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
A BRIEF ILLUSTRATION
Client

Req()

Primary

Backup
Primary

Client

Backup

Order(n, Req)
EVENT COORDINATION

Client → Primary
1. Req()
2. Order(n, Req)
3. Reply(Req, Res)
   - Res()

Primary → Backup
2. Order(n, Req)

Backup → Client
3. Reply(Req, Res)
   - Res()
AUTHENTICATION & INTEGRITY

1. Req()
2. Order(n, Req)
3. Reply(Req, Res)
4. Res()

Client
Primary
Backup
Composition
* OPTIMIZED MESSAGING MIDDLEWARE FOR INSECURE 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 \Rightarrow We said X

Issuer, Verifier, Auditor \{c-structs\}

Principals, can be groups
Addressing + Auth

Can be nested, with references
ELEMENTS OF A Mommie Program

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;
@@}

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

[Non-repudiation]
x -> * : TrustedThirdParty(z)
y -> * : RSA

[Dissemination]
x, y -> * : OPTIMISTIC(a);
OPTIMISTIC({a, b, c} >= 2);
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;
}@}

[Authentication]
a <-> b : Plaintxt, HMAC-SHA1
[Non-repudiation]
x -> * : TrustedThirdParty(z)
y -> * : RSA

ELEMENTS OF A MOMMIE PROGRAM

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?