

# Using Formal Methods at Google

**Thyla van der Merwe**

8 November, 2023

IETF 118

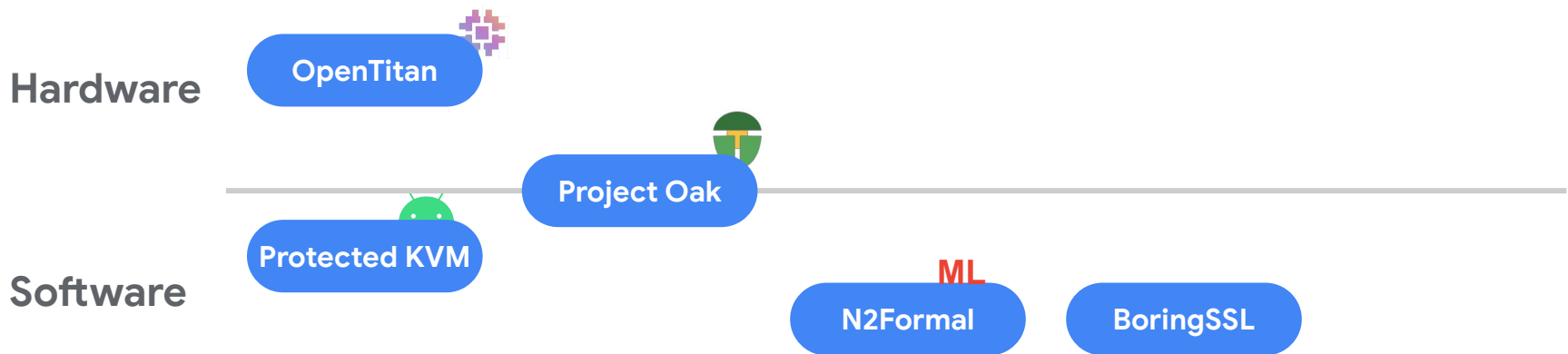
# Outline

- 🔒 What we currently do at Google
- 🔒 What we would like to do at Google
- 🔒 What would help to make that happen



# What we currently do at Google

(A sample of some verification efforts)



BY CAITLIN SADOWSKI, EDWARD AFTANDILJIAN, ALEX EAGLE,  
LIAM MILLER-CUSHON, AND CIERA JASPAN

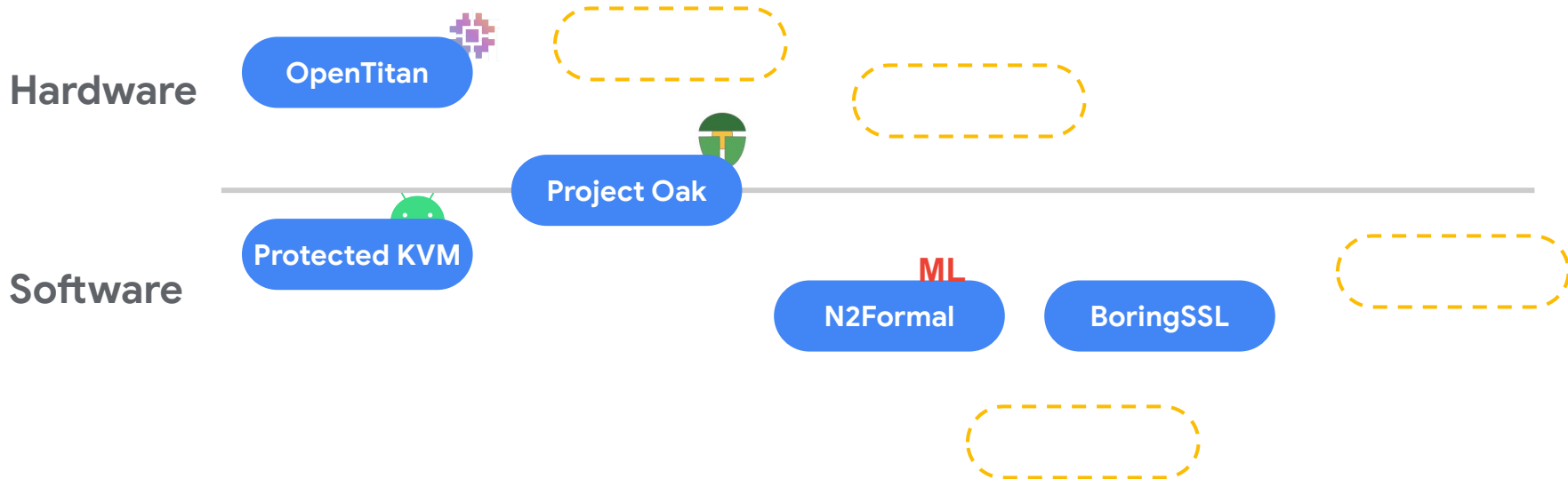
**Lessons  
from Building  
Static Analysis  
Tools at Google**

*some formal methods  
here, not sophisticated*

COMMUNICATIONS OF THE ACM | APRIL 2018 | VOL. 61 | NO. 4

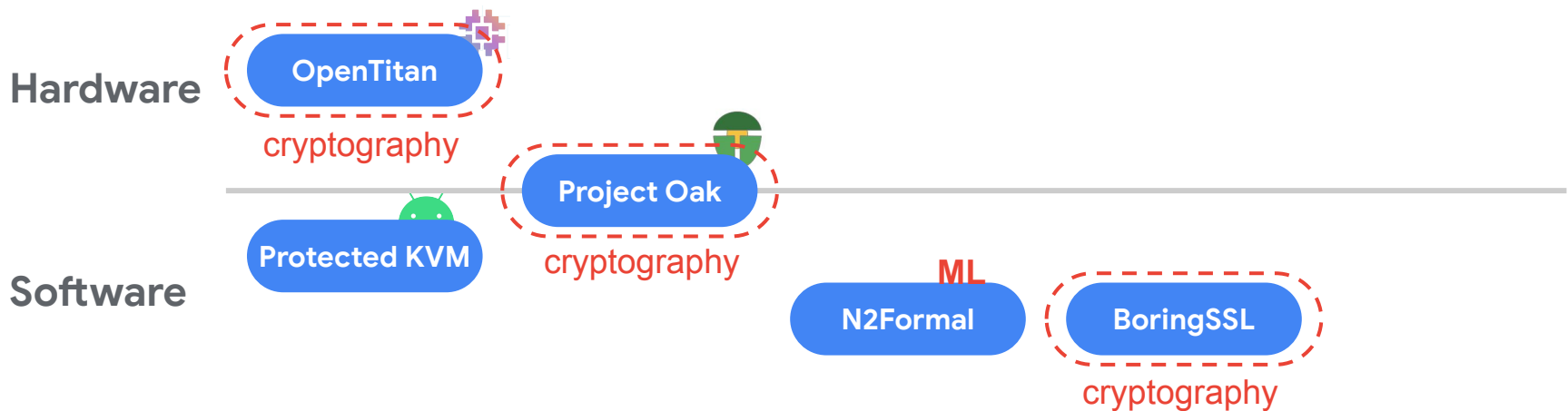
# What we currently do at Google

(A sample of some verification efforts)



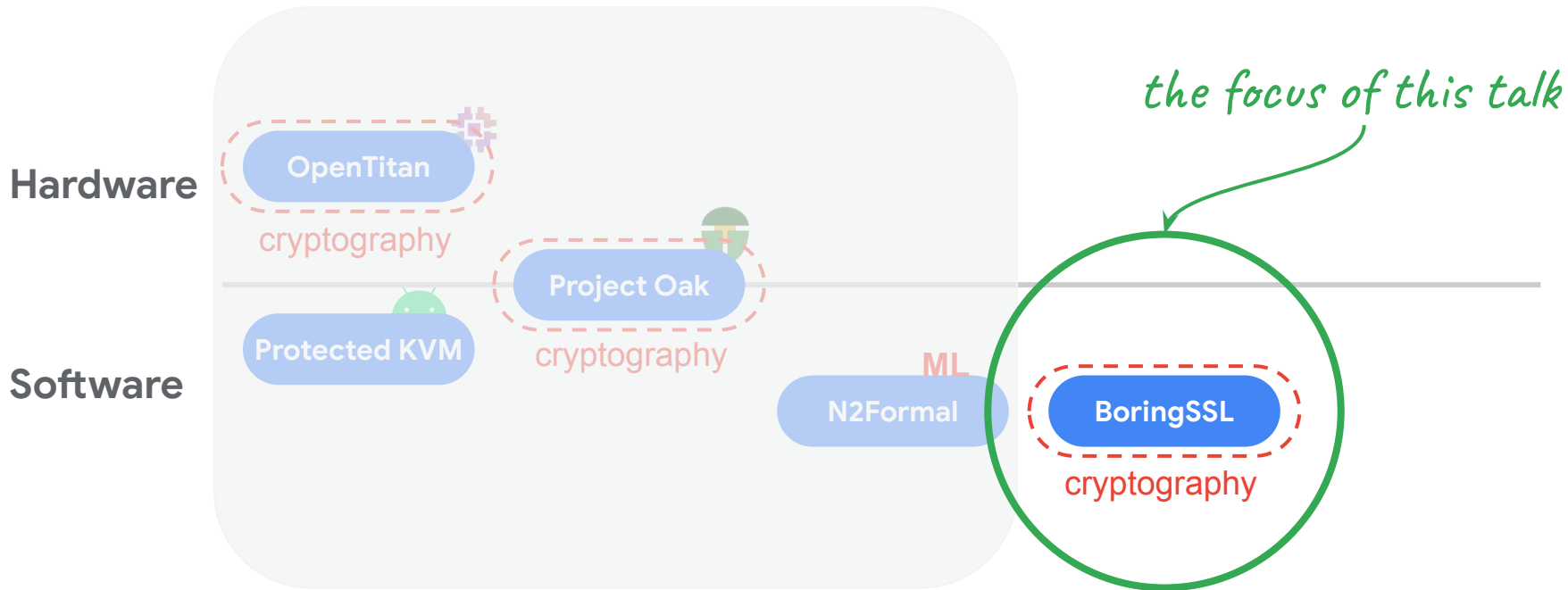
# What we currently do at Google

(A sample of some verification efforts)



# What we currently do at Google

(A sample of some verification efforts)



# What we currently do at Google



**ISE Formal Team**



Andres Erbsen  
Software Engineer  
Tech Lead



Thyla van der Merwe  
Engineering Manager



Bill Harris  
Software Engineer  
Contributor



Brian McSwiggen  
Software Engineer  
Contributor



Jade Philipoom  
Software Engineer  
Contributor



Lukas Zobernig  
Software Engineer  
Contributor

*with input from senior tech leads*

# What we currently do at Google

Build formally verified security-critical software and systems for Google



Mitigate common and subtle cryptography vulnerabilities proactively





# What we currently do at Google

## Software

Cryptographic  
Libraries

## Hardware

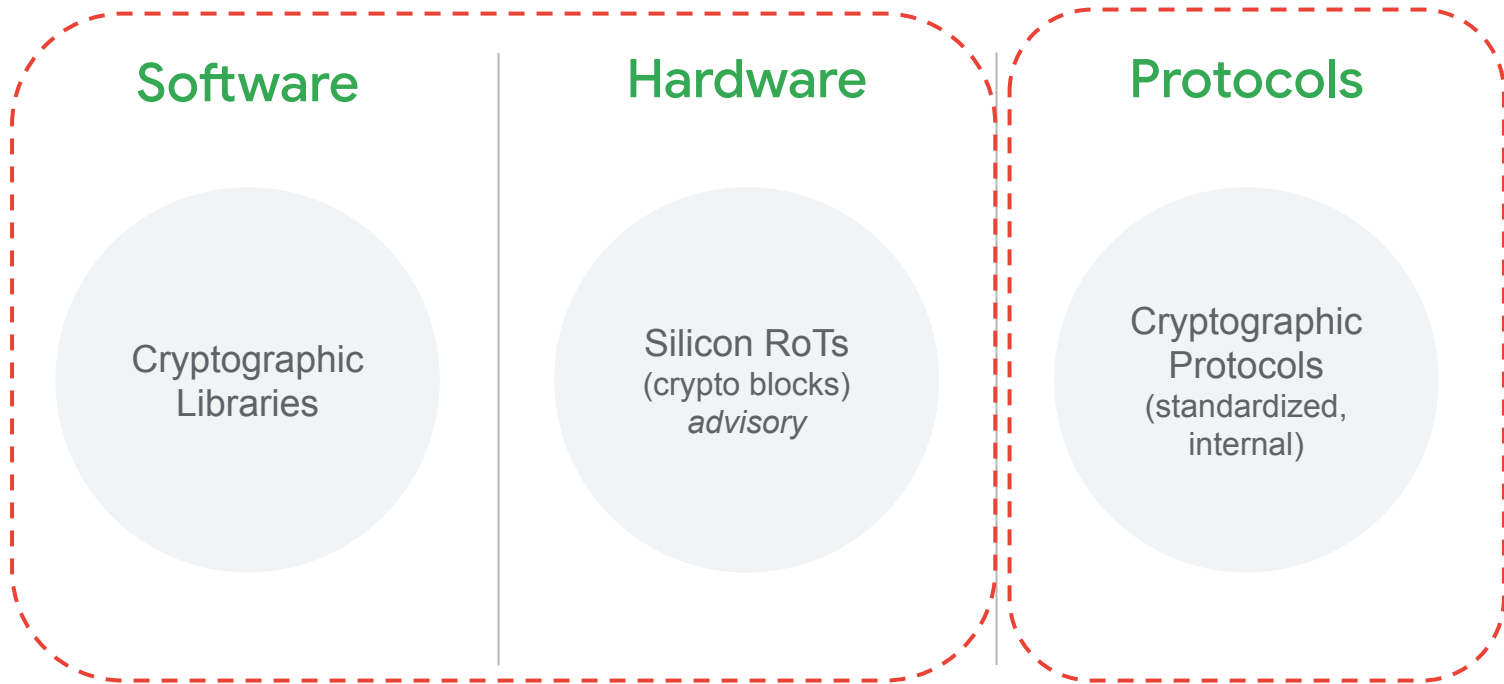
Silicon RoTs  
(crypto blocks)  
*advisory*

## Protocols

Cryptographic  
Protocols  
(standardized,  
internal)

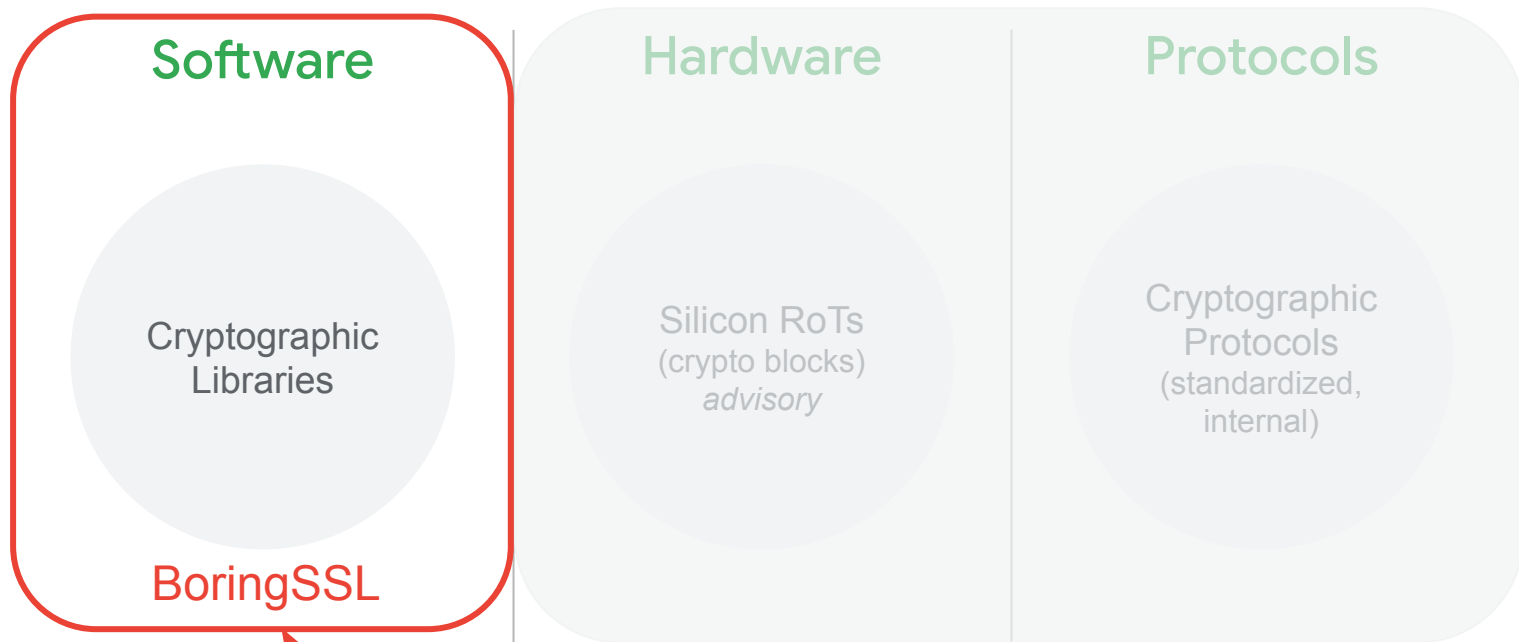
# What we currently do at Google

*verifying protocol design*



*producing verified code*

# What we currently do at Google



*this is the main focus right now*

# What we currently do at Google

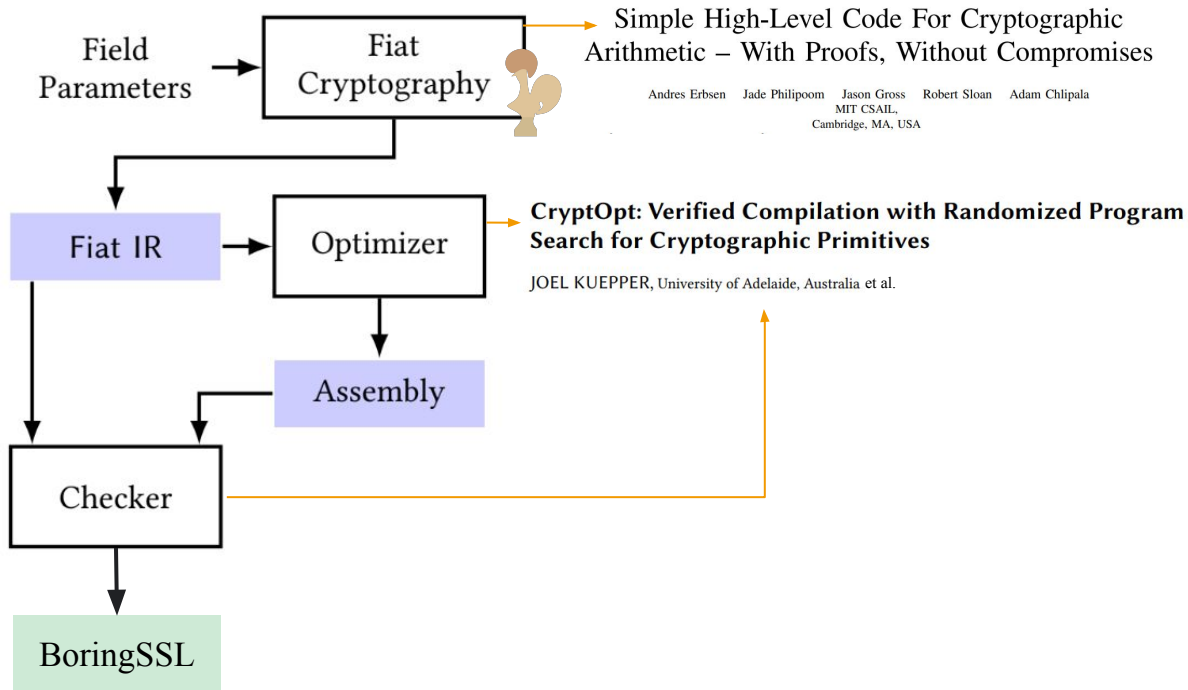


figure borrowed/adapted from CryptOpt paper

# What we currently do at Google

BoringSSL Gerrit							
CHANGES ▾		DOCUMENTATION ▾		BROWSE ▾		Q status:closed author:andreaser@google.com status:merged	
Subject	Owner	Reviewers	Repo	Branch	Updated	Size	Status
do not call memcpy directly in curve25519_64_adx.h	Andres Erbsen	David	boringssl	master	Oct 31	XS	Merged
Add saturated X25519 for x86_64+ADX running Linux	Andres Erbsen	David	boringssl	master	Oct 30	L	Merged
Add table-independent x86+adx asm for P-256	Andres Erbsen	David	boringssl	master	Oct 05	L	Merged
Credit CryptOpt in third_party/ fiat/README.md	Andres Erbsen	Adam, Bob Beck	boringssl	master	Jun 20	S	Merged
Use ADX asm for Curve25519 base-point multiplication	Andres Erbsen	David	boringssl	master	Jun 06	M	Merged
Use packed representation for large Curve25519 table	Andres Erbsen	David	boringssl	master	Jun 06	XL	Merged
Constant-time test that X25519 has a single path.	Andres Erbsen	David	boringssl	master	Jun 01	M	Merged
Generate 64-bit Curve25519 and P256 code for MSVC	Andres Erbsen	David	boringssl	master	Apr 19, 2023	XL	Merged
Add links to proofs of elliptic curve formulas.	Andres Erbsen	David	boringssl	master	Dec 22, 2017	S	Merged

## Formally verified elliptic curve operations in BoringSSL

- Curve25519, C and asm with ~20% performance improvement
- P-256 field arithmetic

# What we would like to do at Google

## Software

coverage of more libraries  
coverage of new algorithms

Cryptographic  
Libraries

## Hardware


Silicon RoTs  
(crypto blocks)  
*advisory*

## Protocols

coverage of Google-critical protocols  
useful to security reviews

Cryptographic  
Protocols  
(standardized,  
internal)

More of this, more quickly?



“The initial learning is quite tricky, and the documentation not all that great.”


ISE Formal Contributor, P1

“In terms of readability, documentation, and debuggability of proof checkers lag behind most other code I interact with.”

ISE Formal Contributor, P2

“Non-backward-compatible updates to theorem provers seem common [but necessary] ... it can still make working with only-mildly-out-of-date forks a pain.”

ISE Formal Contributor, P3




“It’s super helpful talking to someone that already knows their way around the various tricks and pitfalls. ”

ISE Formal Contributor, P1

“There's also a lot of infrastructure needed for any project at scale ... which compounds the documentation problem, since it's project-specific and under development. There's probably some room to standardize more of this infra but that's of course a challenging problem in its own right.”

ISE Formal Contributor, P2





“Experienced pros can probably write proof scripts The Right Way in one shot, but I almost always have to first write out a manual script and then go back to clean it up. Any automated support (either cleaning up an existing script or auto-suggesting as I write) would save a ton of time.”

ISE Formal Contributor, P3

# More accessible to more engineers?

01

**Tools and toolchains are very complicated to use**

They are still largely academic PoCs

Need to be highly skilled, or have access to someone who is

No corporate-level investment in producing polished tooling

02

**Easy-to-follow ramp up documentation is difficult to come by, or is lacking**

Documentation quality varies across tools

Sometimes good for simple examples but not all that useful for more complicated use cases

03

**Benefits are not always easy to sell**

We don't want to use code we can't maintain; proofs and code need to be maintainable

How do we know that a model is correct and appropriate?

# What would help...

01

## More usability research in this space

Work in the area of cryptographic libraries and APIs - can we extend this to formal methods tools?

More SoK-type work covering the pros and cons of the different tools and toolchains (is already some work here)?

02

## Improved documentation and debugging

Descriptive and useful error logging/feedback for both non-interactive and interactive tools is vital to a good user experience

Documented limitations are better than surprises

03

## Stable, well-maintained releases

True of any tooling/software that needs to be used at scale, and/or for critical projects

# What would help...

01

## More usability research in this space

Work in the area of cryptographic libraries and APIs - can we extend this to formal methods tools?

More SoK-type work covering the pros and cons of the different tools and toolchains (is already some work here)?

02

## Improved documentation and debugging

None of this is easy

How can we help?

Industry + Academia

Descriptive and useful error logging/feedback for both non-interactive and interactive tools is vital to a good user experience

03

## Stable, well-maintained releases

True of any tooling/software that needs to be used at scale, and/or for critical projects

# Thank you



Andres Erbsen



Bill Harris



Brian McSwiggen



Jade Philipoom



Lukas Zobernig