Verified Security

Where are we and where should we go?

Paul Vines

Outline

1. Overview of Existing Security Verification Projects
2. Discussion
3. The State of Side-channels
4. Conclusion

Current Works

Cryptographic Primitives

Protocols (TLS)

Secure Apps

Current Works: Crypto Primitives

Tools: *FCF, EasyCrypt…*

RSA-OAEP (EasyCrypt) -- Crypto→ Assembly

HMAC (FCF) -- Crypto→ Assembly

SHA256 (FCF) -- Crypto* → Assembly

HMAC, SHA, RSA (Dafny) -- Functional → Assembly

Current Works: Protocols

SSH (CryptoVerif) -- Crypto → OCaml

miTLS (F7) -- Crypto → .NET

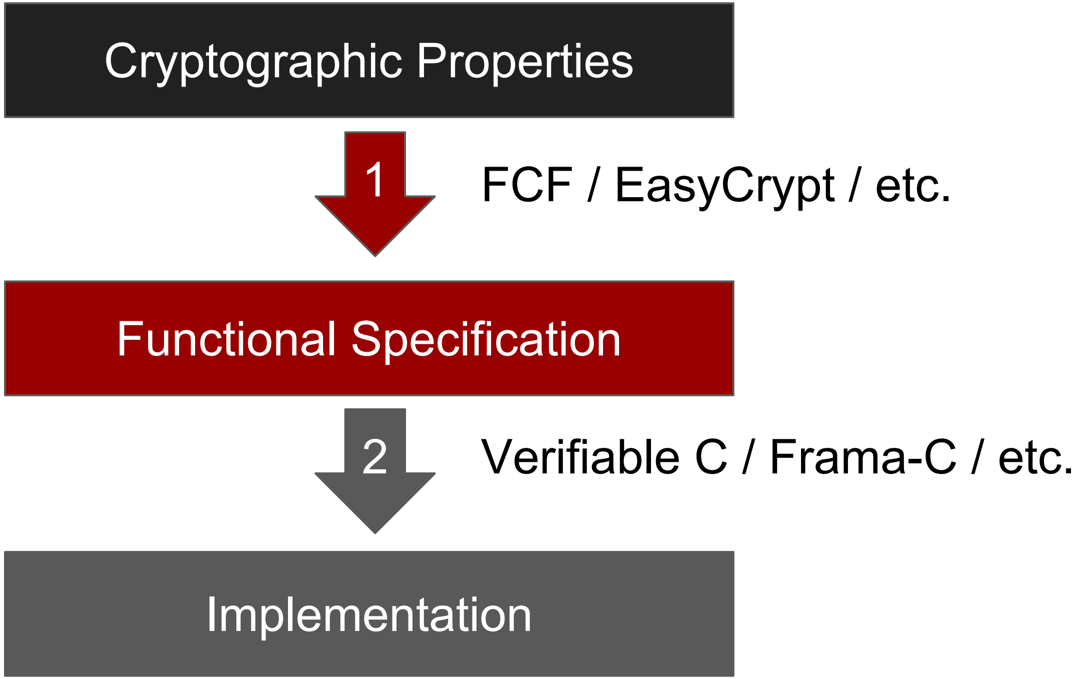
Current Works: Full App Security

Quark (Coq)

Ironclad (Dafny)

Discussion
Where should we go?

Security Verification in 2-Phases



Cryptographic Properties

1 FCF / EasyCrypt / etc.

Functional Specification

2 Verifiable C / Frama-C / etc.

Implementation

Crypto Primitives: Not Worth a Retrofit

- Difficult to capture cryptographic properties
- Low frequency of errors
- *If* it is done, the Ironclad approach of going from FIPS spec to implementation makes more sense
- *However*, we should push cryptographers to use verification-friendly tools (such as FCF/EasyCrypt) when creating new algorithms
  - Cheapens pushing verification out to these primitives in the future
  - Grants greater assurance to algorithmic correctness (Dual EC DRBG)

TLS: When Will The Madness End?

- Many vulnerabilities in the past (and present…)
- 2013 miTLS
- 2014 Finding flaws in miTLS
- 2015 Finding flaws in implementations except miTLS and PolarSSL
- So, we solved it?!

TLS is/would be a Big Win

- TLS is the security behind all network applications most users experience
- History of attacks suggests full concept→implementation verification
  - Vulnerabilities in both *implementation* **and** *specification* that had led to attacks
- Recent history also suggests we can’t know if we’ve succeeded
- **Is miTLS good enough?**
- **Does PolarSSL show verification is not getting us that much?**
- **When will we get a crypto → assembly verified protocol?**

Authentication

Authentication mechanisms are also a source of significant vulnerabilities.

Certificate handling is a juicy target, but also complicated by the ridiculousness of the X509 spec

The State of Side-channels

Side-channels, Now

- Timing Side-channels
  - Lucky 13
  - Secure Coding Methodologies
- Emissions Side-channels
  - PITA attack
  - Physical space dependent

Verifying Absence of Side-channels

- More important to verify than correctness for crypto primitives?
- Timing Side-channels
  - Current works extending CompCert to verify constant-time formulation of programs
  - MAC-then-Encode-then-Encrypt with Cipher-Block-Chaining (MEE-CBC)
  - Making this easier would be nice
- Emissions Side-channels -- No Solutions So Far

Conclusions

What should be done

- Develop a featureful cryptographic framework based on Coq (FCF)
    - Get Cryptographers to use it
- Verify high-level protocols (TLS) from cryptographic properties to small-TCB implementations (assembly)
- Incorporate more esoteric security needs (constant time execution) into implementation verification

?

Retrofitting vs. Rebuilding

Retrofit by creating verified versions of existing protocols

Or

Rebuild by creating and verifying new protocols and include fallback options for compatibility

Retrofitting is the way forward

- Allows a verified version of security software to be adopted gradually and at low-cost to the user.

Retrofitting also avoids the risk of eliminating spec-level bug-features