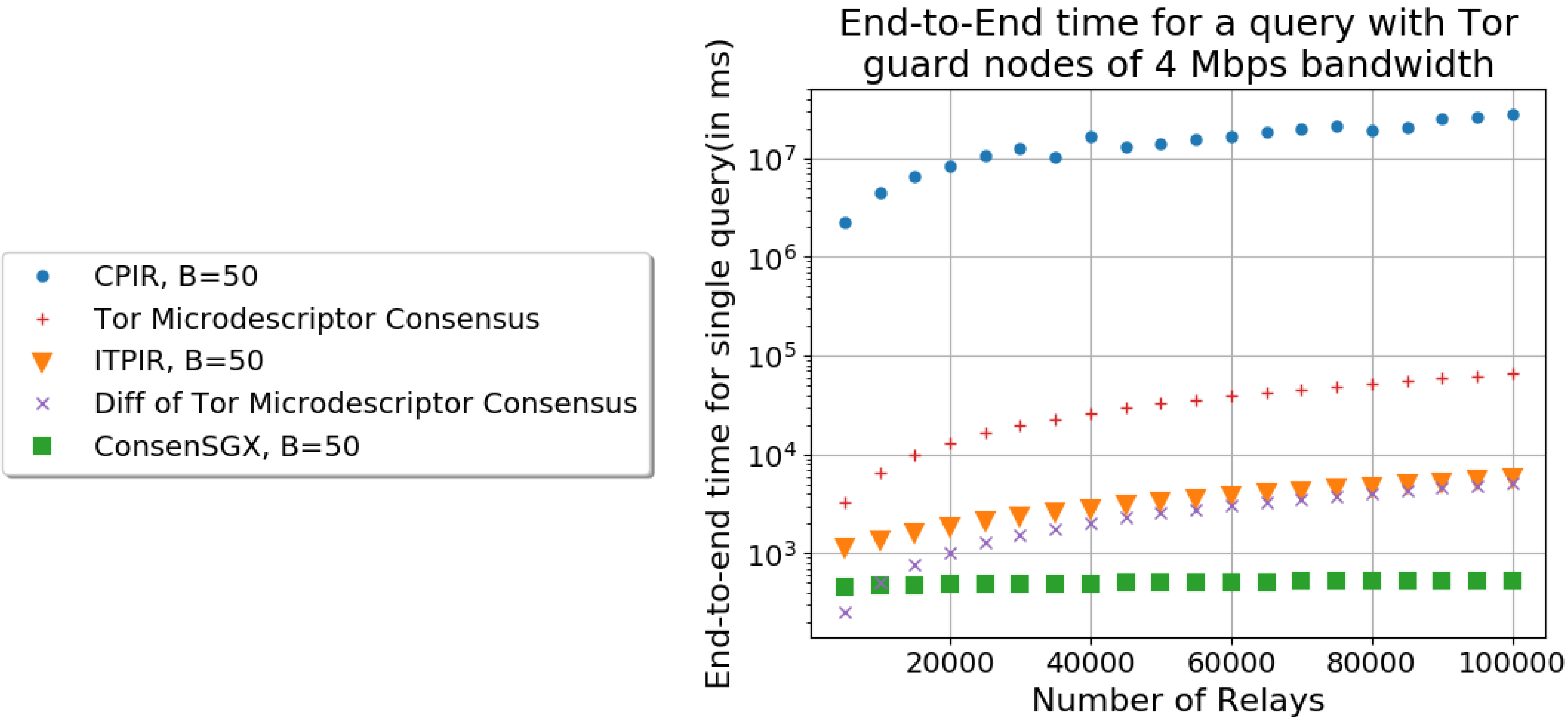
Evaluation

Evaluate ConsenSGX against :

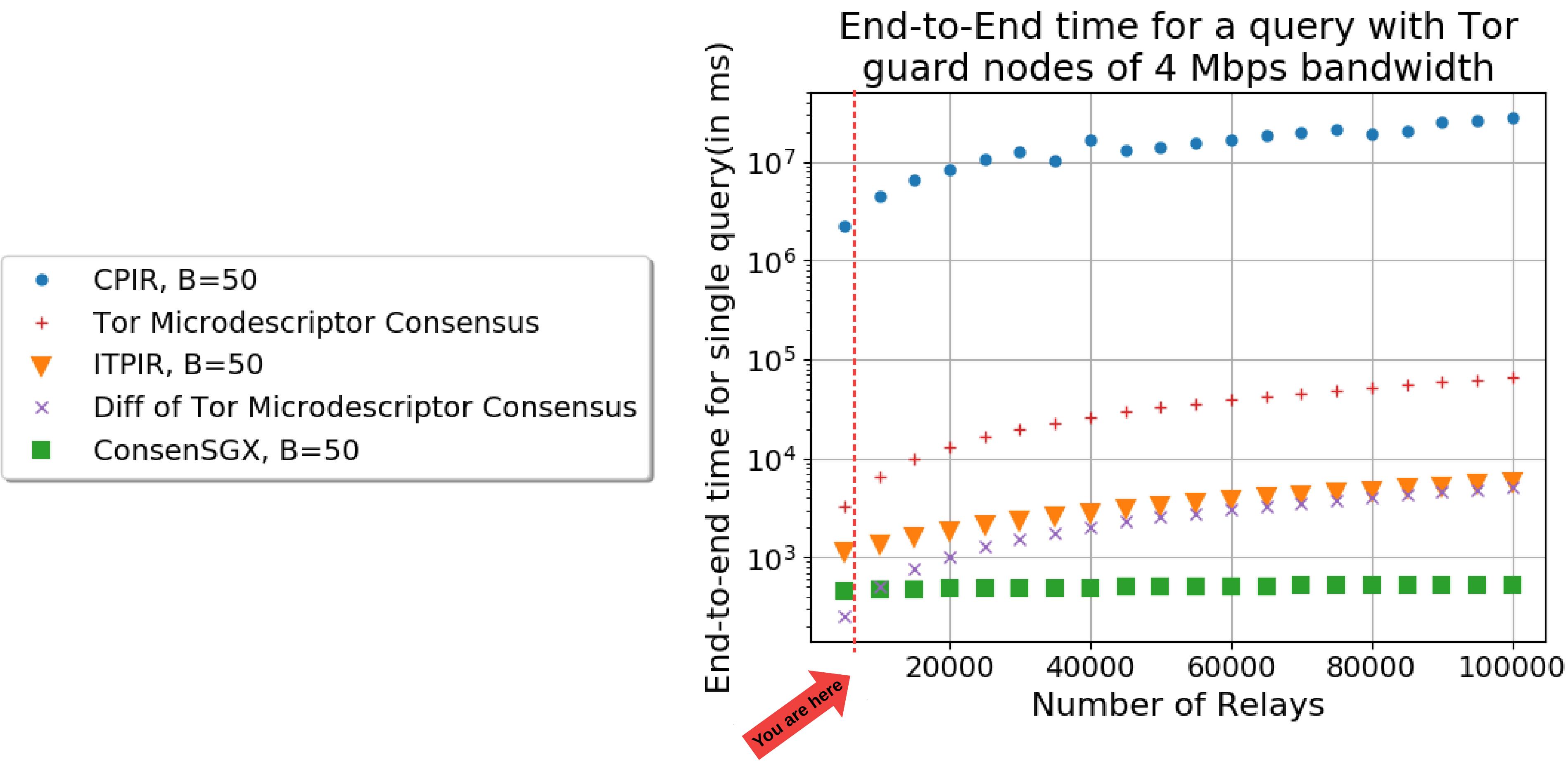
1) CPIR (XPIR)
2) ITPIR (Chor)
3) Microdescriptor consensus model
4) Diff variant of Microdescriptor consensus

B = batch_size or number of descriptors fetched in a request

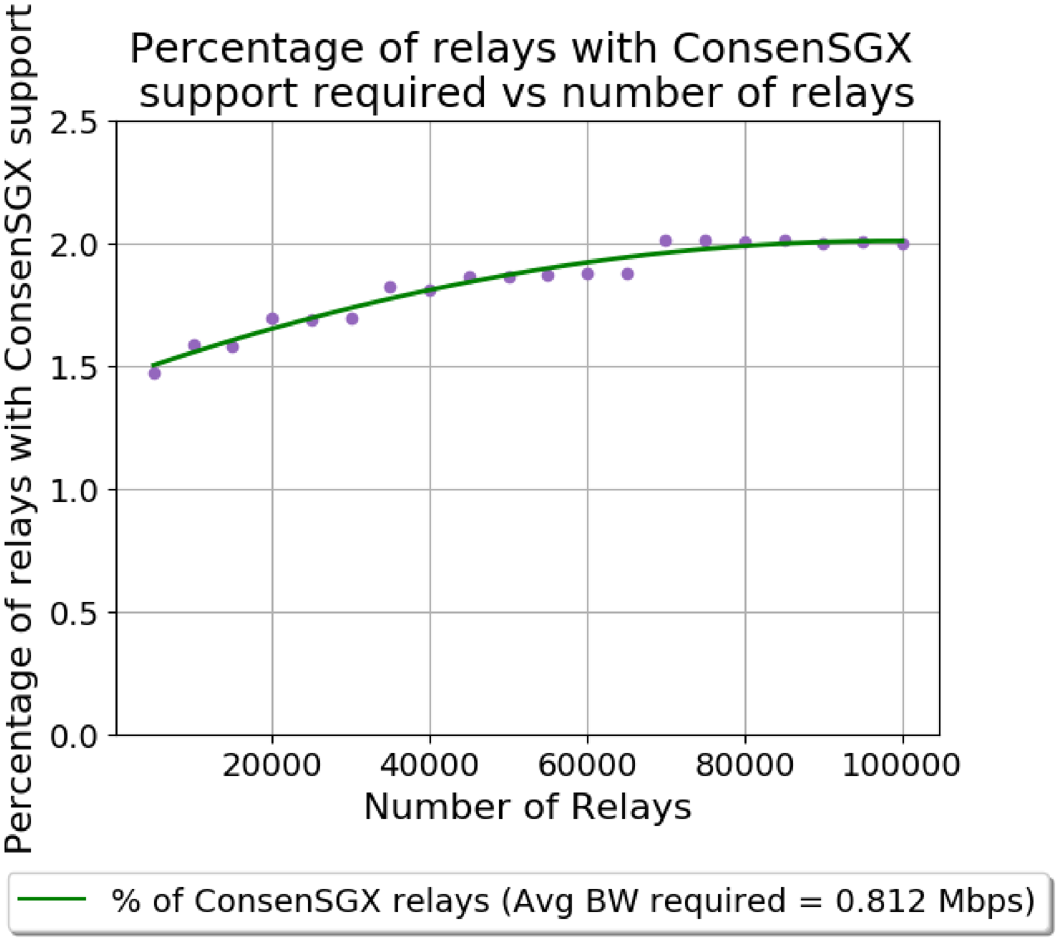
Evaluation


End-to-End time for a query with Tor guard nodes of 4 Mbps bandwidth
CPIR, B=50
Tor Microdescriptor Consensus
ITPIR, B=50
Diff of Tor Microdescriptor Consensus
ConsenSGX, B=50
End-to-end time for single query(in ms)
10^3
10^4
10^5
10^6
10^7
20000
40000
60000
80000
100000
Number of Relays

Evaluation

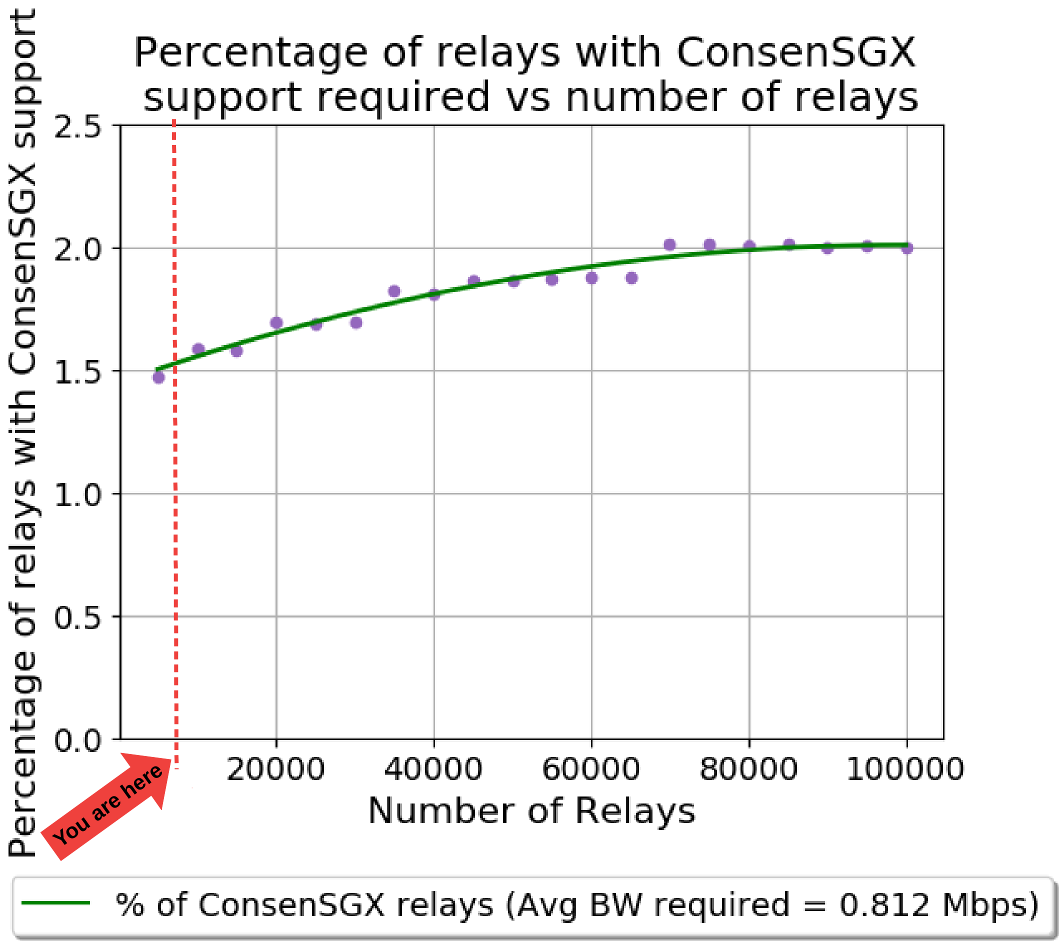

End-to-End time for a query with Tor guard nodes of 4 Mbps bandwidth
CPIR, B=50
Tor Microdescriptor Consensus
ITPIR, B=50
Diff of Tor Microdescriptor Consensus
ConsenSGX, B=50
End-to-end time for single query(in ms)
10^7
10^6
10^5
10^4
10^3
20000
40000
60000
80000
100000
Number of Relays
You are here

Evaluation



Percentage of relays with ConsenSGX support required vs number of relays
Percentage of relays with ConsenSGX support
2.5
2.0
1.5
1.0
0.5
0.0
20000
40000
60000
80000
100000
Number of Relays
% of ConsenSGX relays (Avg BW required = 0.812 Mbps)

Evaluation



Percentage of relays with ConsenSGX support required vs number of relays
Percentage of relays with ConsenSGX support
2.5
2.0
1.5
1.0
0.5
0.0
20000
40000
60000
80000
100000
Number of Relays
You are here
% of ConsenSGX relays (Avg BW required = 0.812 Mbps)

Takeaways

- Distributing the global view of anonymous communications networks is not scalable as the network grows
- Our proposal ConsenSGX defends clients against epistemic attacks while not enforcing clients to maintain a global view of the network
- Evaluations of ConsenSGX show that it is practically deployable and scales well as the Tor network continues to grow.

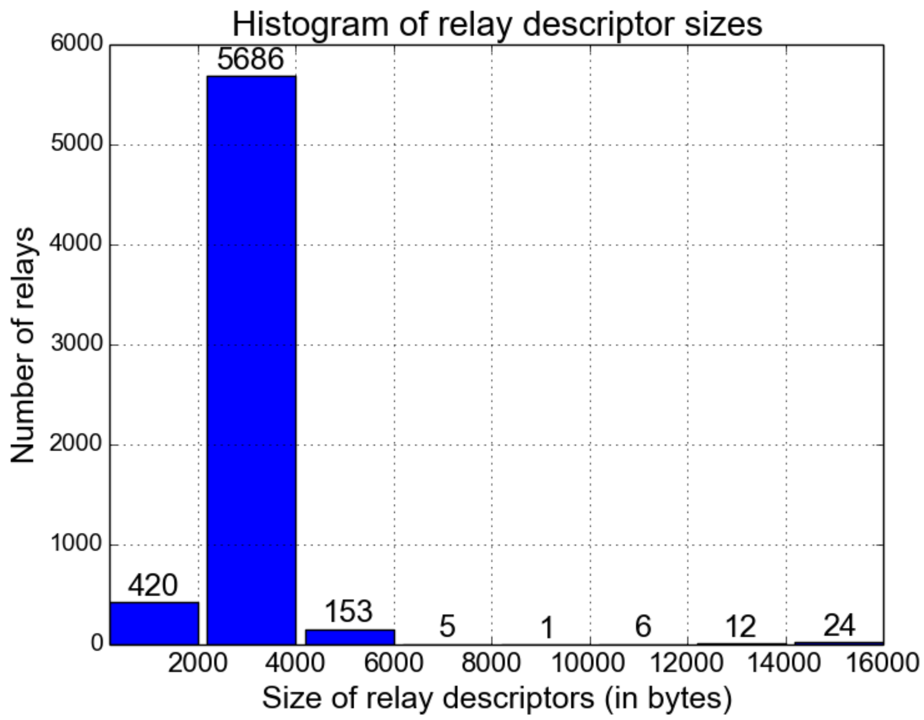
- The paper and source code is available at:

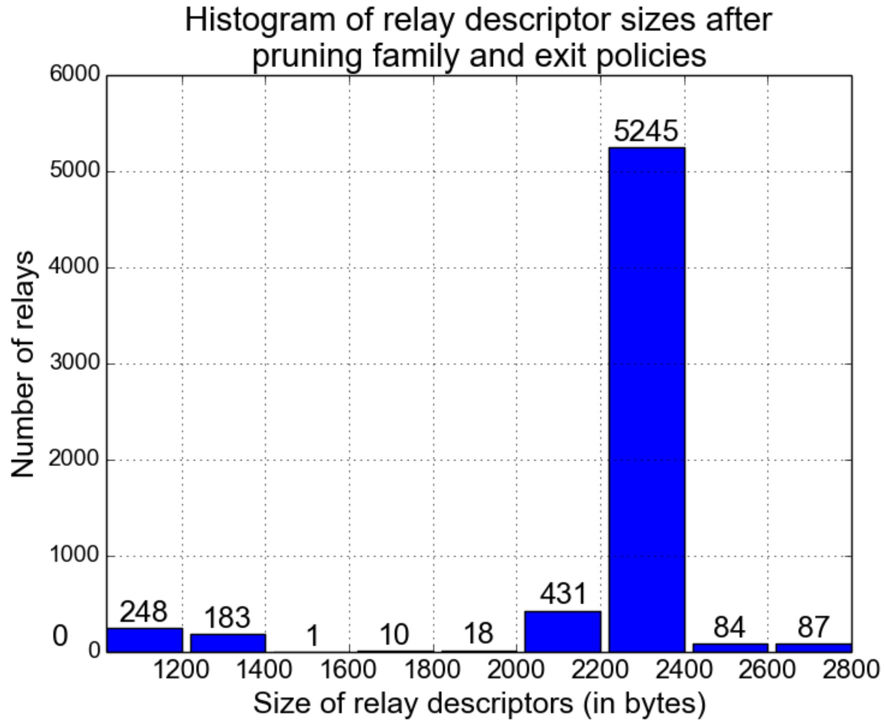
  https://crysp.uwaterloo.ca/software/consensgx/

Approximating Relay Distribution

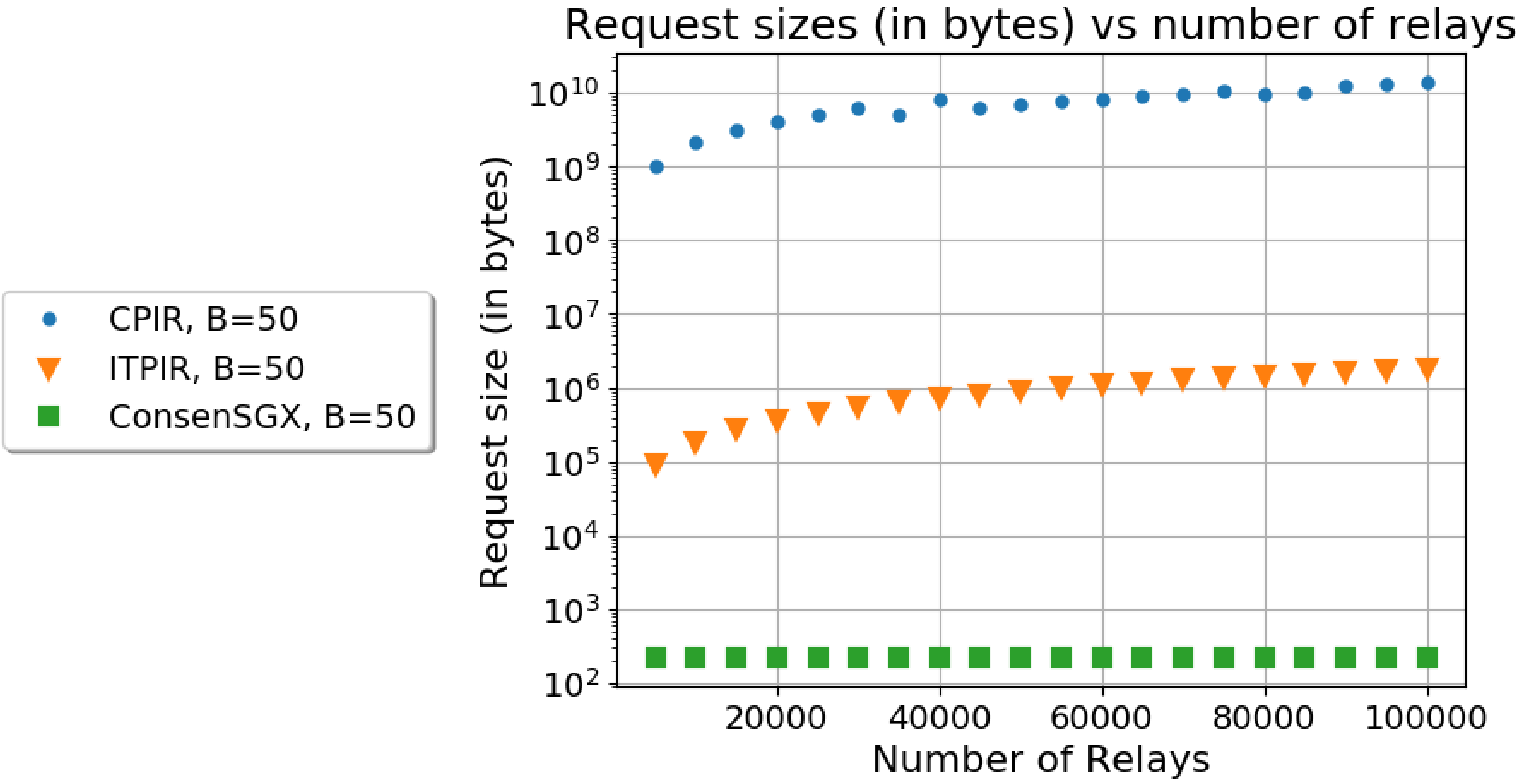
- Imperfect approximations do not harm the security of Tor circuit construction at all
- This mechanism only enables better load distribution of clients across relays
- The bandwidth measurements are themselves noisy
- To deal with potential changes in this distribution, the Directory Authorities can select a few common distributions (exponential, Pareto, etc.)

Relay Descriptor Size Distribution

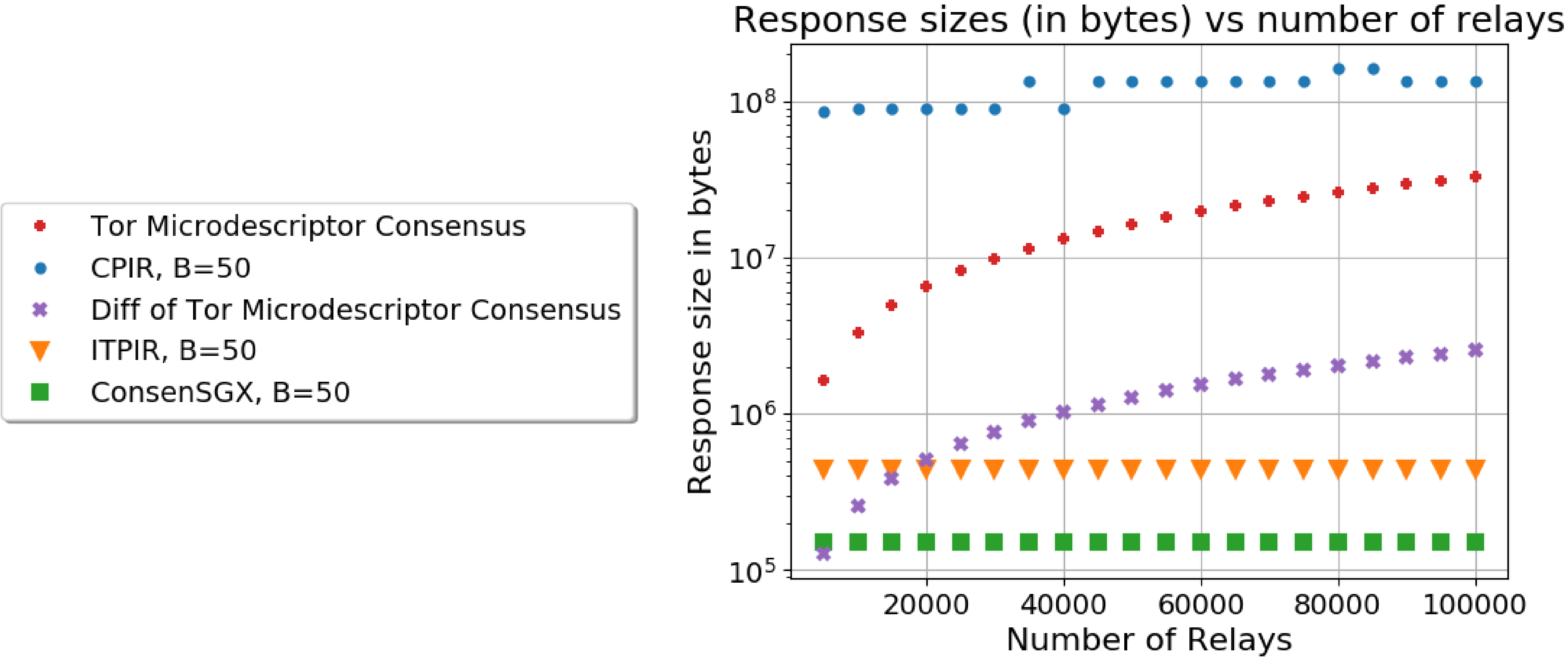




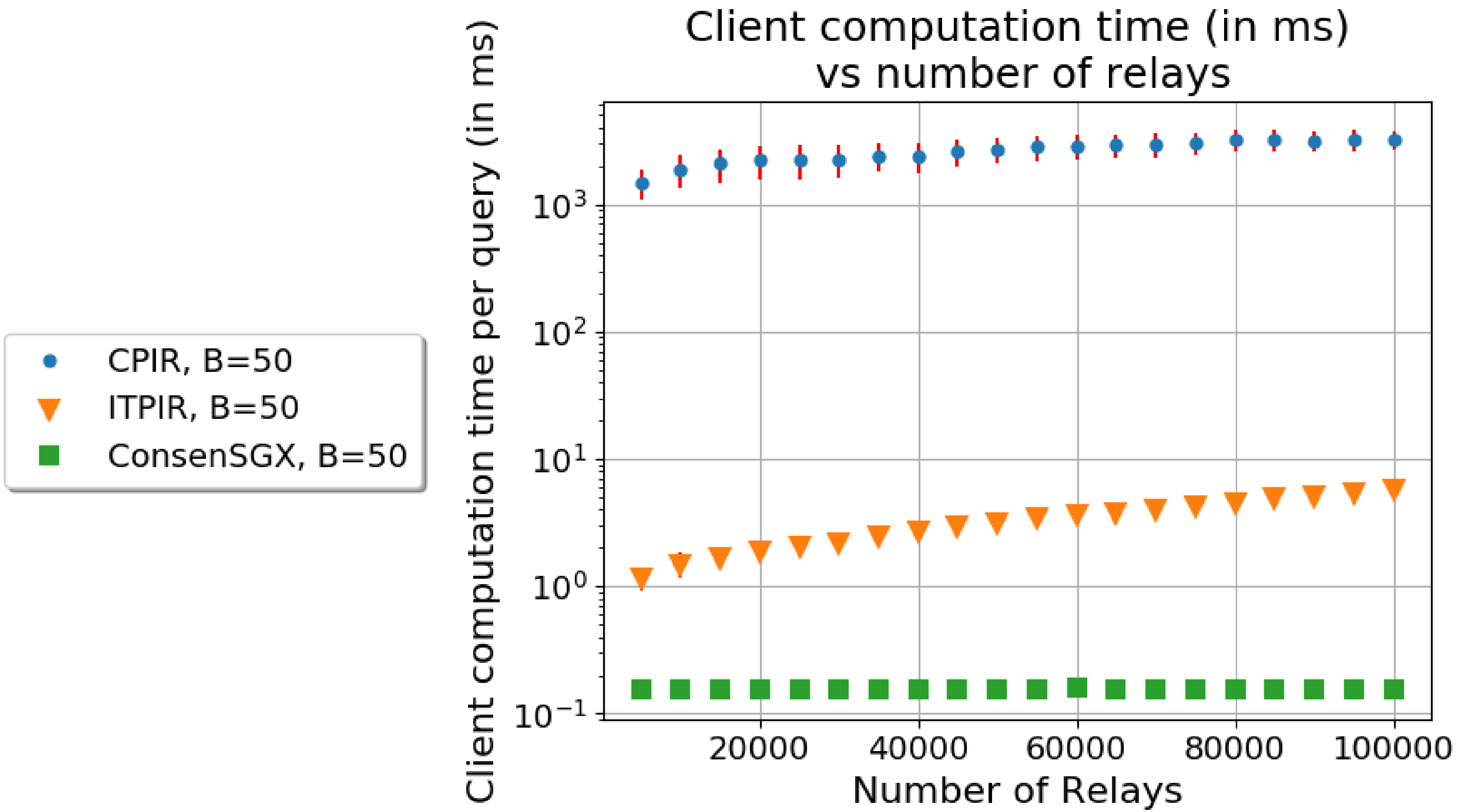
Request Sizes



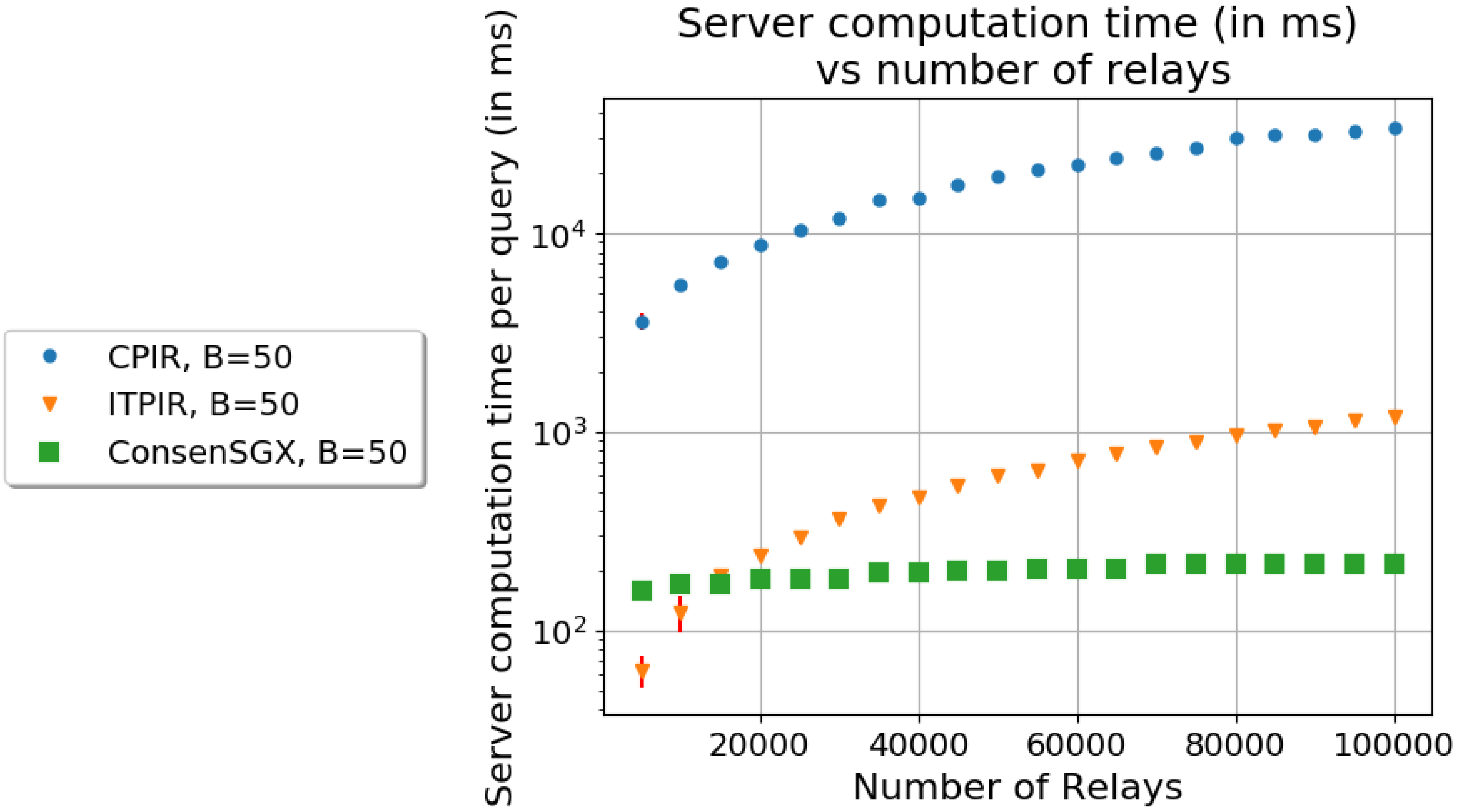
Response Sizes



Client computation overheads



Server computation overheads



Bootstrapping

- Clients need to know the set of ConsenSGX directory caches.
- Like the guard relay selection problem this is a rare operation.
- Once a client has a set of directory caches locally available from a one-time download of the network consenus, they can update them through another full consensus download or ConsenSGX queries itself they locally hold a predefined threshold of unreachable ConsenSGX directory caches.
- Alternatively a set of ConsenSGX serving directory caches can be distributed in the params_consensgx document.

Security Trade-off

- Introduces the security asumption of trusting the underlying TEE used for deploying ConsenSGX in exchange for efficiency.

- Our choice of seperating the long-term signature verification key and the ephemeral assymetric encryption key pair, provides forward secrecy that prevents a malicious processor vendor from inserting a retroactive backdoor.

Compromised TEE

- If the TEE is found to be compromised at a later point of time, it does not trivially deanonymize the client, since traffic over any of the B relays could belong to the client

- Morevoer the Directory Cache only learns that the guard node behind which the client sits queried for those B relay descriptors, but not which client itself did.

Handling Exit Policies

- The existence of very specific exit policies can itself become a deanonymizing attribute
- It would be ideal to limit exit policies to selected exit policy sets
- The directory authorities could also generate an approximation line of relay indices to exit policies

Directory Cache Enrollment

- Provisioning Certification Enclave (PCE) generates a key pair from the fused HW secret.
- It attests other user enclaves on the machine by signing the measurement of that enclave (or output along with measurement)
- The corresponding public keys for PCE verification is available as an X.509 certificate from Intel
- This allows the ORAM controller to generate ephemeral asymmetric keys periodically to enable forward secrecy