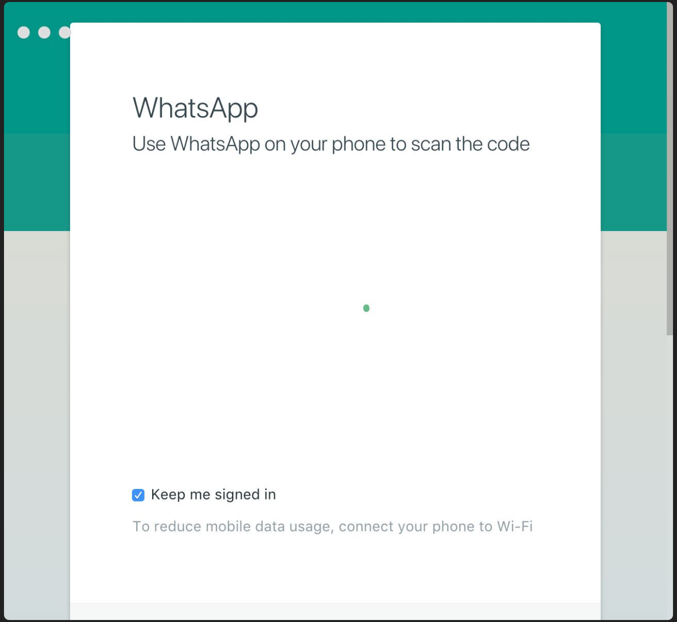
# Verifying Cryptographic Web Applications

Karthikeyan Bhargavan
Inria

# Cryptographic Apps in JavaScript?

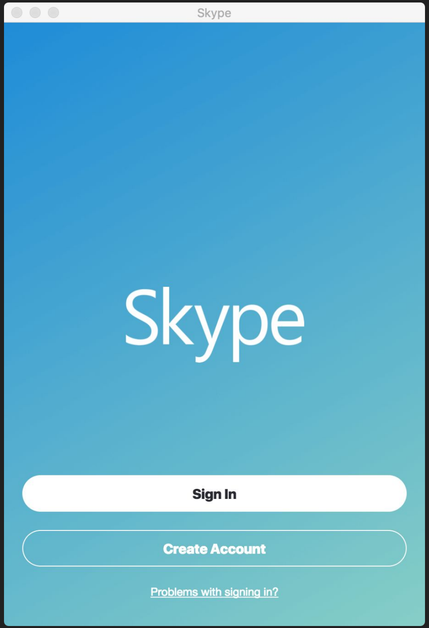


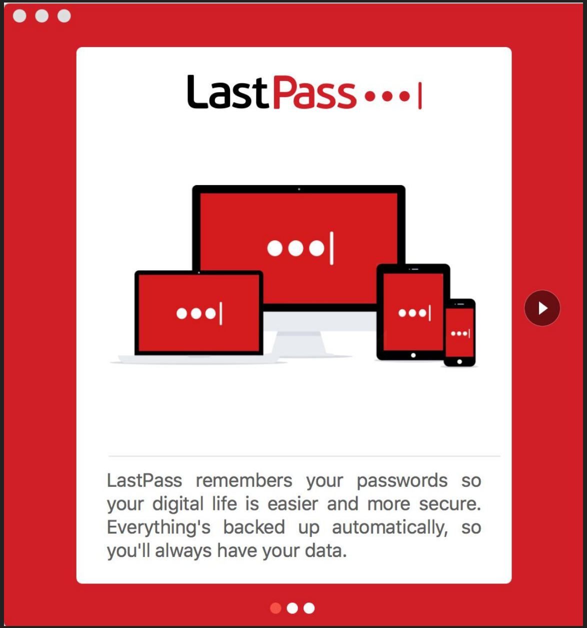
WhatsApp

Use WhatsApp on your phone to scan the code

☑ Keep me signed in

To reduce mobile data usage, connect your phone to Wi-Fi



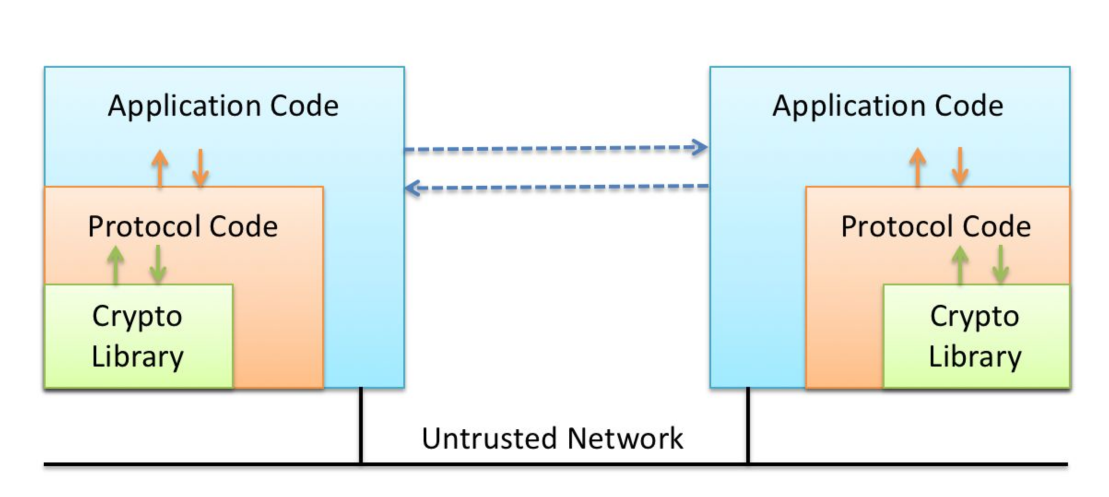


LastPass remembers your passwords so your digital life is easier and more secure. Everything's backed up automatically, so you'll always have your data.

Each app embeds a crypto protocol for end-to-end security

- Little or no reliance on a trusted server, all protections are in JavaScript

# Crypto Web Application Architecture



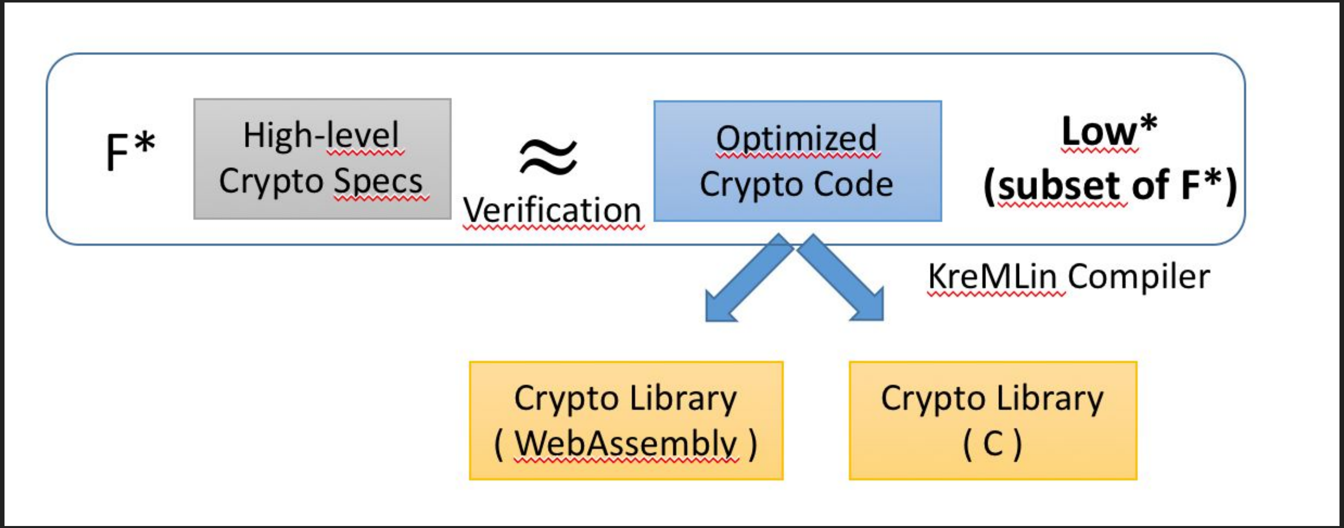
Application
Skype (React/Electron/Node)

Crypto Protocol
Signal (JavaScript)

Crypto Library
curve25519 (JS compiled from C)
window.crypto (native C library)

Challenge: *Verify the crypto library and protocol code and protect both from application bugs*

# HACL*: Verified Crypto in C and WASM

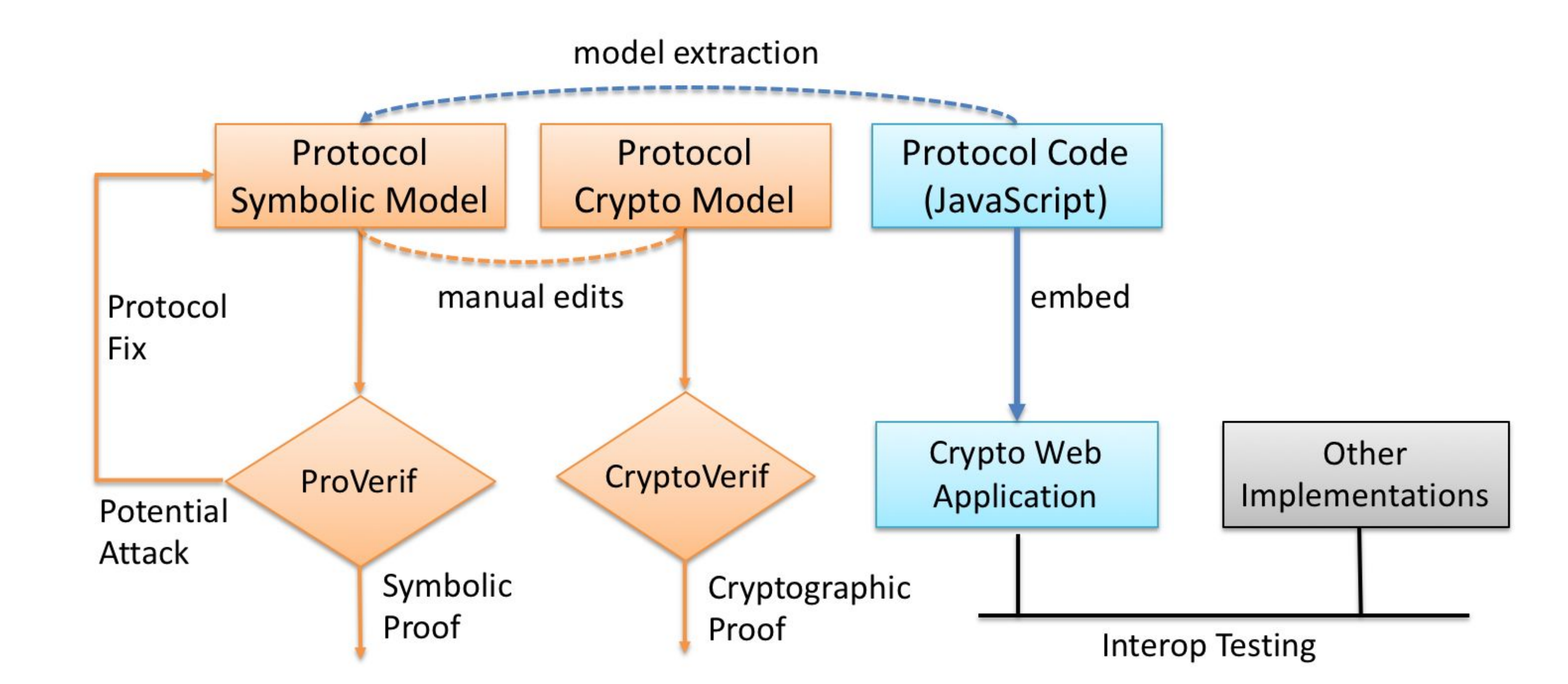


- Verified modern (+ upcoming) crypto primitives

  Chacha20, Poly1305, Curve25519, Ed25519, SHA-3, RSA-PSS, PQ crypto etc
- We prove correctness, memory safety, secret independence

  Open challenge: propagate side channel protections from WASM to assembly

# Verifying crypto protocols in JavaScript



Methodology applied to Signal, TLS 1.3

Open challenge: formal guarantees for model extraction tools

# Protecting Crypto from Application Bugs

- Write protocol and crypto in a “defensive” style

  Self-contained, no calls to application code or untrusted libraries

  Provide clean APIs that hide (most) secrets from application

- Write application in statically typed Flow/Typescript

  Prevents common errors, can enforce correct use of protocol API I

  Open challenge: can we enforce type abstraction for secrets

- Write and verify crypto/protocol/application in F*

  Compilers to C, JavaScript, WebAssembly

  Open challenge: formal guarantees for compilers and typechecker

# Conclusion

- Many exciting research challenges in verifying cryptographic web applications
  - Typed sub-languages (WASM, Flow) help verification a **LOT**
- Additional language support would be a big help
  - Ask: Constant-time guarantees for WebAssembly
  - Ask: Enforce type abstraction in Flow
- Composing verified and unverified JS is a challenge
  - Ask: Link formal semantics of Wasm and JS