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)

Hardware
- OpenTitan
- Project Oak
- Protected KVM

Software
- N2Formal
- BoringSSL

some formal methods here, not sophisticated
What we currently do at Google

(A sample of some verification efforts)

Hardware
- OpenTitan
- Protected KVM

Software
- Project Oak
- N2Formal
- BoringSSL
- ML
What we currently do at Google

(A sample of some verification efforts)
What we currently do at Google

(A sample of some verification efforts)

Hardware
- OpenTitan
- Protected KVM

Software
- Project Oak
- N2Formal

the focus of this talk
What we currently do at Google

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

Software
Cryptographic Libraries

Hardware
Silicon RoTs (crypto blocks) *advisory*

Protocols
Cryptographic Protocols (standardized, internal)

producing verified code
What we currently do at Google

Software

- Cryptographic Libraries

- BoringSSL

Hardware

- Silicon RoTs (crypto blocks) advisory

Protocols

- Cryptographic Protocols (standardized, internal)

*this is the main focus right now*
What we currently do at Google

Software

Cryptographic Libraries

BoringSSL

figure borrowed/adapted from CryptOpt paper
What we currently do at Google

- 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
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What would help...</strong></td>
<td><strong>None of this is easy</strong></td>
<td><strong>How can we help?</strong></td>
</tr>
<tr>
<td>01</td>
<td>More usability research in this space</td>
<td>None of this is easy</td>
</tr>
<tr>
<td></td>
<td>Work in the area of cryptographic libraries and APIs - can we extend this to formal methods tools?</td>
<td>Industry + Academia</td>
</tr>
<tr>
<td></td>
<td>More SoK-type work covering the pros and cons of the different tools and toolchains (is already some work here)?</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>Improved documentation and debugging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Descriptive and useful error logging/feedback for both non-interactive and interactive tools is vital to a good user experience</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>Stable, well-maintained releases</td>
<td></td>
</tr>
<tr>
<td></td>
<td>True of any tooling/software that needs to be used at scale, and/or for critical projects</td>
<td></td>
</tr>
</tbody>
</table>
Thank you

Andres Erbsen  Bill Harris  Brian McSwiggen  Jade Philipoom  Lukas Zobernig