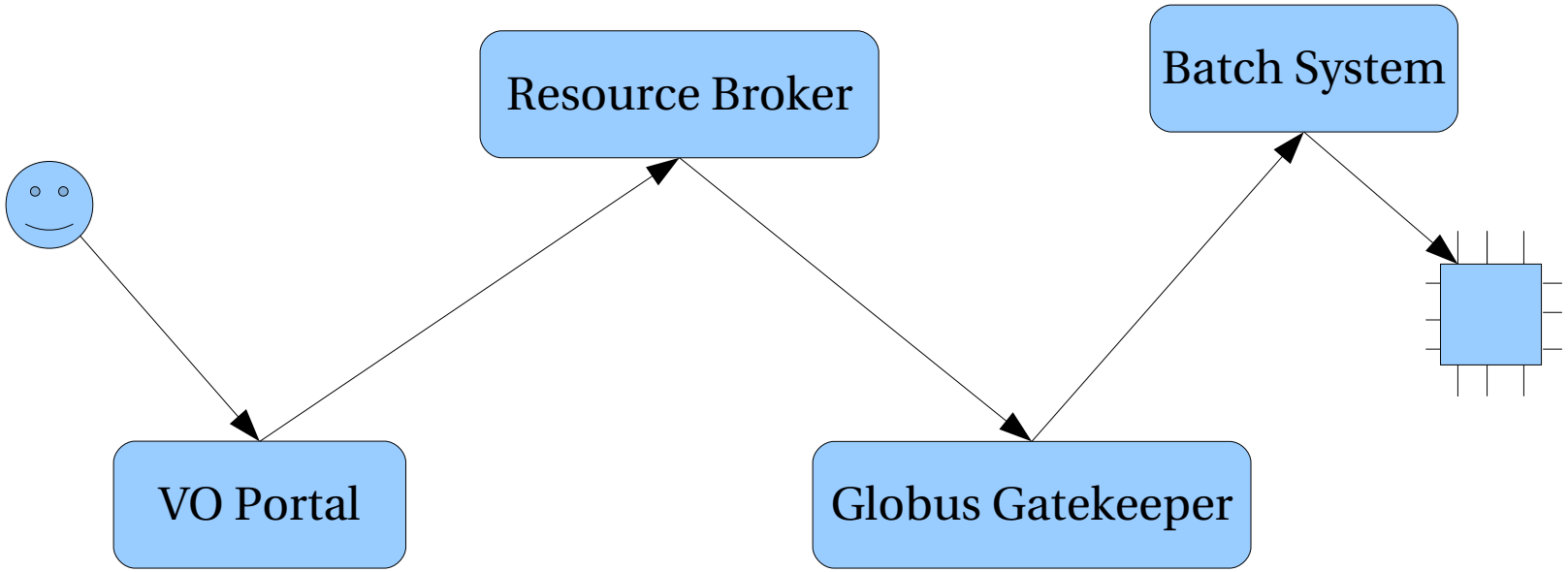
End-to-end security

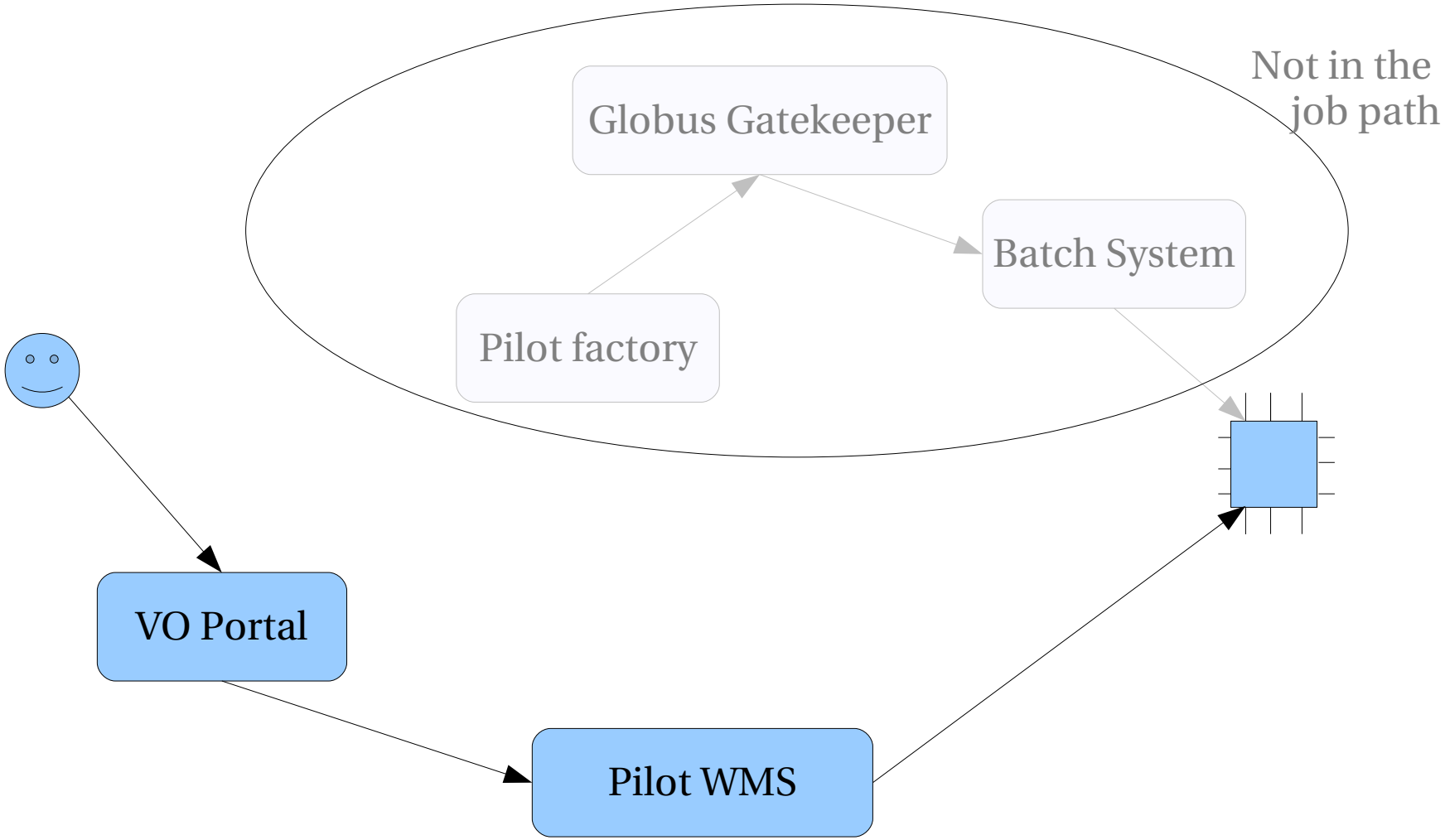
What it is, why we need it, and how it's being implemented

Ian Alderman and Igor Sfiligoi

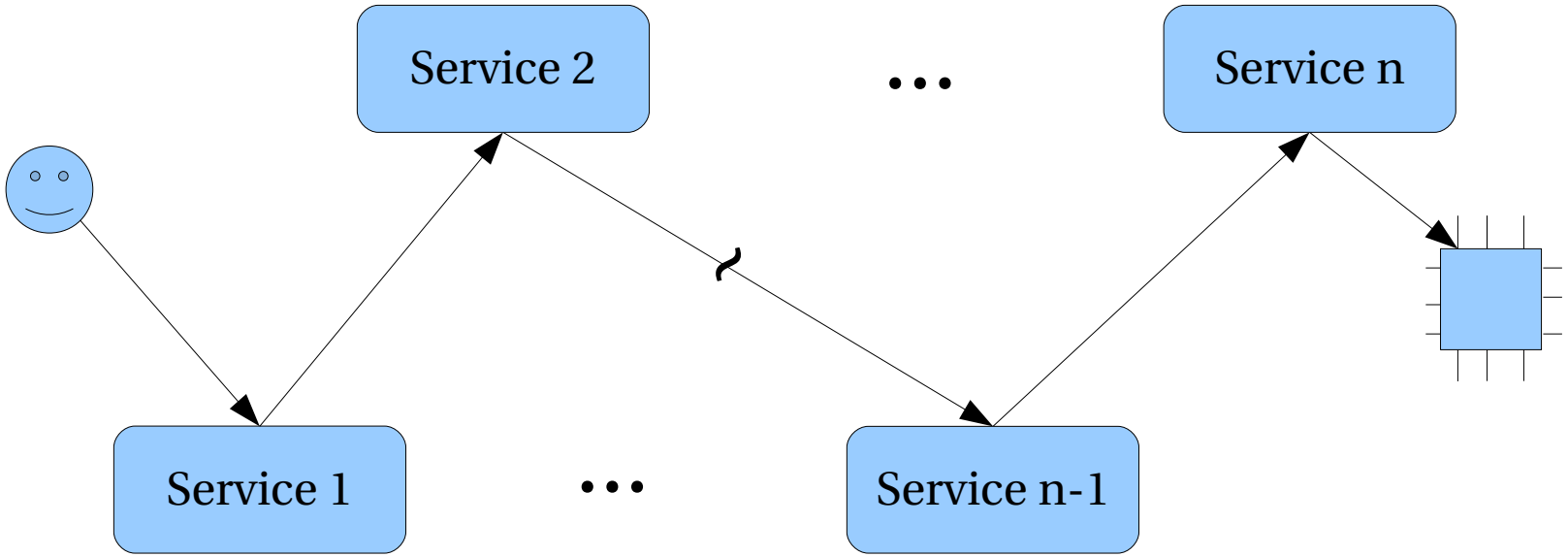
Typical PUSH model



Typical PULL MODEL

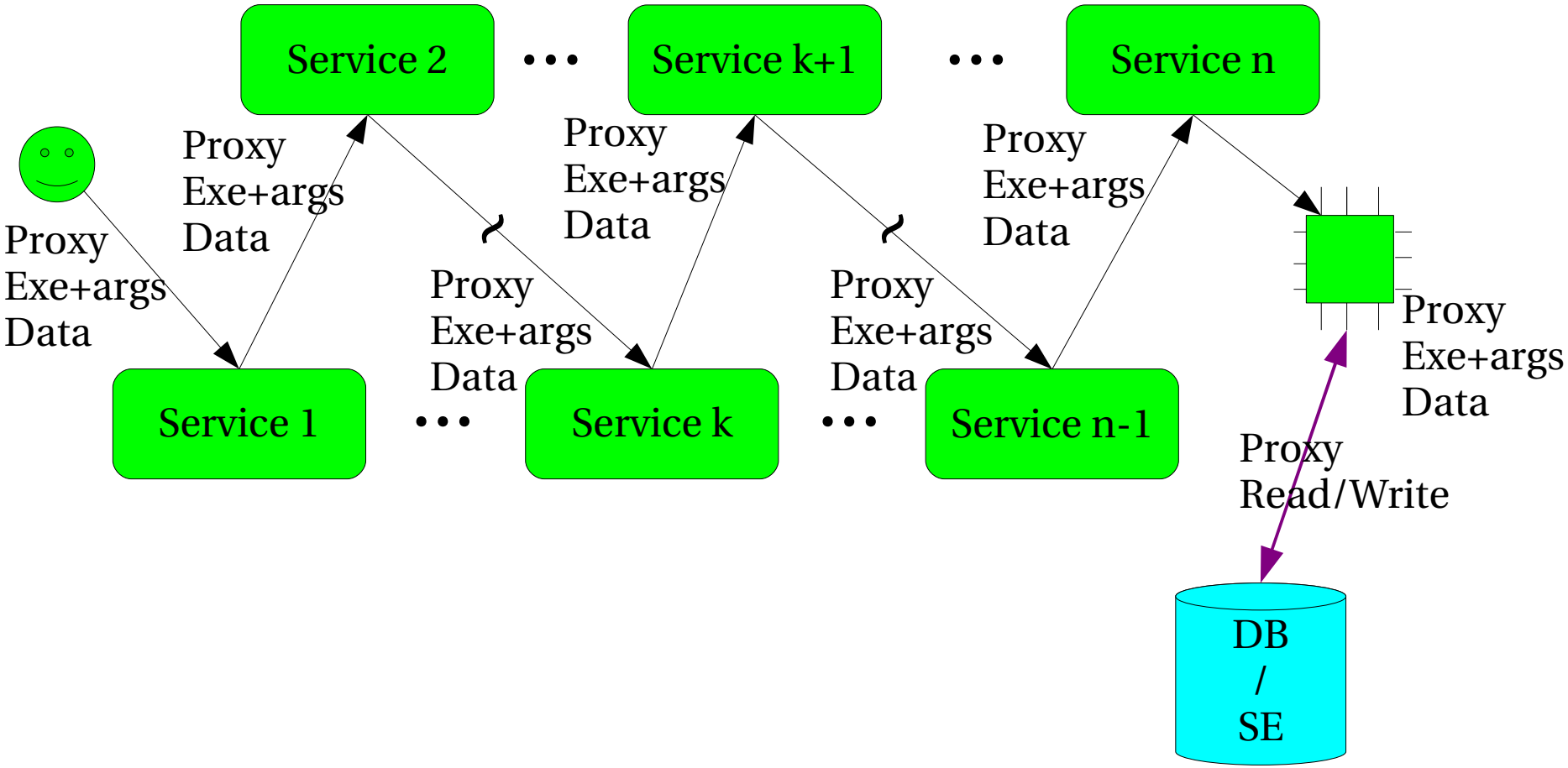


In general

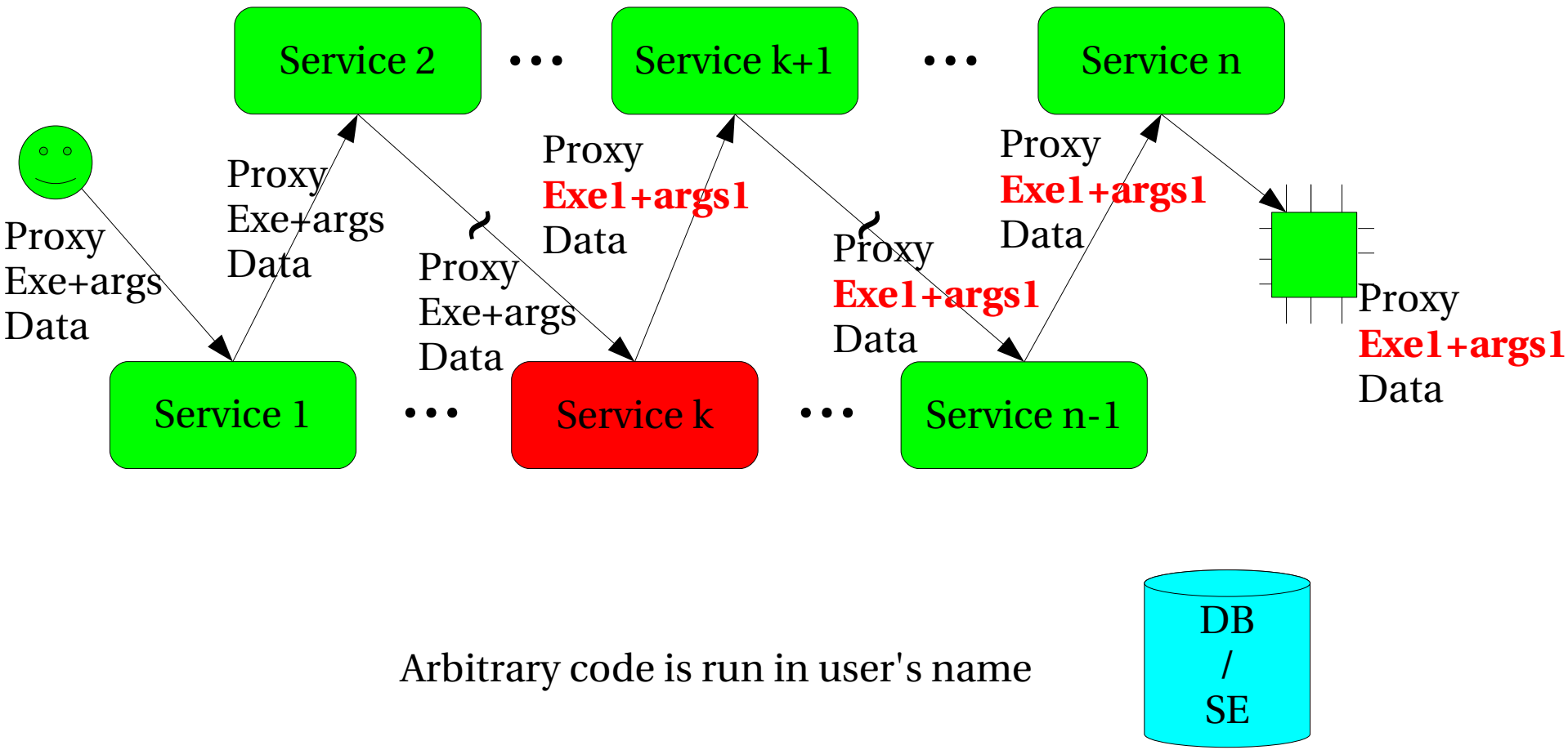


Please notice that the final batch system should be considered
“just another service”

Today, we need to trust all the services

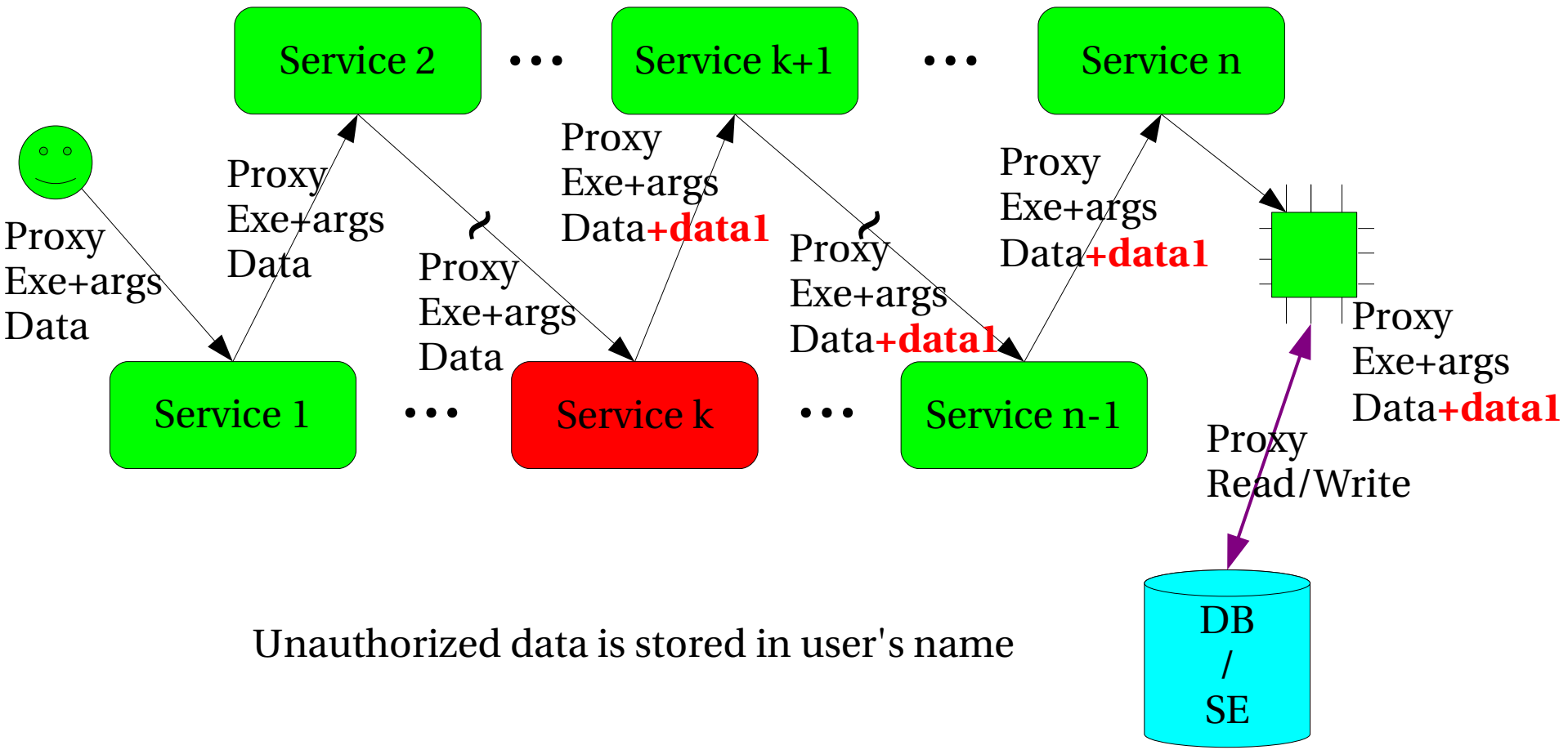


Three results of compromise - 1



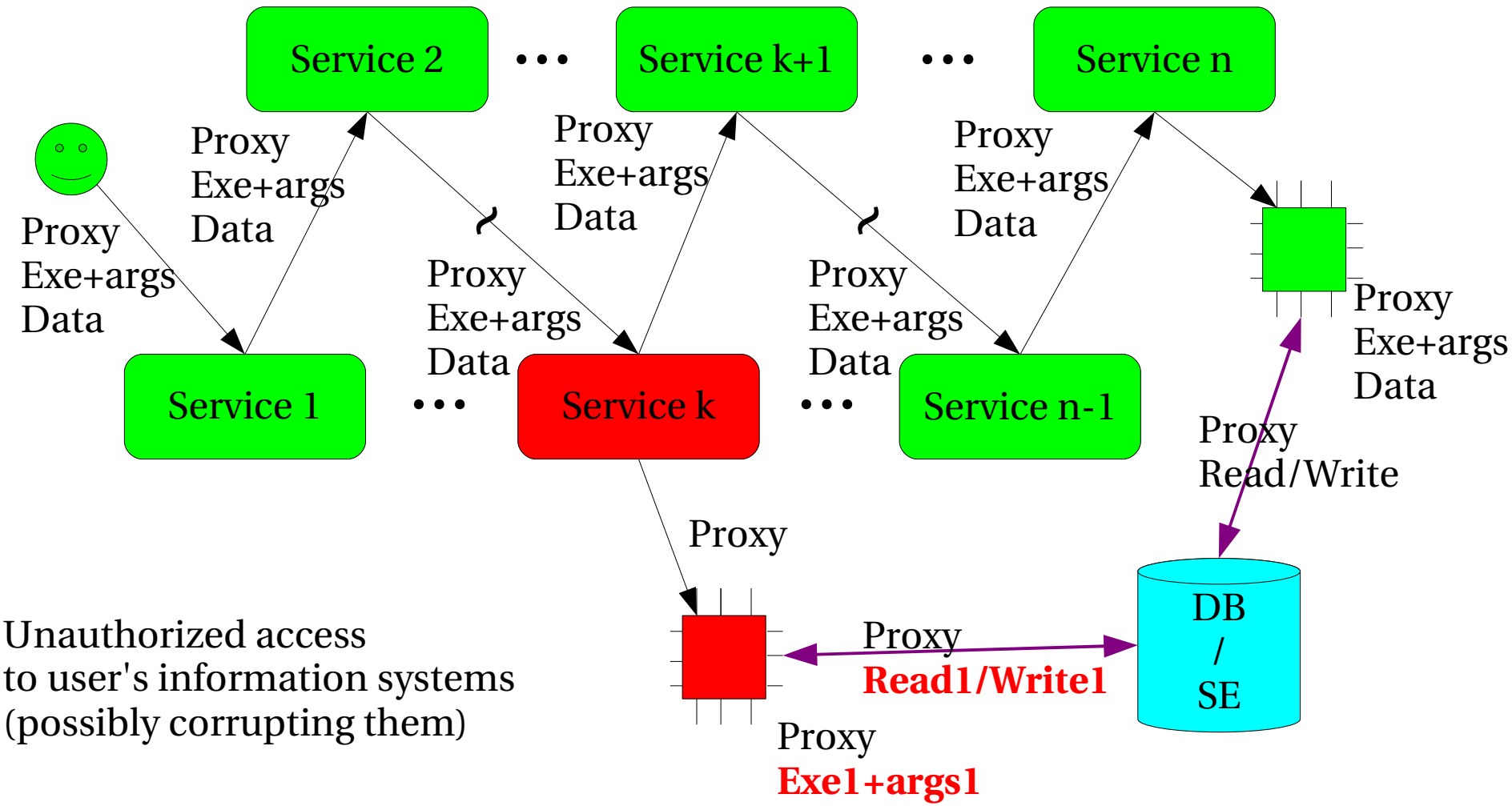
Arbitrary code is run in user's name

Three results of compromise - 2

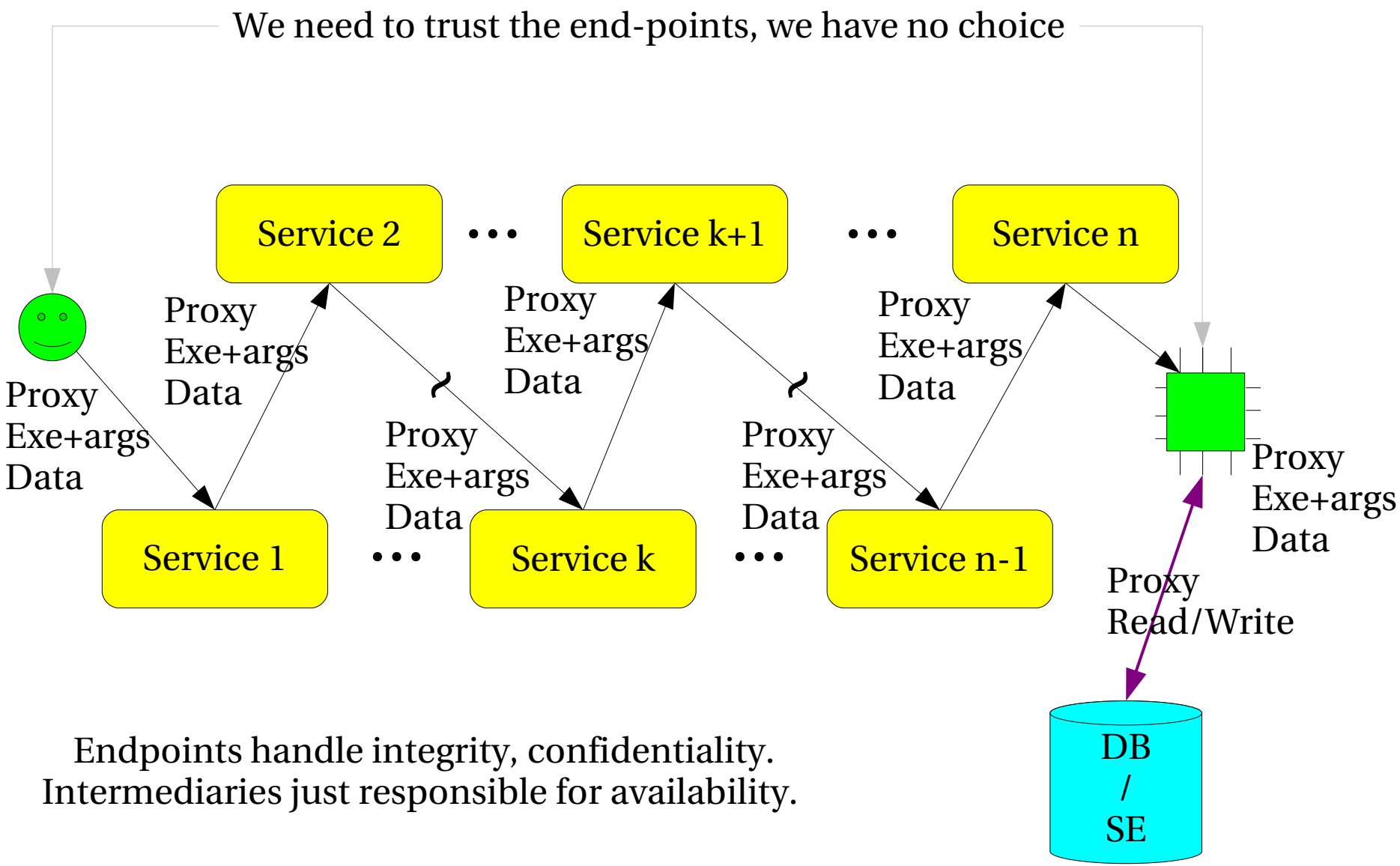


Unauthorized data is stored in user's name

Three results of compromise - 3



Can we live with untrusted services?

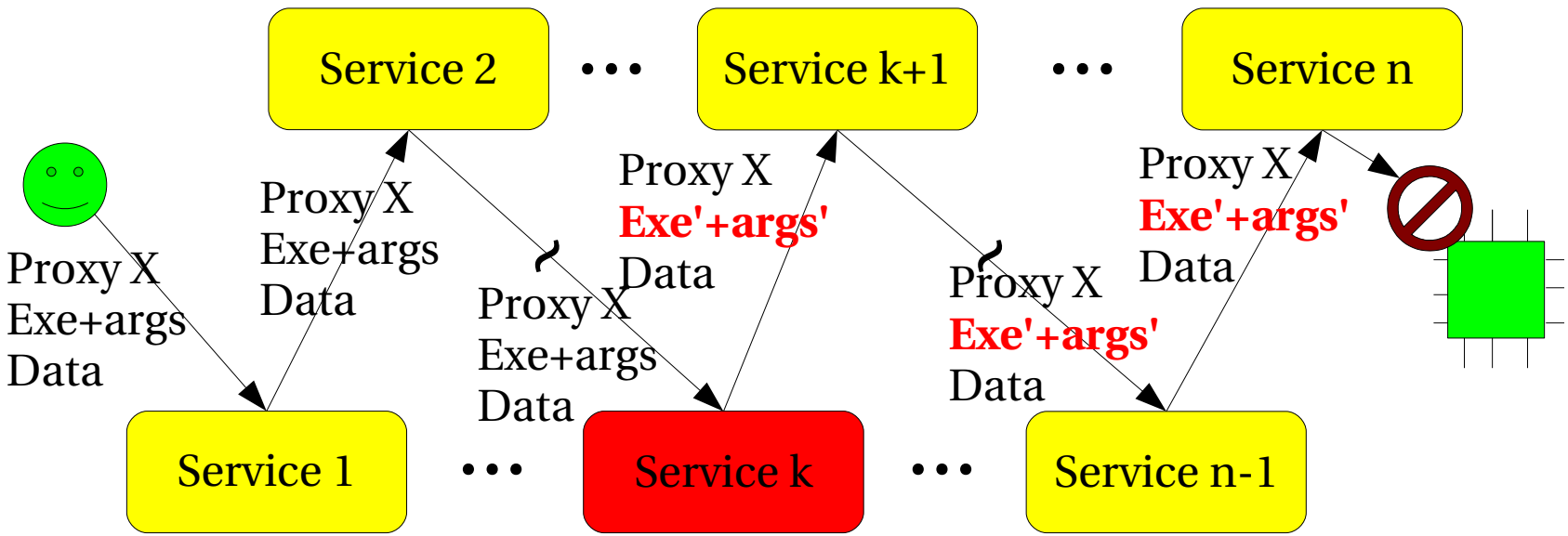


Endpoints handle integrity, confidentiality.
Intermediaries just responsible for availability.

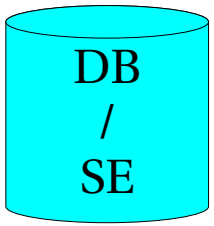
Sign executable+args

Proxy X = Proxy with rule "Execute **`iff hash(Exe+args)==myhash`**"

hash() can be md5(), sha1() or similar



End point checks signature before execution

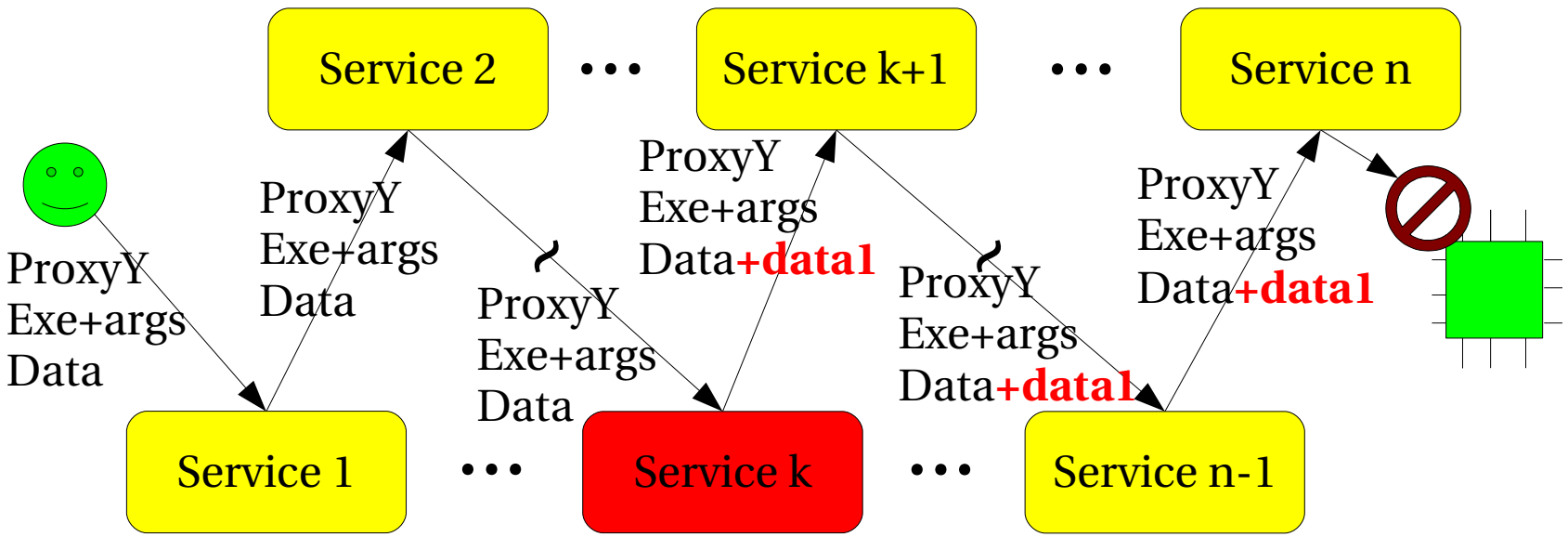


Requirements:
WN must be able to determine integrity of executables.
User specifies policy; Worker node interprets it.
WN must be able to associate task with proxy.

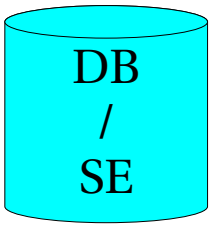
Sign data

ProxyY = Proxy with rule “Stage data iff hash(data)==myhash”

hash() can be md5(), sha1() or similar



End point checks signature before staging data
Since without data the job could fail, it should not run the job either



Requirements:
WN must be able to determine integrity of data.

Integrity

- WN must be able to verify the integrity of executables, arguments, and input data. User must be able to verify the integrity of results.
- WN's check signatures on executables, arguments and input, and sign output data (results). User can verify the signature on the results, and verify whether the WN was consistent with policy.
- The task's signed ClassAd specifies executables, arguments, and input data; external files are “included” through cryptographic hashes. Completed task ClassAds are signed by WNs before output data is returned to the client; output data file hashes are included.

Requirement
Solution
Implementation

Integrity

- WN must be able to verify the integrity of executables, arguments, and input data. User must be able to verify the integrity of results.
- WN's check signatures on executables, arguments and input, and sign output data (results). User can verify the signature on the results, and verify whether the WN was consistent with policy.
- The task's signed ClassAd specifies executables, arguments, and input data; external files are "included" through cryptographic hashes. Completed task ClassAds are signed by WNs before output data is returned to the client; output data file hashes are included.

```
[
  owner = "ian"
  executable = "a.out"
  input = "file1.txt"
  arguments = "-safe"
  ...
  executable_hash = "de1a..."
  input_hash = "be5ed23a0..."
  ...
  cad_sig = "long opaque string"
]
```

Policy extended proxies

- Unique executable, arguments, input data ⇔ unique proxy. Intermediaries don't need to understand, interpret or enforce policy.
- Each task has a unique proxy certificate issued by the submitting user's proxy. Intermediaries are only assumed to provide availability.
- An additional proxy is generated by the user for each task containing the policies and signature. Policy and signatures are included in proxy certificates as an X509.v3 attribute which can be ignored by intermediaries.

Policies

- Users must be able to specify policies describing appropriate uses of their tasks and credentials. Worker nodes and resources must be able to interpret and enforce policies.
- Users specify policies such as acceptable WN trust roots and signs executables, arguments and input data. Policy enforcement is performed by trusted endpoints and policy aware resources.
- Users specify policy in the ClassAd language with additional primitives. ClassAd policy expressions are evaluated by endpoints and resources in addition to local access control settings. (ACLs + capabilities)

Hash of public key x: $\mathbf{hP_x}$

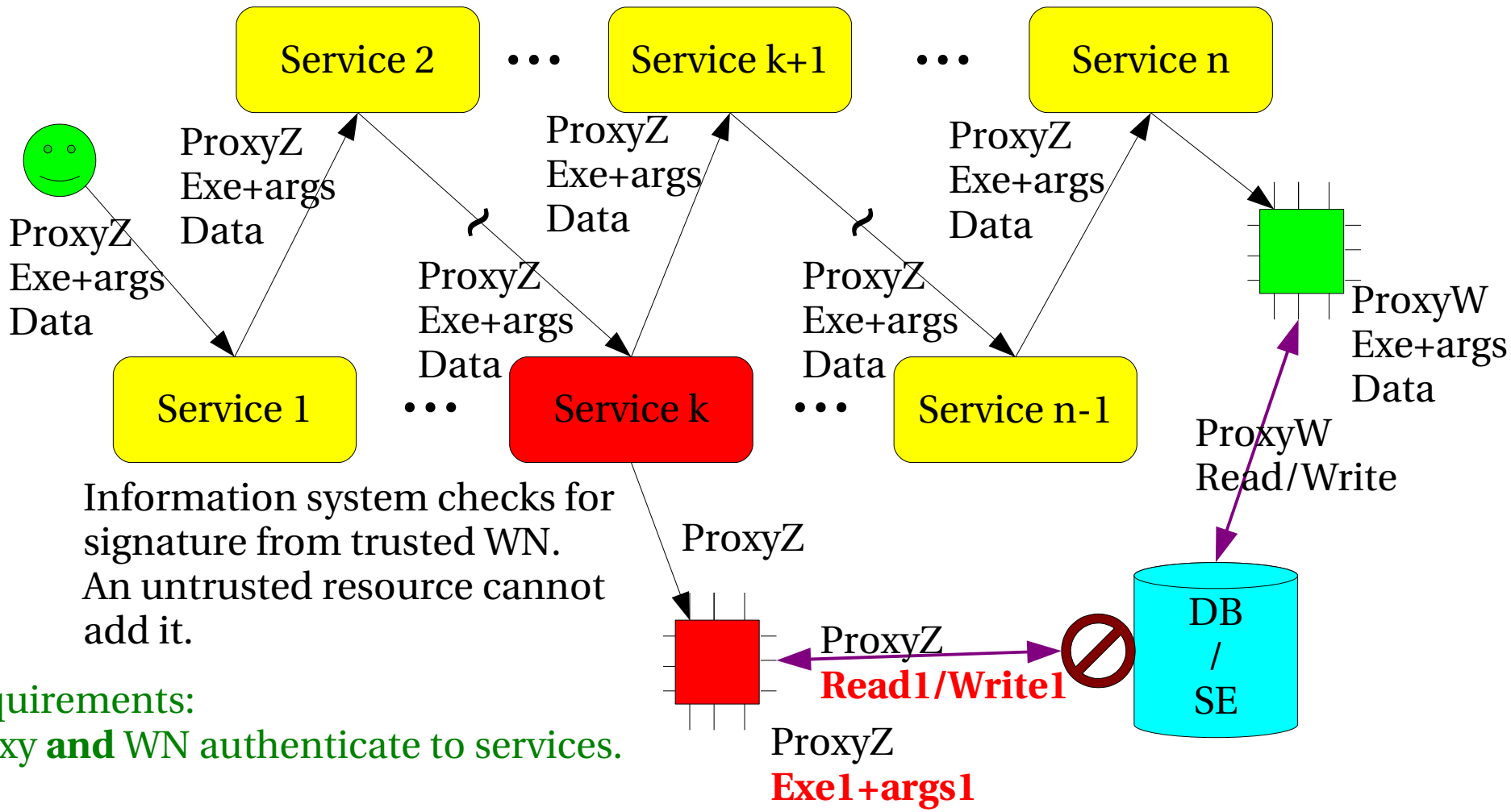
$$\mathbf{execute_aae = exec_ca(hP_e)}$$

$$\mathbf{access_aae = exec_ca(hP_e)\ \&\&\ resource_ca(hP_r)}$$

Authenticating the WN too

ProxyZ = Proxy with rule “Allow access to my info systems only if signed by trusted WN”

ProxyW = ProxyZ signed with the WN host cert (or equivalent)



Requirements:
Proxy **and** WN authenticate to services.

Two key authentication

- When a WN authenticates with a service on behalf of a user, the service must be able to authenticate both the user and the WN to enforce the user's policy.
- Proxy delegation includes the additional step of including a signature performed by the WN on the delegation chain.
- The resource enforces policy by verifying that the signing WN is consistent with user-specified policy.

Protoproposal: Generalizing End-to-end Security to GSI

- Job description language: ClassAds.
  - GSI: Can a job be completely described as a set of attribute-value pairs?
- Policy language: ClassAds expressions.
  - GSI: Most policies are simple enough?
- Proxy certificate attribute type: added v3 atribute.
  - GSI: same?

Summary

Adding the three roles and have trusted worker nodes and information systems use them will remove the requirement of having trustworthy WMS services

Requirements

- WN must be able to verify the integrity of executables, arguments, and input data.
- User must be able to verify the integrity of results.
- User must be able to specify policies describing appropriate uses of their tasks and credentials.
- Worker node must be able to interpret and enforce policies.
- Intermediaries don't need to understand, interpret, or enforce policy.
- Unique executable, arguments, input data <=> unique proxy.
- When a WN authenticates with a service on behalf of a user, the service must be able to authenticate both the user and the WN to enforce the user's policy.