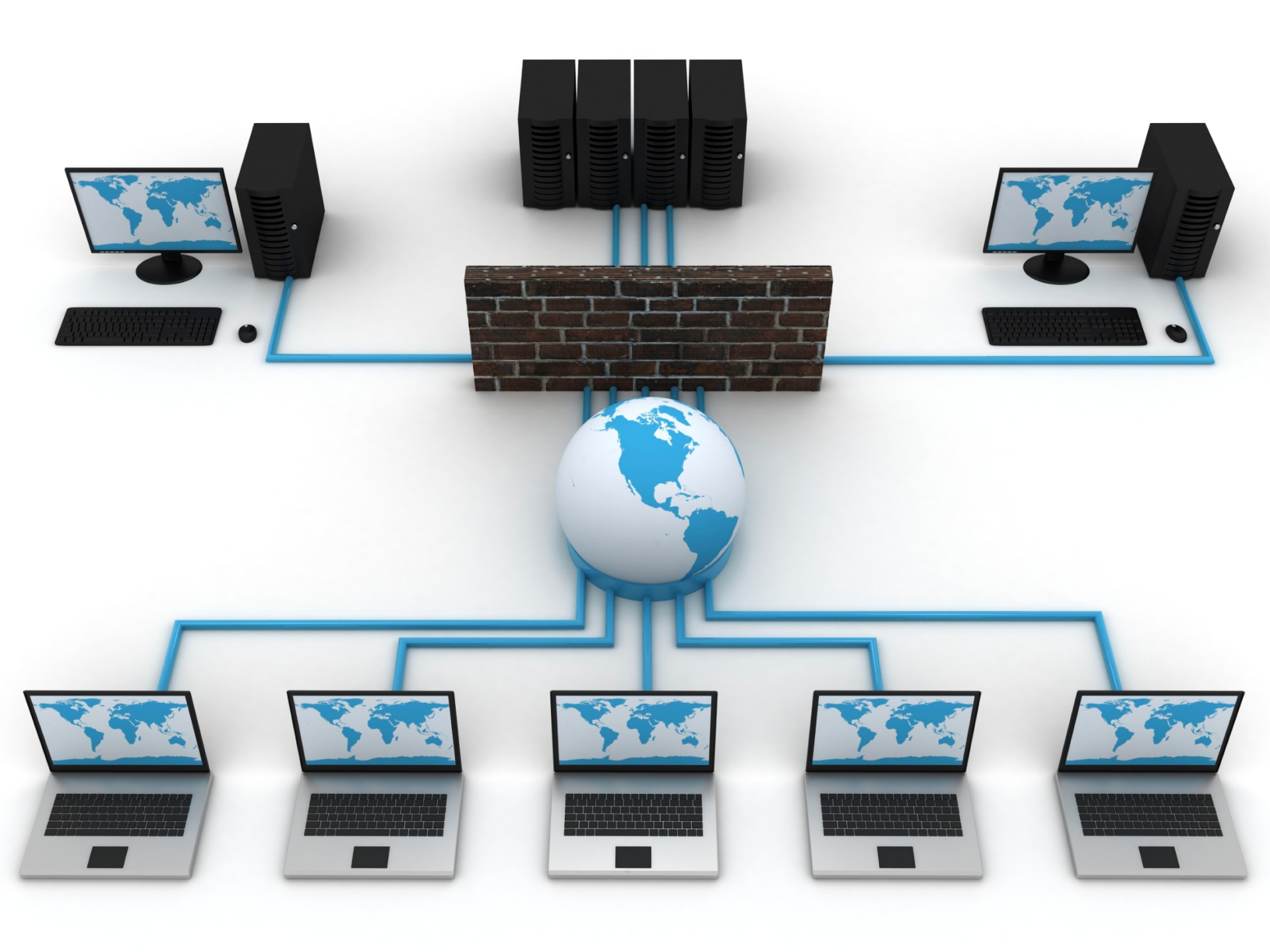
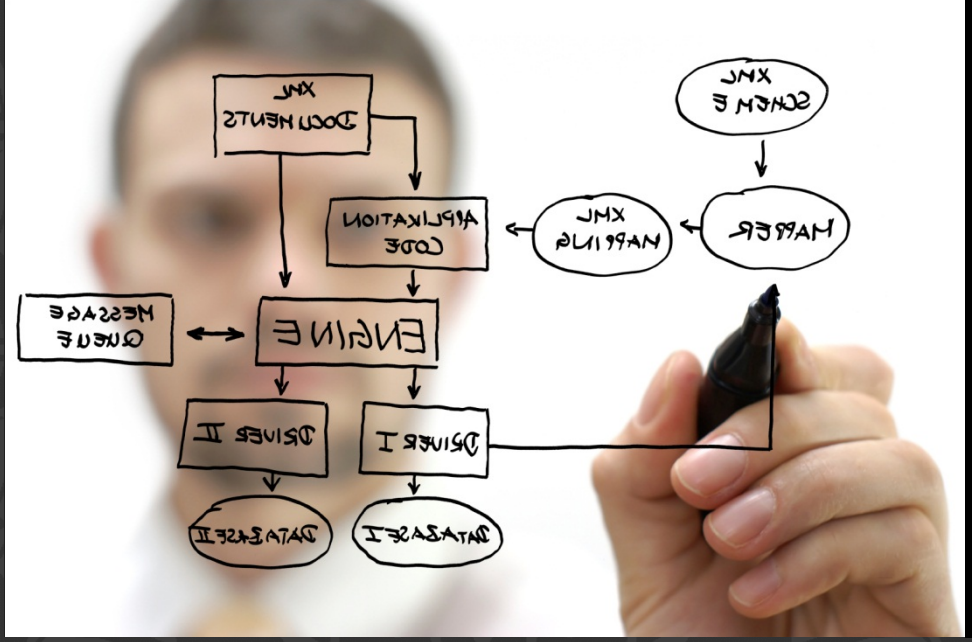
MOMMIE KNOWS BEST

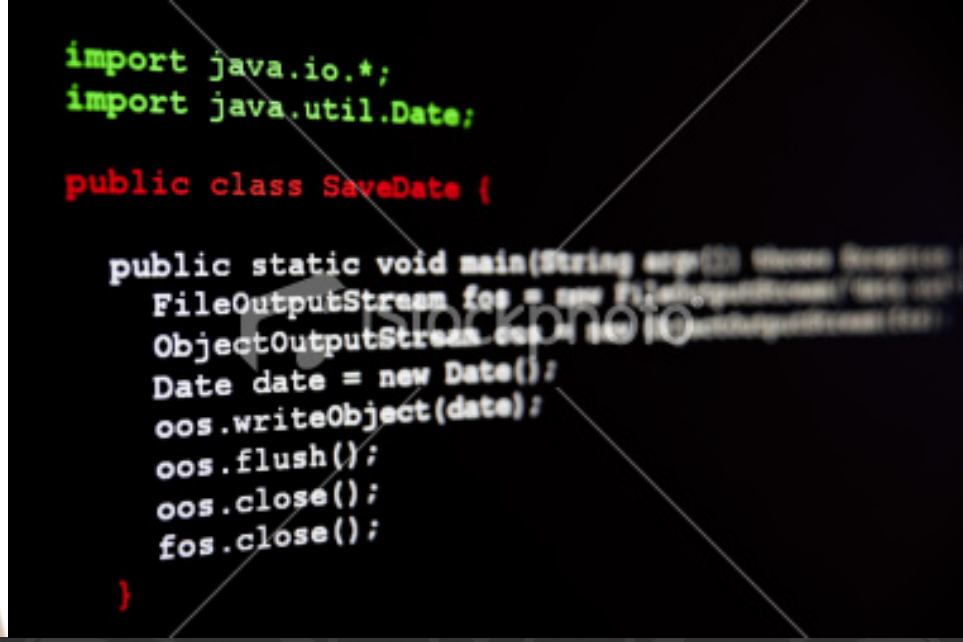
SYSTEMATIC OPTIMIZATIONS FOR VERIFIABLE DISTRIBUTED ALGORITHMS

Petros Maniatis (Intel), Mike Dietz (Rice), Babi Papamanthou (Brown)

Abstraction







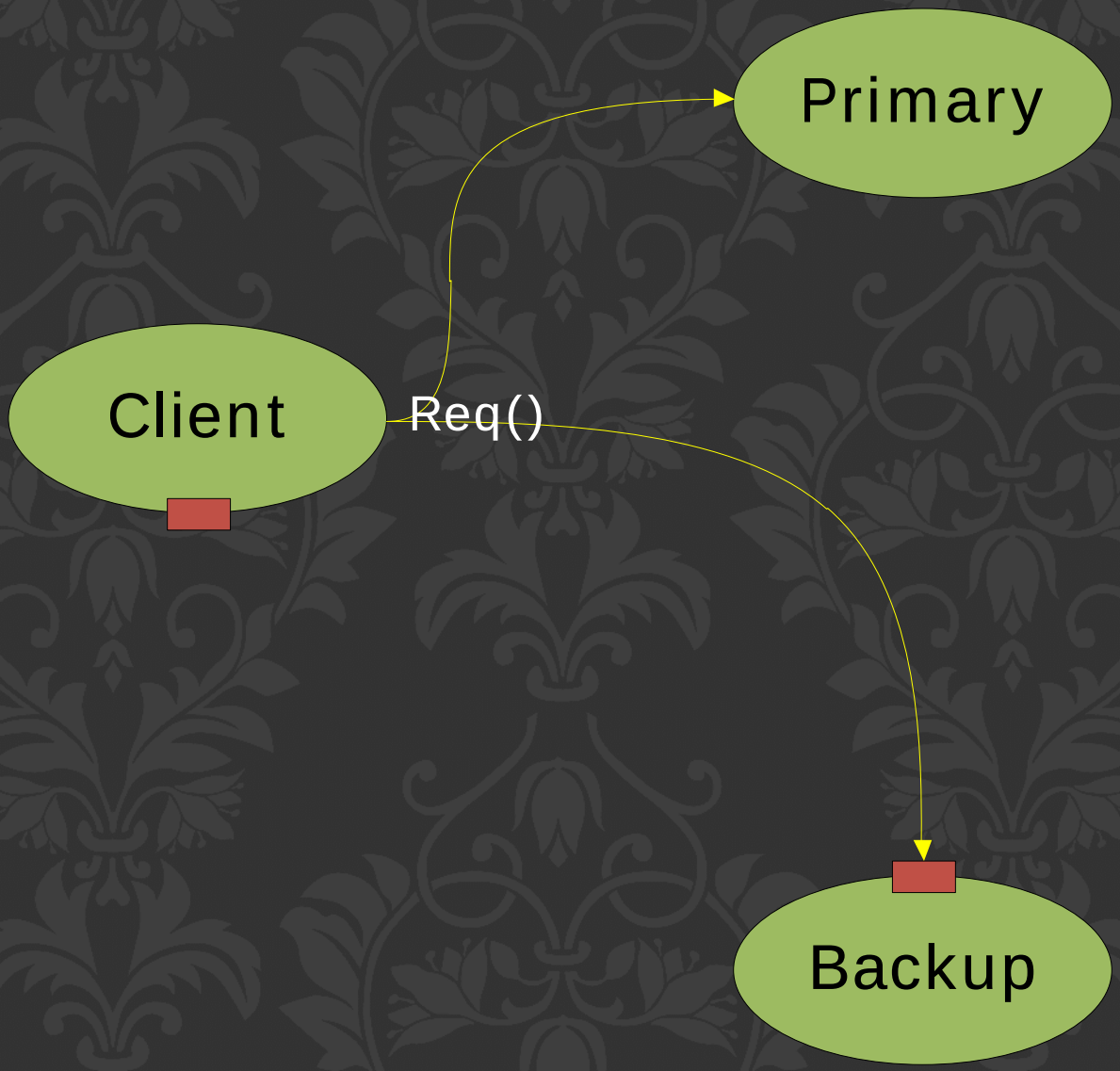
Consistency
Security
Privacy
Fault tolerance
Correctness

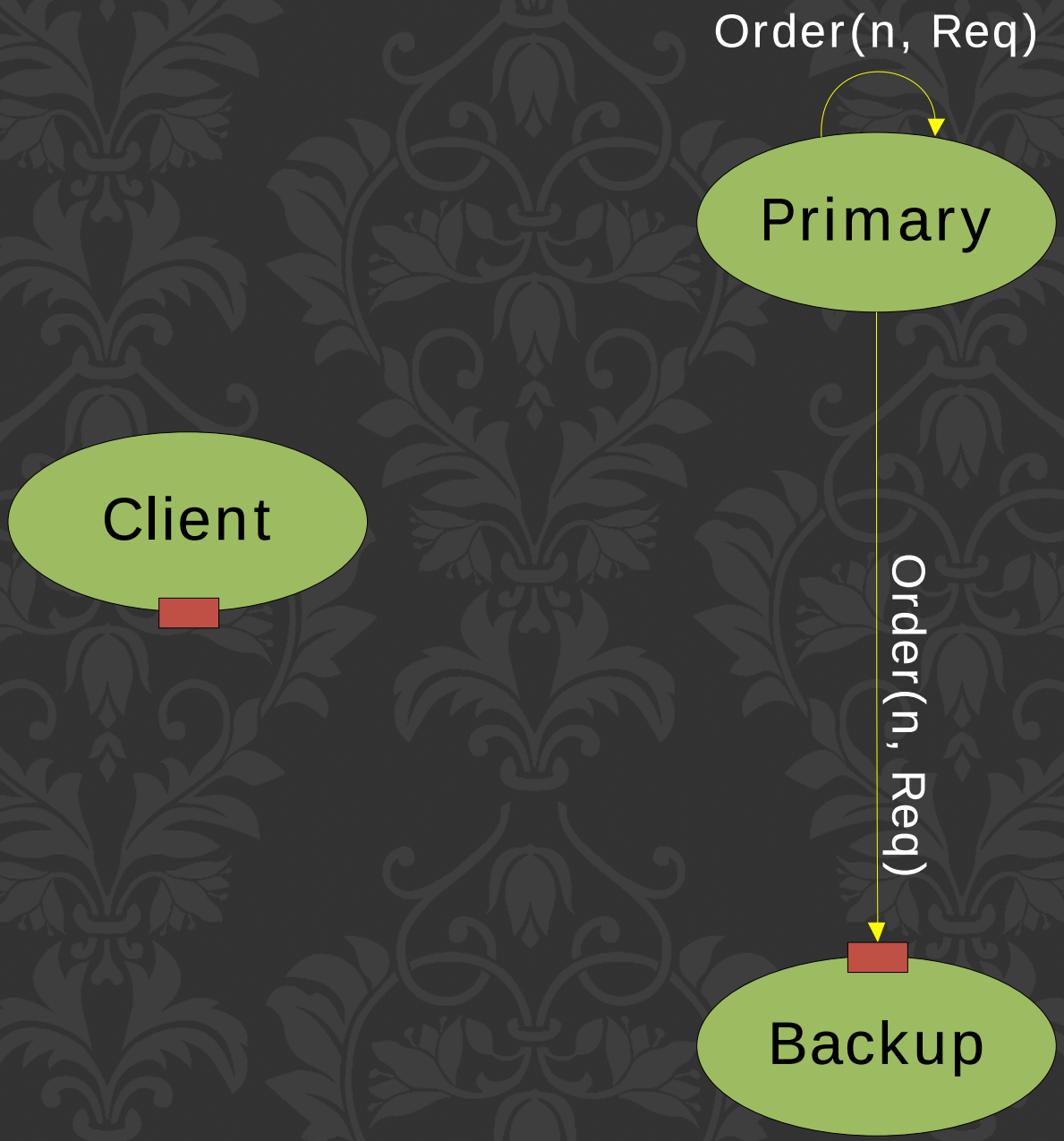
Data structures
Cryptography
Ordering
Representation
Retransmissions

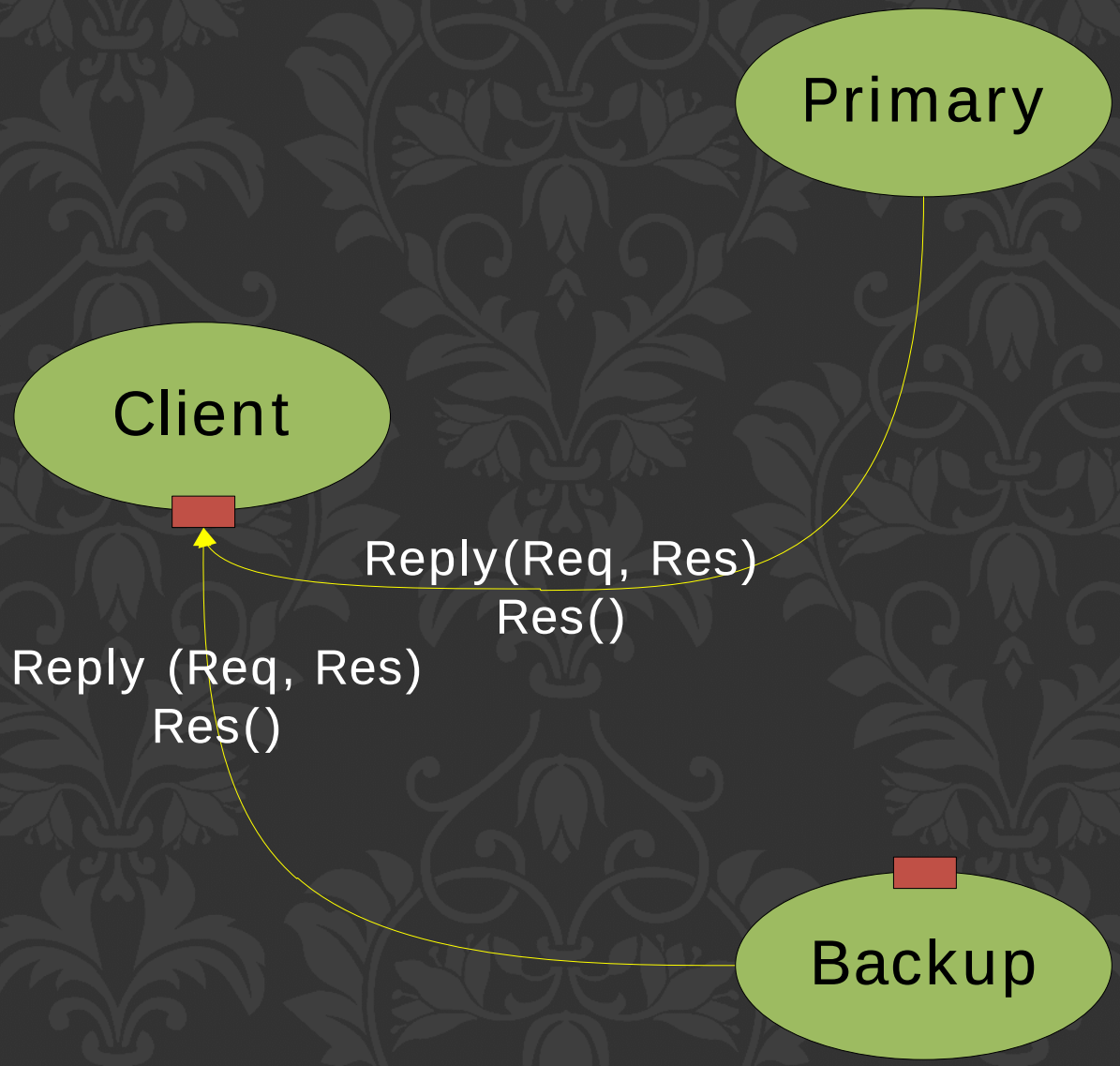

Declarativity

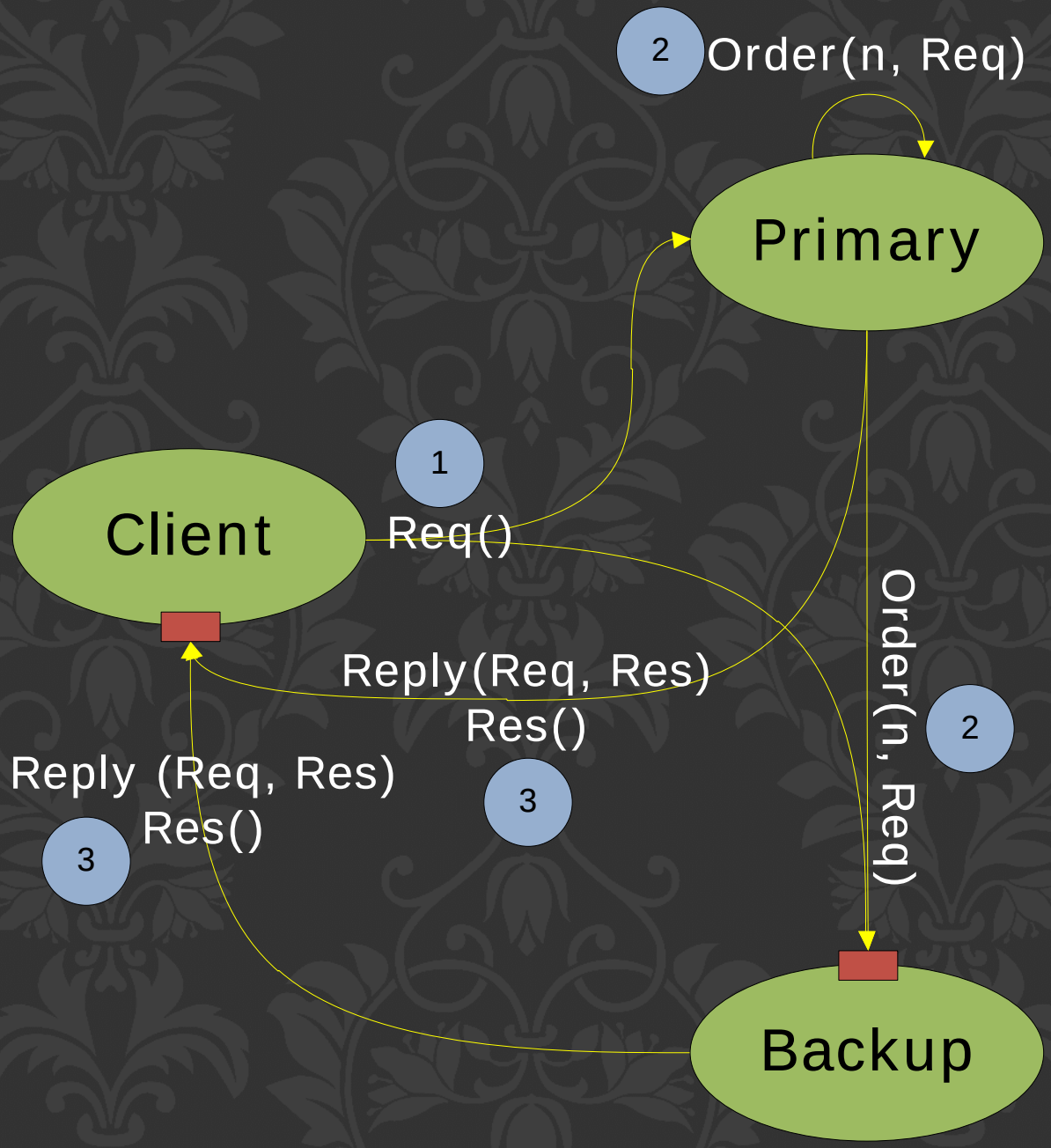

BFT

A BRIEF ILLUSTRATION

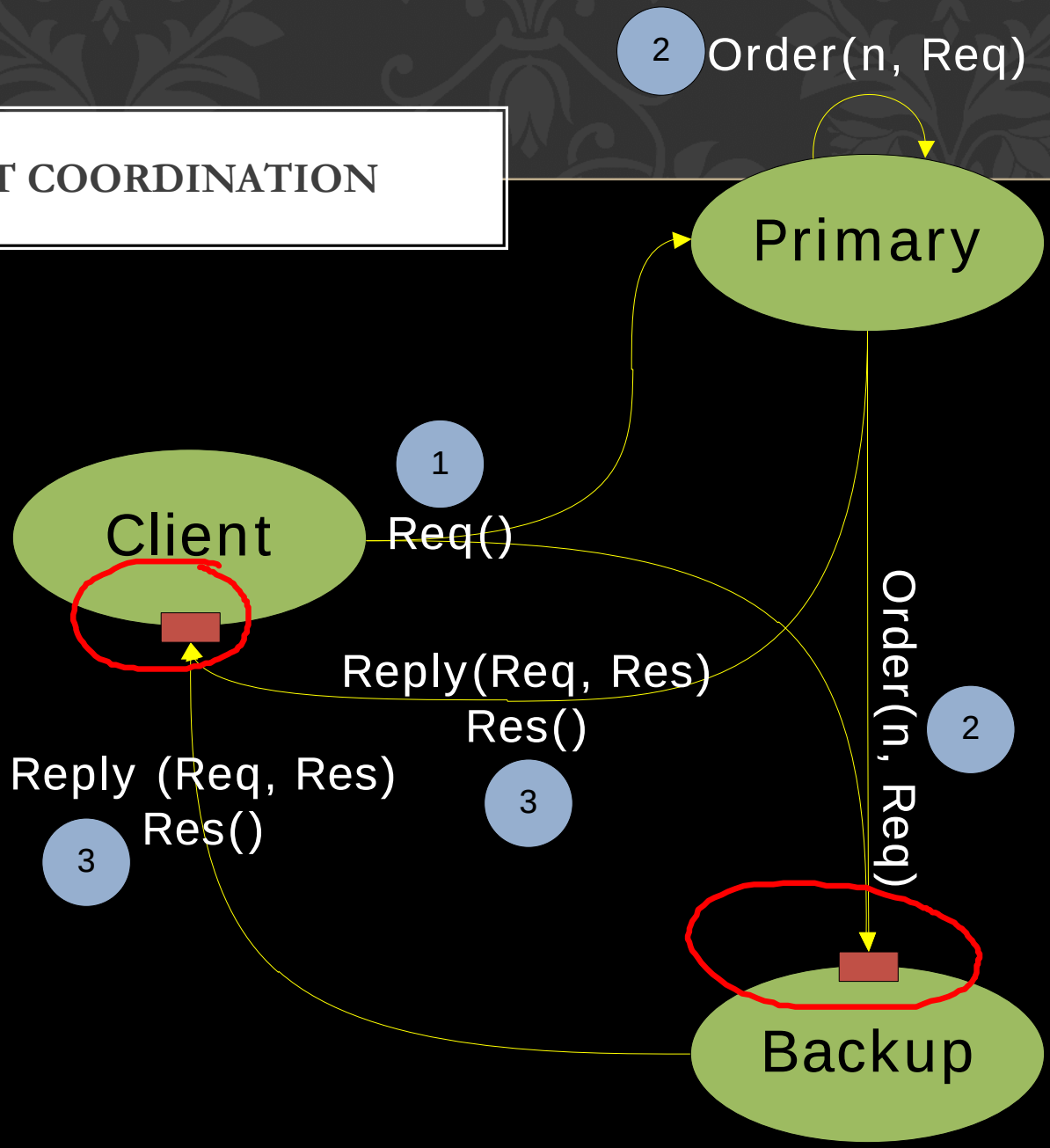

Primary
Client
Req()
Backup


Order(n, Req)
Primary
Client
Order(n, Req)
Backup

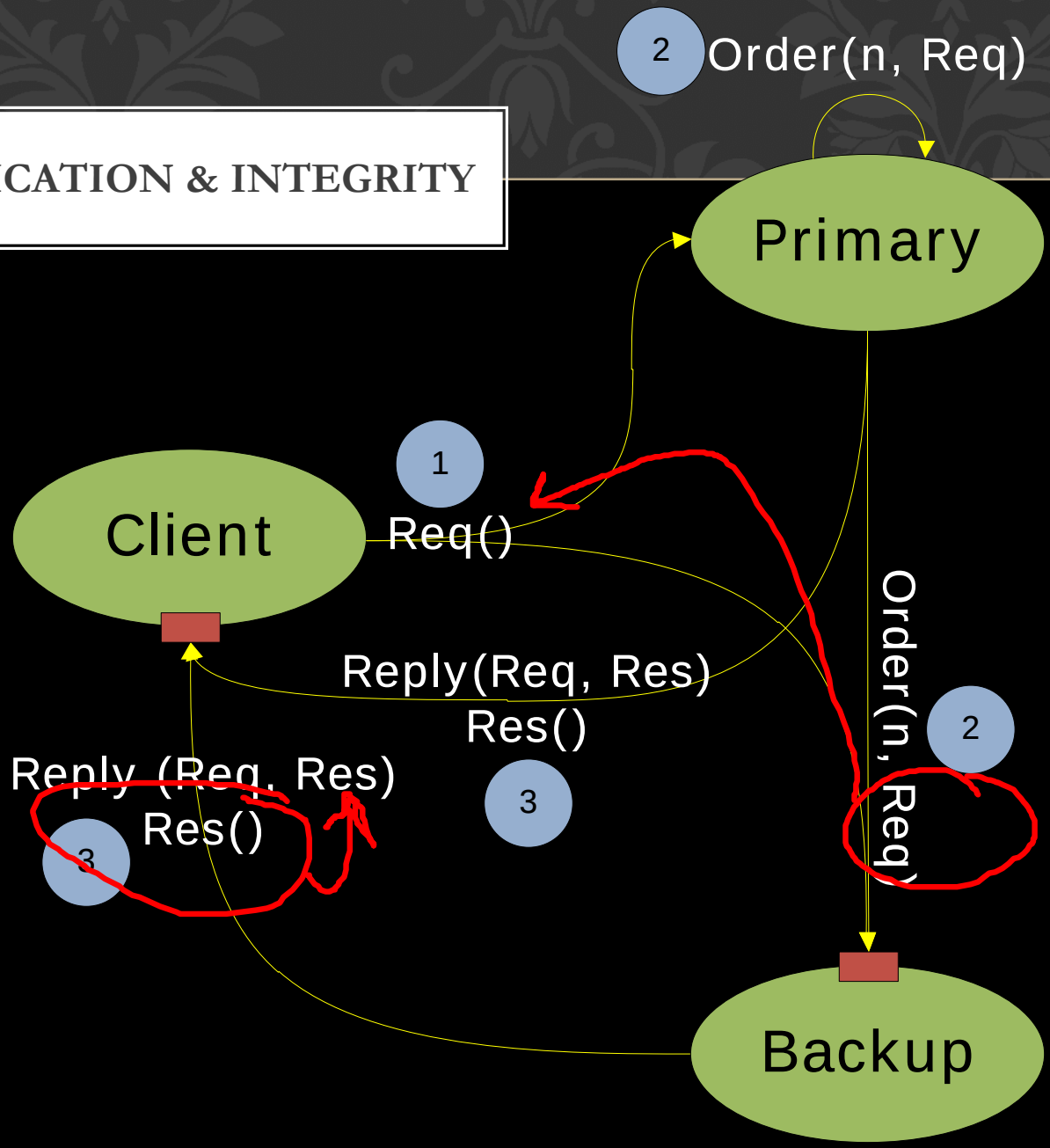

Primary
Client
Reply(Req, Res)
Res()
Reply (Req, Res)
Res()
Backup


2
Order(n, Req)
Primary
1
Client
Req()
Order(n, Req)
2
Reply(Req, Res)
Res()
3
Reply (Req, Res)
Res()
3
Backup

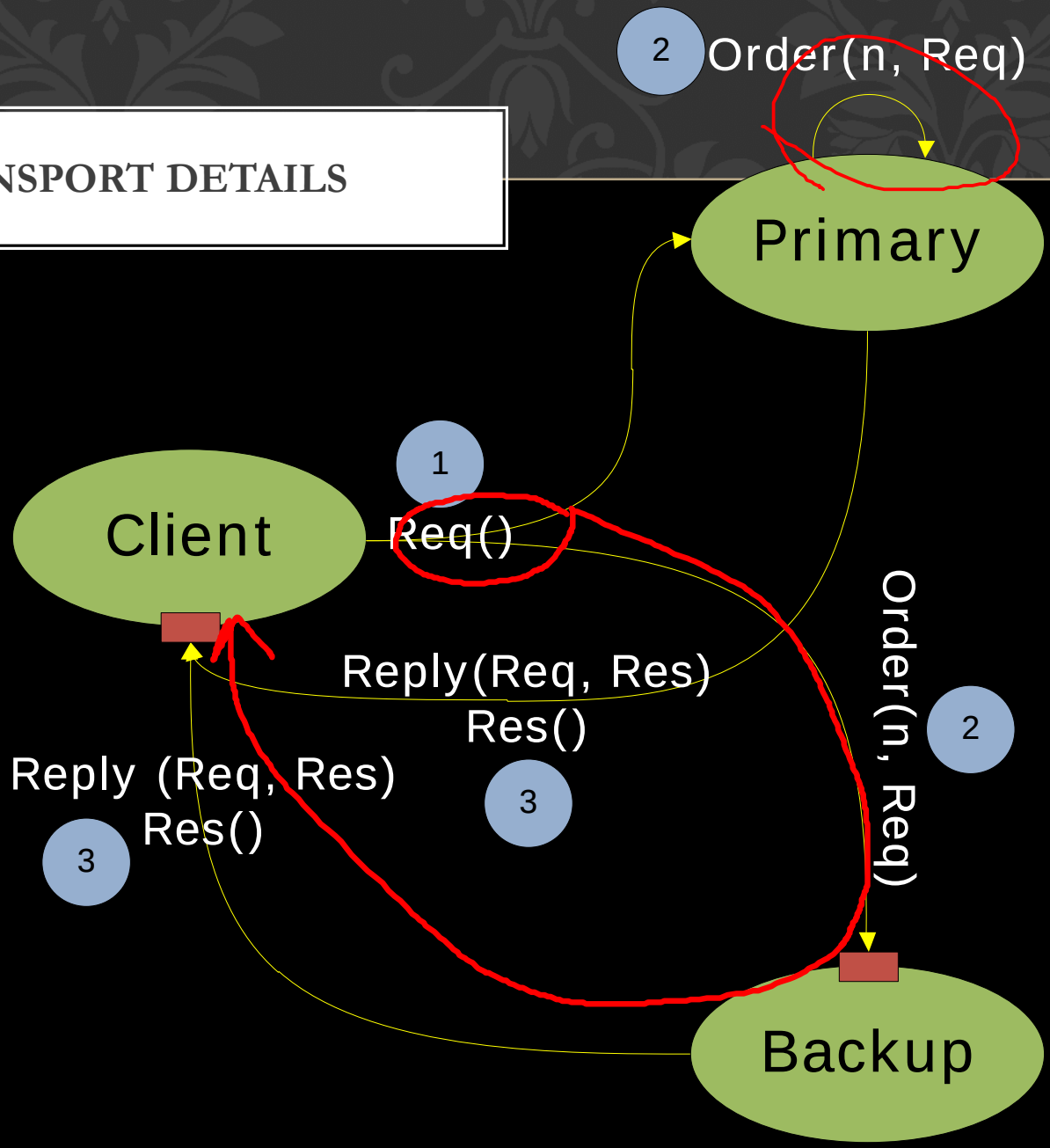
EVENT COORDINATION



AUTHENTICATION & INTEGRITY

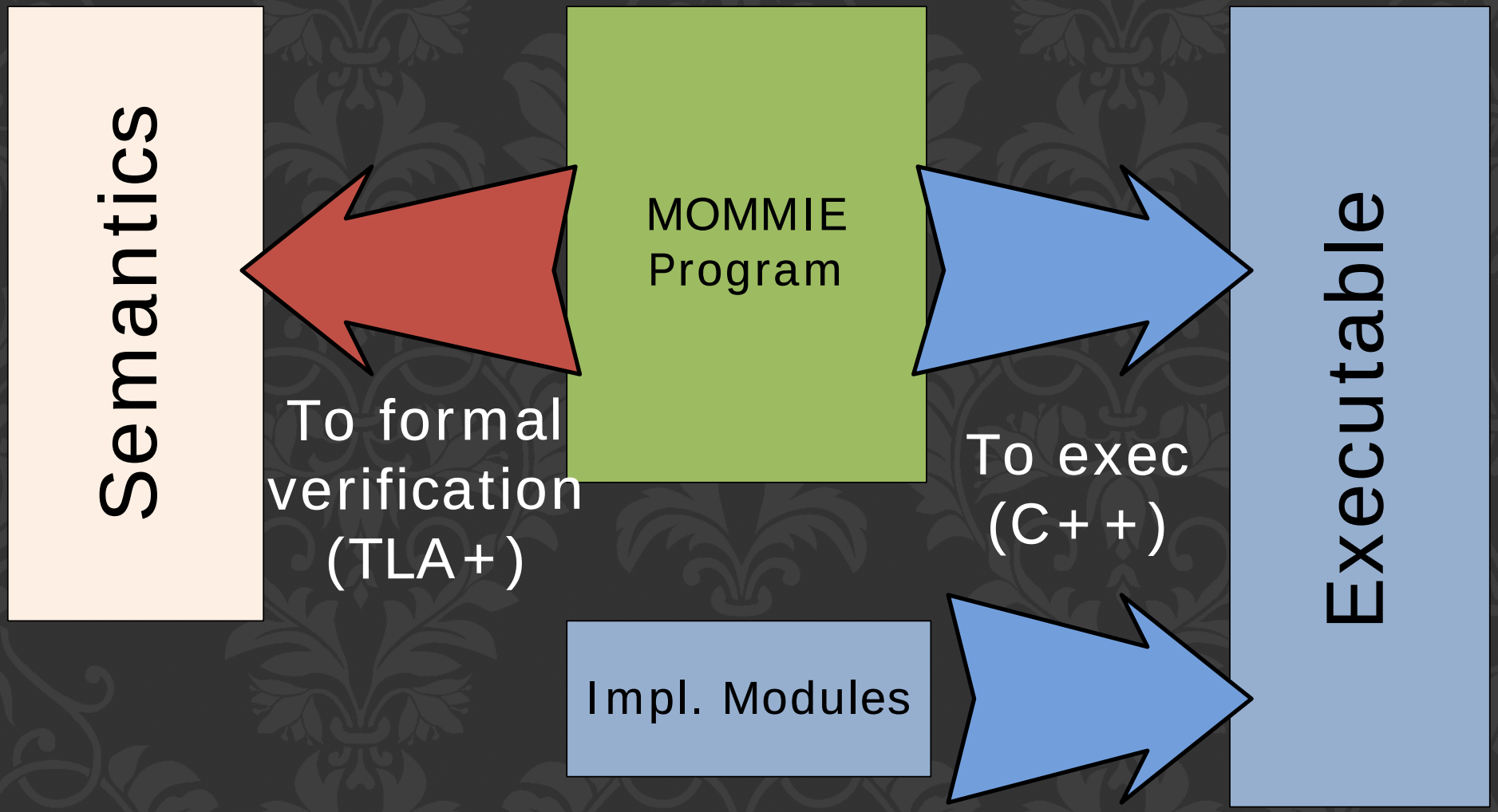


TRANSPORT DETAILS



Composition

*** O**PTIMIZED **M**ESSAGING
MIDDLEWARE FOR **I**NSECURE
ENVIRONMENTS
That's **MOMMIE** for you


Semantics
To formal verification (TLA+)
MOMMIE Program
To exec (C++)
Impl. Modules
Executable

THE MOMMIE STATEMENT POOL

Cached, disseminated among MOMMIE Pools

Statement algebra:

I said X & You said X ⇒ We said X

[Issuer, Verifier, Auditor] {c-structs}

Principals, can be groups

Addressing + Auth

Can be nested, with references

ELEMENTS OF A MOMMIE PROGRAM

Spec

```
WHEN [PRIMARY, ME, NOONE] Order order {{ m==req; seqNo==executeSeqNo; }}
ANDWHEN [c C == 1, ME NOONE] Request req
@@{
        Result res <- execute(req);
        PUBLISH [ME, c, NOONE] Reply(req, res);
        executeSeqNo <- executeSeqNo + 1;
@@}
```

Deployment

[Authentication]

```
a <-> b :  PlainTxt, HMAC-SHA1
```

[Non-repudiation]

```
x -> * :   TrustedThirdParty(z)
y -> * :   RSA
```

[Dissemination]

```
x,y -> * : OPTIMISTIC(a);
           OPTIMISTIC({a,b,c} >= 2);
```

ELEMENTS OF A MOMMIE PROGRAM

Spec

```
WHEN [PRIMARY, ME, NOONE] Order order {{ m==req; seqNo==executeSeqNo; }}
ANDWHEN [c C == 1, ME NOONE] Request req
@@{
        Result res <- execute(req);
        PUBLISH [ME, c, NOONE] Reply(req, res);
        executeSeqNo <- executeSeqNo + 1;
@@}
```

Deployment

```
[Authentication]
a <-> b :  PlainTxt, HMAC-SHA1

[Non-
x ->
y ->
```

```
[Dissemination]
x,y -> * : OPTI
```

Could be generated from mathematical optimizer

QUESTIONS I CAN ANSWER

Will it actually make writing provably robust programs easier?

Can it result in sufficiently fast/optimized code?

Can all functionality and optimizations compose safely?

What are we proving, really, for the whole system?

TLA+? Seriously?

Another language? Seriously?

"MOMMIE"? Seriously!

THANK YOU

Q?