Our research

Solving security problems
- programming securely with cryptography

Devising formal methods

Developing practical tools and systems

- miTLS, HACL*, ProVerif, CryptoVerif, ProScript, CryptoCat, QuickChick, ...
Finding attacks in TLS
Current team

Researchers (6)

PostDocs (2)

PhD Students (4)

Interns (4)

Visitors (3)

Researchers

PostDocs

PhD Students

Interns

Visitors

Engineers (2)

Diverse and international 11 nationalities Our working language is English

Collaborators
Use formal methods to achieve security of critical software

HTTPS stack
Modern cryptographic library
Secure messaging app
Web browser core
Compilers & monitors
TCP/IP network stack...
Tools for analyzing abstract models of crypto protocols

ProVerif

• ProVerif
  - symbolic model (Dolev-Yao)
  - fully automatic, efficient, precise, produces attack traces
  - wide range of crypto primitives and properties

CryptoVerif

• CryptoVerif
  - computational model
  - semi-automatic: sequence of crypto games
  - exact security: bound on attack probability

Recent case studies

• Recent case studies: TLS 1.2 & 1.3, Signal, ARINC823
  - upcoming TLS 1.3: big redesign, new hope for verification
From verifying protocol models to actual implementations

Protocol models

Protocol implementations

- Capture core behavior: succinct, abstract, high-level
- Great for finding logical flaws [3Shake] and incorrect use of crypto [Lucky13] early in the protocol design
- e.g. TLS 1.2 & 1.3 in ~1000 lines of ProVerif (best paper at Oakland'17)

- Large software projects: interoperable, efficient
- Concrete packet formats, multiple protocol modes
- Support legacy ciphersuites, complex APIs, composable subprotocols

- More attacks: message parsing [HeartBleed], state machine [FREAK]
Verified reference implementation of TLS 1.2 & 1.3
Microsoft Research and Inria
Built on top of our HACL* crypto library
Towards a verified HTTPS stack
HTTPS ecosystem critical, complex
HTTPS ecosystem critical, complex and broken

• 20 years of attacks & fixes
  fff    ff
  i
  ff   r
  lr   rr
  r
  i
  ff
  R

• Mainstream implementations
  r . i . .
  r i i
Strong verified security

Widespread deployment

- efficiency
- interoperability
- drop-in
Everest stack verified with

Functional programming language

Semi-automated verification using SMT

Interactive verification using dependent types
Is verified code secure in practice?

Unsafe languages

Web browser/server
2,000,000+ LOC

Insecure interoperability

Ooops
Secure compilation

Secure interoperability with lower-level code

Dynamic enforcement, but at what cost?

in software, 10x? 100x? 1000x?

Micro-policies

new tagged hardware architecture

large metadata tag to each word

software defined. very flexible, fine-grained ] . . . fast ...

... average 10% runtime overhead
Use formal methods to achieve security of critical software

HTTPS stack
Modern cryptographic library
Secure messaging app
Web browser/server core
Compilers & monitors
TCP/IP network stack ...