Schrodinger's Squirrel

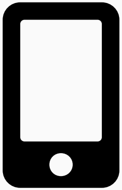
Formal security proofs in a post-quantum world

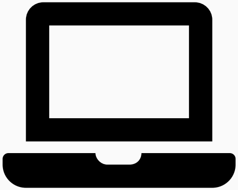
Charlie Jacomme
CISPA Helmholtz Center for Information Security

November 19, 2021

Formal Methods for Security and Privacy

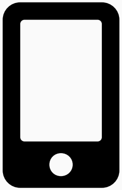
The beautiful and frightening technological revolution

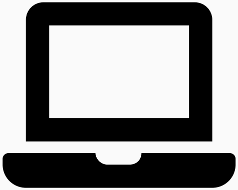




The beautiful and frightening technological revolution

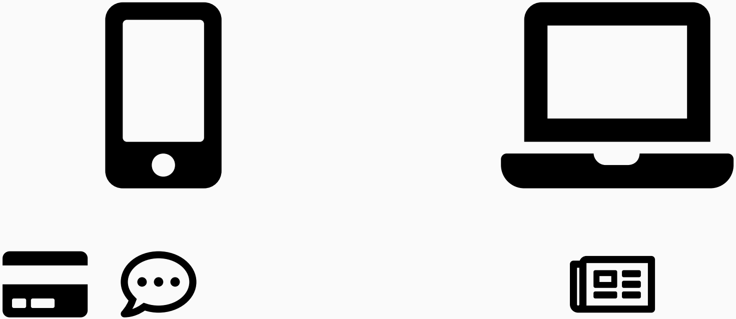
It's my life!



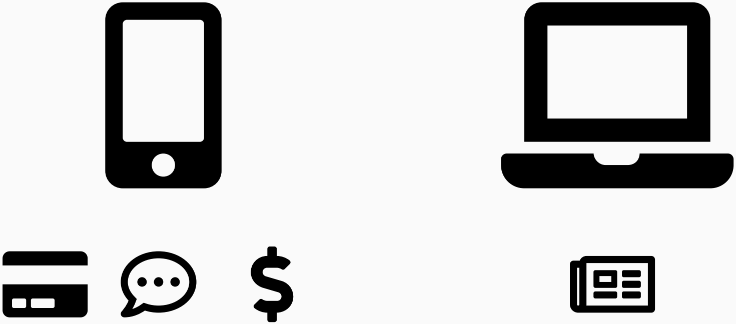


The beautiful and frightening technological revolution

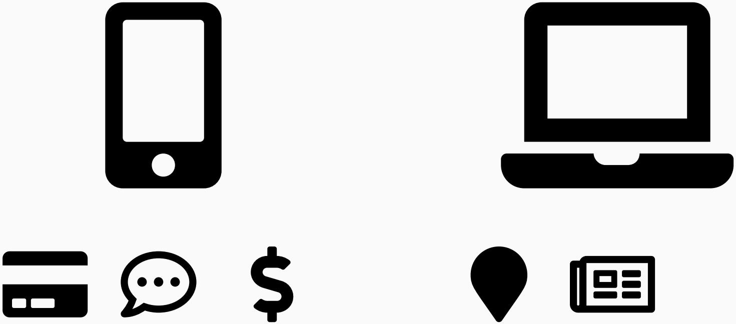
The beautiful and frightening technological revolution



The beautiful and frightening technological revolution



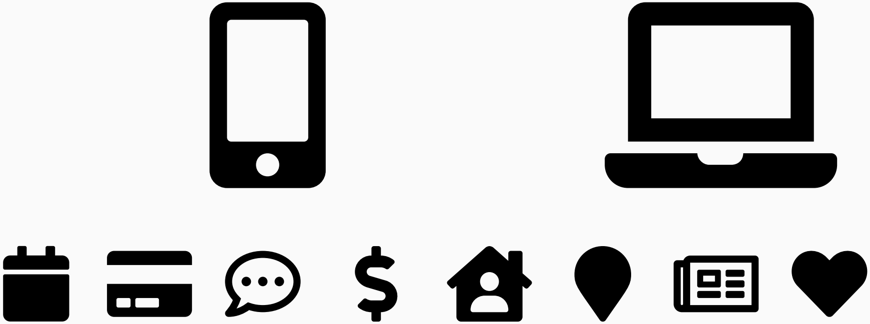
The beautiful and frightening technological revolution



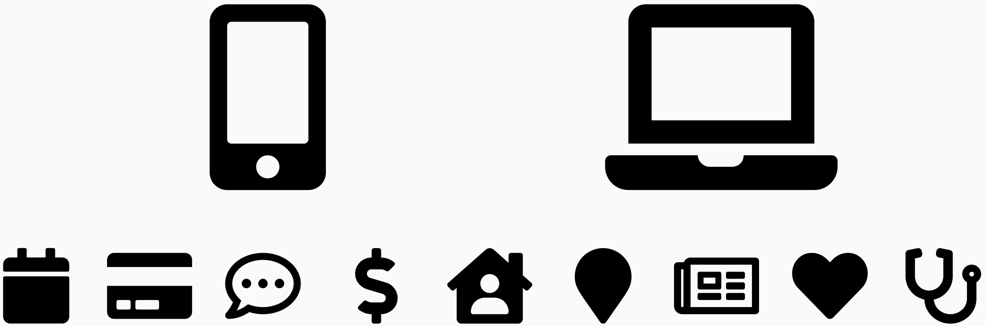
The beautiful and frightening technological revolution

The beautiful and frightening technological revolution

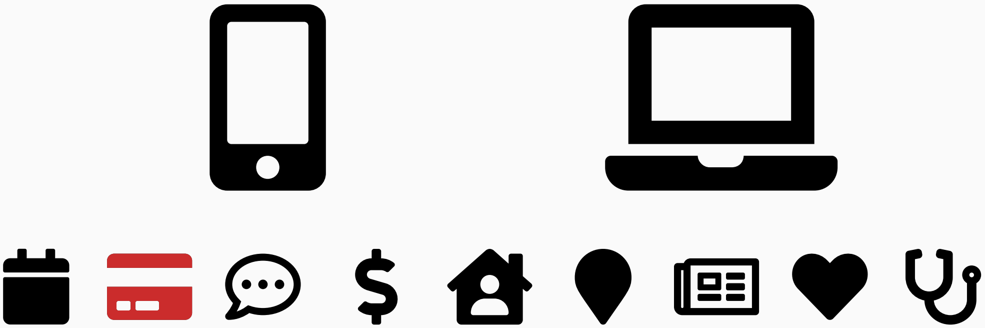
The beautiful and frightening technological revolution

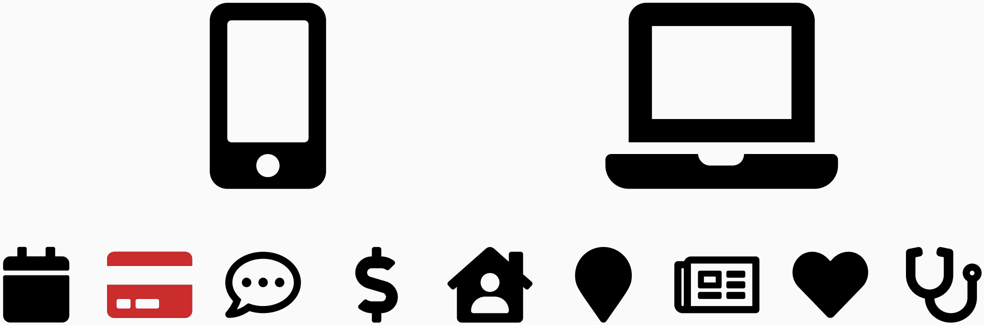


The beautiful and frightening technological revolution



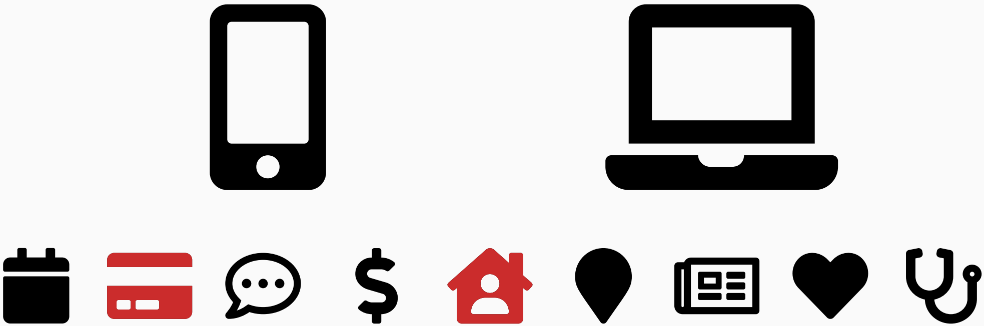
The beautiful and frightening technological revolution





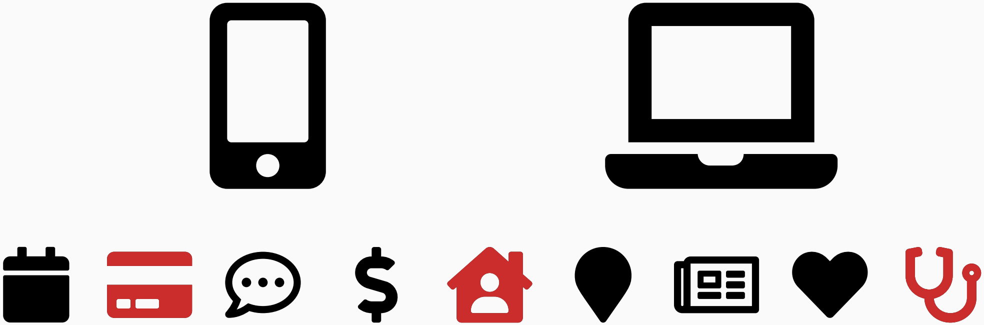
Sacrifice privacy in exchange of services...

The beautiful and frightening technological revolution



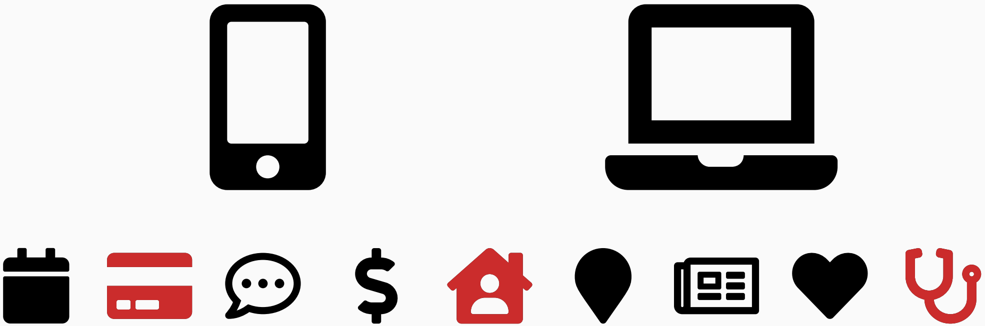
Sacrifice privacy in exchange of services. . .

The beautiful and frightening technological revolution



Sacrifice privacy in exchange of services…

The beautiful and frightening technological revolution



Sacrifice privacy in exchange of services...

but our data is used against us!

Some people even need privacy to survive:

- Reporters in dangerous countries.
- Homosexual in countries where it is punished by law (still 69 in the world...).
- Uighurs tracked through their smartphones in China.

If we can't have privacy, nobody can

Security and Privacy Matter !

Security and Privacy Matter !

We need:

- systems designed to provide security and privacy;

Security and Privacy Matter !

We need:

- systems designed to provide security and privacy;
- with guarantees that they do;

Security and Privacy Matter !

We need:

- systems designed to provide security and privacy;
- with guarantees that they do;
- used in practice.

The (first) difficulty

Protocols

SSH
TLS
GPG
...

The (first) difficulty

Primitives	Protocols
RSA	SSH
Elliptic curves	TLS
...	GPG
	...

The (first) difficulty

Implementation	Primitives	Protocols
C++	RSA	SSH
Java	Elliptic curves	TLS
Python	...	GPG
...		...

The (first) difficulty

OS	Implementation	Primitives	Protocols
	C++	RSA	SSH
	Java	Elliptic curves	TLS
	Python	...	GPG
	...		...

The (first) difficulty

Hardware	OS	Implementation	Primitives	Protocols
		C++ Java Python ...	RSA Elliptic curves ...	SSH TLS GPG ...

The (first) difficulty

Hardware	OS	Implementation	Primitives	Protocols	Users
		C++ Java Python ...	RSA Elliptic curves ...	SSH TLS GPG ...	

The (first) difficulty

If any link of the chain is broken, everything is.

Hardware	OS	Implementation	Primitives	Protocols	Users
		C++ Java Python ...	RSA Elliptic curves ...	SSH TLS GPG ...	

The (first) difficulty

Hardware	OS	Implementation	Primitives	Protocols	Users
		C++	RSA	SSH	
		Java	Elliptic curves	TLS	
		Python	...	GPG	
		...		...	

The goal

Since the 80's

Provide guarantees on the protocol assuming that the other layers are secure.

The goal

Since the 80's

Provide formal guarantees on the protocol assuming that the other layers are secure.

The goal

Since the 80's

Provide formal guarantees on the protocol assuming that the other layers are secure.

$\hookrightarrow$ a mathematical proof on an abstract model [Goldwasser,Micali,Dolev,Yao]

The goal

Since the 80's

Provide formal guarantees on the protocol assuming that the other layers are secure.

$\hookrightarrow$ a mathematical proof on an abstract model [Goldwasser,Micali,Dolev,Yao]

$$\forall \mathcal{A}.\ P \mid\mid \mathcal{A} \models \phi$$

The goal

Since the 80's

Provide formal guarantees on the protocol assuming that the other layers are secure.

$\hookrightarrow$ a mathematical proof on an abstract model [Goldwasser,Micali,Dolev,Yao]

$$\forall \mathcal{A}.\ P \mid\mid \mathcal{A} \models \phi$$

P - model of the protocol.

The goal

Since the 80's

Provide formal guarantees on the protocol assuming that the other layers are secure.

$\hookrightarrow$ a mathematical proof on an abstract model [Goldwasser,Micali,Dolev,Yao]

$$\forall \mathcal{A}.\ P \mid\mid \mathcal{A} \models \phi$$

P - model of the protocol.

$\mathcal{A}$ - attacker model

The goal

Since the 80's

Provide formal guarantees on the protocol assuming that the other layers are secure.

$\hookrightarrow$ a mathematical proof on an abstract model [Goldwasser,Micali,Dolev,Yao]

$$\forall \mathcal{A}.\ P \mid\mid \mathcal{A} \models \phi$$

P - model of the protocol.

$\mathcal{A}$ - attacker model

ϕ - security property

Second Difficulty - The modeling

Attacker Model

Second Difficulty - The modeling

Attacker Model

Computation Model

- Turing Machines or inference rules
- Assumptions on primitives (RSA)
- Timing attacks

Second Difficulty - The modeling

Attacker Model

Computation Model

- Turing Machines or inference rules
- Assumptions on primitives (RSA)
- Timing attacks

Compromise Model

- Malwares, Keylogger
- Phishing
- Long-term/ephemeral key reveal

Second Difficulty - The modeling

Attacker Model

Computation Model

- Turing Machines or inference rules
- Assumptions on primitives (RSA)
- Timing attacks

Compromise Model

- Malwares, Keylogger
- Phishing
- Long-term/ephemeral key reveal

Protocol Model

- Optional behaviours or parameters
- Modeling of parsing, serialization
- Communications channels

Second Difficulty - The modeling

Attacker Model

Computation Model

- Turing Machines or inference rules
- Assumptions on primitives (RSA)
- Timing attacks

Compromise Model

- Malwares, Keylogger
- Phishing
- Long-term/ephemeral key reveal

Security Properties

- Secrecy, PFS, PCS
- Authentication
- Unlinkability

Protocol Model

- Optional behaviours or parameters
- Modeling of parsing, serialization
- Communications channels

The goal

Strong guarantees

Get proofs of security, with all modelings as realistic as possible.

The goal

Strong guarantees

Get proofs of security, with all modelings as realistic as possible.

It is very very very very very difficult

We want to prove over realistic models that something is impossible, even when considering all possible attackers.

- Undecidable;
- complexity of proofs grows very quickly, and cannot be managed by hand.

The solution

Computer-Aided Cryptography (since 2000)

Tools that help us carry-out, verify or automate the proofs.

The solution

Computer-Aided Cryptography (since 2000)

Tools that help us carry-out, verify or automate the proofs.

But. . .

- Inherent trade-off between the realism and automation/proof-size;

The solution

Computer-Aided Cryptography (since 2000)

Tools that help us carry-out, verify or automate the proofs.

But. . .

- Inherent trade-off between the realism and automation/proof-size;
- no single tool will be the best at everything.

The landscape of computer-aided cryptography

Symbolic Tools | Computational Tools

The landscape of computer-aided cryptography

Symbolic Tools	Computational Tools
(Proverif,Tamarin,Deepsec, . . .)	(EasyCrypt, CryptoVerif, Squirrel . . .)

The landscape of computer-aided cryptography

	Symbolic Tools (Proverif,Tamarin,Deepsec, . . .)	Computational Tools (EasyCrypt, CryptoVerif, Squirrel . . .)
Attacker	Fixed set of computations	Turing Machines

The landscape of computer-aided cryptography

	Symbolic Tools (Proverif,Tamarin,Deepsec, . . .)	Computational Tools (EasyCrypt, CryptoVerif, Squirrel . . .)
Attacker	Fixed set of computations	Turing Machines
Compromise	Many	Few

The landscape of computer-aided cryptography

	Symbolic Tools (Proverif,Tamarin,Deepsec, ...)	Computational Tools (EasyCrypt, CryptoVerif, Squirrel ...)
Attacker	Fixed set of computations	Turing Machines
Compromise	Many	Few
Protocol	Full specification	Core parts in isolation

The landscape of computer-aided cryptography

	Symbolic Tools (Proverif,Tamarin,Deepsec, ...)	Computational Tools (EasyCrypt, CryptoVerif, Squirrel ...)
Attacker	Fixed set of computations	Turing Machines
Compromise	Many	Few
Protocol	Full specification	Core parts in isolation

State of the art

- Many tools used successfully, both to prove security or discover new vulnerabilities on complex systems.

The landscape of computer-aided cryptography

	Symbolic Tools (Proverif,Tamarin,Deepsec, . . .)	Computational Tools (EasyCrypt, CryptoVerif, Squirrel . . .)
Attacker	Fixed set of computations	Turing Machines
Compromise	Many	Few
Protocol	Full specification	Core parts in isolation

State of the art

- Many tools used successfully, both to prove security or discover new vulnerabilities on complex systems.
- Still many limitations, and still very difficult to work on realistic models.

What I have been doing

Theory - Make proofs easier for realistic models

Practice - Actually make proofs for realistic models

What I have been doing

Theory - Make proofs easier for realistic models

- Composition results to cut computational and symbolic proofs into modular pieces;
  [Comon, J., Scerri - CCS'20]

Practice - Actually make proofs for realistic models

What I have been doing

Theory - Make proofs easier for realistic models

- Composition results to cut computational and symbolic proofs into modular pieces;
  [Comon, J., Scerri - CCS'20]
- Automation of basic proofs steps in the computational world;
  [Barthe, J., Kremer - LICS'20, TOCL] & [BGJKS - CSF'19] & [BFGGJS - CCS'18]

Practice - Actually make proofs for realistic models

What I have been doing

Theory - Make proofs easier for realistic models

- Composition results to cut computational and symbolic proofs into modular pieces;
  [Comon, J., Scerri - CCS'20]
- Automation of basic proofs steps in the computational world;
  [Barthe, J., Kremer - LICS'20, TOCL] & [BGJKS - CSF'19] & [BFGGJS - CCS'18]
- A new computational tool allowing for easier proofs of complex protocols;
  Squirrel Prover [Baelde, Delaune, J., Koutsos, Moreau - S&P'21]

Practice - Actually make proofs for realistic models

What I have been doing

Theory - Make proofs easier for realistic models

- Composition results to cut computational and symbolic proofs into modular pieces;
  [Comon, J., Scerri - CCS'20]
- Automation of basic proofs steps in the computational world;
  [Barthe, J., Kremer - LICS'20, TOCL] & [BGJKS - CSF'19] & [BFGGJS - CCS'18]
- A new computational tool allowing for easier proofs of complex protocols;
  Squirrel Prover [Baelde, Delaune, J., Koutsos, Moreau - S&P'21]

Practice - Actually make proofs for realistic models

- Extensive analysis in Proverif of multi-factor authentication;
  6000 scenarios generated and verified in 5 minutes [Kremer, J. - CSF'18, TOPS]

What I have been doing

Theory - Make proofs easier for realistic models

- Composition results to cut computational and symbolic proofs into modular pieces;
  [Comon, J., Scerri - CCS'20]
- Automation of basic proofs steps in the computational world;
  [Barthe, J., Kremer - LICS'20, TOCL] & [BGJKS - CSF'19] & [BFGGJS - CCS'18]
- A new computational tool allowing for easier proofs of complex protocols;
  Squirrel Prover [Baelde, Delaune, J., Koutsos, Moreau - S&P'21]

Practice - Actually make proofs for realistic models

- Extensive analysis in Proverif of multi-factor authentication;
  6000 scenarios generated and verified in 5 minutes [Kremer, J. - CSF'18, TOPS]
- Modular analysis of SSH in Squirrel, with optional feature of agent forwarding;
  Carried out first in the composition paper and then in the Squirrel one.

Today's presentation

A new attacker model

The tools should be able to provide guarantees against quantum attackers.

Today's presentation

A new attacker model

The tools should be able to provide guarantees against quantum attackers.

- What changes with a quantum attackers?

Today's presentation

A new attacker model

The tools should be able to provide guarantees against quantum attackers.

- What changes with a quantum attackers?
- Can tools already provide guarantees about them?

Today's presentation

A new attacker model

The tools should be able to provide guarantees against quantum attackers.

- What changes with a quantum attackers?
- Can tools already provide guarantees about them?
- If not, what can we do to fix them?

What is the fuss about quantum attackers?

Quantum computers

When?

No scaling quantum computers

Quantum computers

When?

No scaling quantum computers yet...

Quantum computers

When?

No scaling quantum computers yet...

The issue

Quantum computers allow for a significant speed up for solving many problems

$\Rightarrow$ breaks RSA, computes discrete logarithms...

Quantum computers

When?

No scaling quantum computers yet. . .

The issue

Quantum computers allow for a significant speed up for solving many problems

$\Rightarrow$ breaks RSA, computes discrete logarithms. . .

$\hookrightarrow$ We need new primitives, new protocols and new proofs.

Attacker models

	Symbolic Tools (Proverif,Tamarin,Deepsec, . . .)	Computational Tools (EasyCrypt, CryptoVerif, . . .)
Attacker	Fixed set of possible computations on abstract messages	Turing Machines on bitstrings

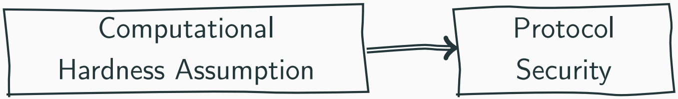
Attacker models

	Symbolic Tools (Proverif,Tamarin,Deepsec, . . .)	Computational Tools (EasyCrypt, CryptoVerif, . . .)
Attacker	Fixed set of possible computations on abstract messages	Turing Machines on bitstrings
Post-quantum?	Abstract reasoning still valid	

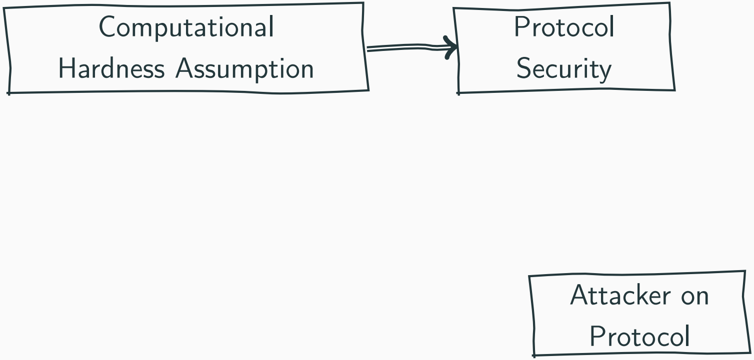
Attacker models

	Symbolic Tools (Proverif,Tamarin,Deepsec, . . .)	Computational Tools (EasyCrypt, CryptoVerif, . . .)
Attacker	Fixed set of possible computations on abstract messages	Turing Machines on bitstrings
Post-quantum?	Abstract reasoning still valid	Quantum Turing Machines

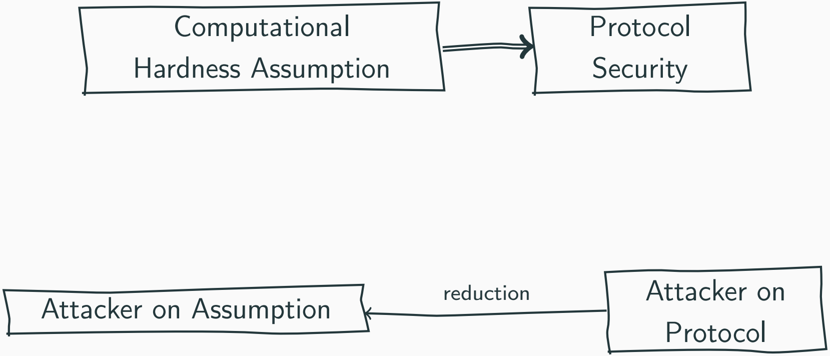
A first look at classical computational proofs



A first look at classical computational proofs



A first look at classical computational proofs



Classical Proofs

Two ingredients

- An assumption

  (a computational assumption that holds for any attacker, e.g. RSA is unbreakable)

Classical Proofs

Two ingredients

- An assumption

  (a computational assumption that holds for any attacker, e.g. RSA is unbreakable)

- A reduction

  (the construction of a new attacker using the one against the assumption,

  similar to NP-hardness proofs or undecidability proofs)

Classical Proofs

Two ingredients

- An assumption over quantum computers

  (a computational assumption that holds for any attacker, e.g. RSA is unbreakable)

- A reduction over quantum computers

  (the construction of a new attacker using the one against the assumption,

  similar to NP-hardness proofs or undecidability proofs)

Classical Proofs

Two ingredients

- An assumption over quantum computers

  (a computational assumption that holds for any attacker, e.g. RSA is unbreakable)

- A reduction over quantum computers

  (the construction of a new attacker using the one against the assumption,

  similar to NP-hardness proofs or undecidability proofs)

A tale of two issues

- No drop in quantum replacement for some classical assumptions (DDH).

Classical Proofs

Two ingredients

- An assumption over quantum computers

  (a computational assumption that holds for any attacker, e.g. RSA is unbreakable)

- A reduction over quantum computers
  (the construction of a new attacker using the one against the assumption,

  similar to NP-hardness proofs or undecidability proofs)

A tale of two issues

- No drop in quantum replacement for some classical assumptions (DDH).
- There are ways to manipulate a classical attacker that cannot be done with a quantum one.

What is a classical attacker?

Probabilistic attacker model

A deterministic computer $\mathcal{A}$ with a random string ρ and inputs $\vec{i}$

$$\mathcal{A}(\rho, \vec{i})$$

What is a classical attacker?

Probabilistic attacker model

A deterministic computer $\mathcal{A}$ with a random string ρ and inputs $\vec{i}$

$$\mathcal{A}(\rho, \vec{i})$$

Allows to simulate weird executions

Run twice the attacker with the same source of randomness on two distinct inputs:

$\mathcal{A}(\rho, \vec{i}_1)$ and $\mathcal{A}(\rho, \vec{i}_2)$

What is a classical attacker?

Probabilistic attacker model

A deterministic computer $\mathcal{A}$ with a random string ρ and inputs $\vec{i}$

$$\mathcal{A}(\rho, \vec{i})$$

Allows to simulate weird executions

Run twice the attacker with the same source of randomness on two distinct inputs:

$\mathcal{A}(\rho, \vec{i_1})$ and $\mathcal{A}(\rho, \vec{i_2})$

Impossible computation with a quantum computer

What is a quantum attacker?

It is impossible to

What is a quantum attacker?

It is impossible to

- Run twice a quantum computer with fixed randomness

What is a quantum attacker?

It is impossible to

- Run twice a quantum computer with fixed randomness
- Duplicate a quantum state (no-cloning theorem)

What is a quantum attacker?

It is impossible to

- Run twice a quantum computer with fixed randomness
- Duplicate a quantum state (no-cloning theorem)

$\hookrightarrow$ Reductions must not use techniques relying on this (e.g., rewinding)

Quantum-sound reductions

Many pitfalls

Must be careful about

- manipulations of the attacker's state;
- mentions of the attacker's randomness;

Quantum-sound reductions

Many pitfalls

Must be careful about

- manipulations of the attacker's state;
- mentions of the attacker's randomness;
- arguments about numbers of queries made to an oracle (QROM);
- arguments about complexity classes.

Quantum-sound reductions

Many pitfalls

Must be careful about

- manipulations of the attacker's state;
- mentions of the attacker's randomness;
- arguments about numbers of queries made to an oracle (QROM);
- arguments about complexity classes.

$\hookrightarrow$ The computational tools do this kind of things...

Contributions

Our contributions[1]

[1] Joint work with Cas Cremers, Caroline Fontaine, and discussions with Hubert Comon.

Contributions

Our contributions[1]

- Take the BC logic - a logic for deriving computational security guarantees

[1] Joint work with Cas Cremers, Caroline Fontaine, and discussions with Hubert Comon.

Contributions

Our contributions[1]

- Take the BC logic - a logic for deriving computational security guarantees
- Make it sound for quantum attackers

[1] Joint work with Cas Cremers, Caroline Fontaine, and discussions with Hubert Comon.

Contributions

Our contributions[1]

- Take the BC logic - a logic for deriving computational security guarantees
- Make it sound for quantum attackers
- Take the Squirrel Prover - an interactive prover for the BC logic

[1] Joint work with Cas Cremers, Caroline Fontaine, and discussions with Hubert Comon.

Contributions

Our contributions[1]

- Take the BC logic - a logic for deriving computational security guarantees
- Make it sound for quantum attackers
- Take the Squirrel Prover - an interactive prover for the BC logic
- Extend it to support the adapted PQ sound logic

[1] Joint work with Cas Cremers, Caroline Fontaine, and discussions with Hubert Comon.

Contributions

Our contributions[1]

- Take the BC logic - a logic for deriving computational security guarantees
- Make it sound for quantum attackers
- Take the Squirrel Prover - an interactive prover for the BC logic
- Extend it to support the adapted PQ sound logic
- Use it on some new protocols - KEM based post-quantum key exchanges

[1] Joint work with Cas Cremers, Caroline Fontaine, and discussions with Hubert Comon.

Related work

Some related work

- John Watrous. Zero-knowledge against quantum attacks.
  - $\hookrightarrow$ Identified the no cloning theorem as an issue.

Related work

Some related work

- John Watrous. Zero-knowledge against quantum attacks.
  $\hookrightarrow$ Identified the no cloning theorem as an issue.
- Fang Song. A note on quantum security for post-quantum cryptography.
- Tommaso Gagliardoni. Quantum Security of Cryptographic Primitives.
  $\hookrightarrow$ Identified classes of valid reductions for pen and paper proofs.

Related work

Some related work

- John Watrous. Zero-knowledge against quantum attacks.
  $\hookrightarrow$ Identified the no cloning theorem as an issue.
- Fang Song. A note on quantum security for post-quantum cryptography.
- Tommaso Gagliardoni. Quantum Security of Cryptographic Primitives.
  $\hookrightarrow$ Identified classes of valid reductions for pen and paper proofs.
- EasyPQC - [BBFGHKSWZ - CCS'21] (parallel work)
  $\hookrightarrow$ Post-quantum sound EasyCrypt - hard to scale to protocols

A post-quantum BC logic

The BC logic

The BC logic[2]

A first-order logic to prove the security of protocols.

[2] [Bana,Comon-CCS'14]

The BC logic[2]

A first-order logic to prove the security of protocols.

$\hookrightarrow$ a proof implies the existence of a reduction.

[2] [Bana,Comon-CCS'14]

The BC logic

The BC logic[2]

A first-order logic to prove the security of protocols.

$\hookrightarrow$ a proof implies the existence of a reduction.

A computationally sound logic

Three main ingredients:

- terms, and their interpretation so that terms can syntactically describe all behaviours of a protocol;

  $\hookrightarrow$ if there exists an attack on the protocol, we can see it on the terms.

[2] [Bana,Comon-CCS'14]

The BC logic

The BC logic[2]

A first-order logic to prove the security of protocols.

$\hookrightarrow$ a proof implies the existence of a reduction.

A computationally sound logic

Three main ingredients:

- terms, and their interpretation so that terms can syntactically describe all behaviours of a protocol;

  $\hookrightarrow$ if there exists an attack on the protocol, we can see it on the terms.
- logical predicates and rules (with axioms about e.g. RSA) to reason over the terms;

[2] [Bana,Comon-CCS'14]

The BC logic

The BC logic[2]

A first-order logic to prove the security of protocols.

$\hookrightarrow$ a proof implies the existence of a reduction.

A computationally sound logic

Three main ingredients:

- terms, and their interpretation so that terms can syntactically describe all behaviours of a protocol;

  $\hookrightarrow$ if there exists an attack on the protocol, we can see it on the terms.
- logical predicates and rules (with axioms about e.g. RSA) to reason over the terms;
- prove the soundness of the rules, i.e., they correspond to valid reduction.

  $\hookrightarrow$ if there is an attack on the protocol, there is an attack against the axioms.

[2] [Bana,Comon-CCS'14]

Going post-quantum

Make it post-quantum sound

- New primitives;

  $\hookrightarrow$ design new axioms and rules.

Make it post-quantum sound

- New primitives;
  - $\hookrightarrow$ design new axioms and rules.
- Verify the post-quantum soundness of the rules;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way?

Going post-quantum

Make it post-quantum sound

- New primitives;
  - $\hookrightarrow$ design new axioms and rules.
- Verify the post-quantum soundness of the rules;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way?
- Verify the term interpretation;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way?

Going post-quantum

Make it post-quantum sound

- New primitives;
  - $\hookrightarrow$ design new axioms and rules. straightforward
- Verify the post-quantum soundness of the rules;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way?
- Verify the term interpretation;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way?

Going post-quantum

Make it post-quantum sound

- New primitives;
  - $\hookrightarrow$ design new axioms and rules. straightforward
- Verify the post-quantum soundness of the rules;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way? No
- Verify the term interpretation;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way?

Make it post-quantum sound

- New primitives;
  - $\hookrightarrow$ design new axioms and rules. straightforward
- Verify the post-quantum soundness of the rules;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way? No
- Verify the term interpretation;
  - $\hookrightarrow$ do we manipulate the attacker in a bad way? Yes

The BC logic[3]

[3] [Bana,Comon-CCS'14]

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

[3] [Bana,Comon-CCS'14]

The BC logic

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs **BC terms**

[3] [Bana,Comon-CCS'14]

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	

[3] [Bana,Comon-CCS'14]

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk

[3] [Bana,Comon-CCS'14]

The BC logic

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	BC terms
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk
$m \xleftarrow{\$} \mathcal{A}(1^\eta)$	

3 [Bana,Comon-CCS'14]

The BC logic

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk
$m \xleftarrow{\$} \mathcal{A}(1^\eta)$	$\mathsf{att}_0()$

[3] [Bana,Comon-CCS'14]

The BC logic

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk
$m \xleftarrow{\$} \mathcal{A}(1^\eta)$	$\mathsf{att}_0()$
$t \xleftarrow{\$} \mathtt{enc}(m, sk)$	

[3] [Bana,Comon-CCS'14]

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk
$m \xleftarrow{\$} \mathcal{A}(1^\eta)$	$\mathsf{att}_0()$
$t \xleftarrow{\$} \mathtt{enc}(m, sk)$	$\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})$

[3] [Bana,Comon-CCS'14]

The BC logic

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk
$m \xleftarrow{\$} \mathcal{A}(1^\eta)$	$\mathsf{att}_0()$
$t \xleftarrow{\$} \mathtt{enc}(m, sk)$	$\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})$
$x \xleftarrow{\$} \mathcal{A}(t)$	

[3] [Bana,Comon-CCS'14]

The BC logic

The BC logic[3]

Protocols are now expressed only with terms, i.e., purely syntactic construct, where everything becomes pure functional calls.

Classical proofs	**BC terms**
$sk \xleftarrow{\$} \{0,1\}^\eta$	sk
$m \xleftarrow{\$} \mathcal{A}(1^\eta)$	$\mathsf{att}_0()$
$t \xleftarrow{\$} \mathtt{enc}(m, sk)$	$\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})$
$x \xleftarrow{\$} \mathcal{A}(t)$	$\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk}))$

[3] [Bana,Comon-CCS'14]

From protocols to terms

A protocol

new sk;
in(x);
if $x = \text{sk}$ then
  out(ko)
else
  out(ok)

From protocols to terms

A protocol

$$\begin{aligned}&\text{new sk};\\&\text{in}(x);\\&\text{if } x = \text{sk then}\\&\quad \text{out}(ko)\\&\ \text{else}\\&\quad \text{out}(ok)\end{aligned}$$

Becomes a term

$$\text{if } (\text{att}_0() = \text{sk}) \text{ then } ko \text{ else } ok$$

Some rules

Some rules

Refl

$$\frac{}{u \sim u}$$

Some rules

Refl

$$\frac{}{u \sim u}$$

=ind

$$\frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}$$ when n does not occur in t

Some rules

Refl

$$\frac{}{u \sim u}$$

=ind

$$\frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}$$ when n does not occur in t

If-f

$$\frac{\phi \sim \mathsf{false} \qquad u \sim w}{\mathsf{if}\ \phi\ \mathsf{then}\ u\ \mathsf{else}\ v \sim w}$$

Some rules

Refl
$$\frac{}{u \sim u}$$

=ind
$$\frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}$$ when n does not occur in t

If-f
$$\frac{\phi \sim \mathsf{false} \quad u \sim w}{\mathsf{if}\ \phi\ \mathsf{then}\ u\ \mathsf{else}\ v \sim w}$$

$$\mathsf{if}\ \mathsf{att}_0() = \mathsf{sk}\ \mathsf{then}\ \mathit{ko}\ \mathsf{else}\ \mathit{ok} \sim \mathit{ok}$$

Some rules

Refl
$$\frac{}{u \sim u}$$

=ind
$$\frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}$$ when n does not occur in t

If-f
$$\frac{\phi \sim \mathsf{false} \quad u \sim w}{\text{if } \phi \text{ then } u \text{ else } v \sim w}$$

If-f
$$\frac{}{\text{if att}_0() = \text{sk then } ko \text{ else } ok \sim ok}$$

Some rules

Refl
$$\frac{}{u \sim u}$$

=ind
$$\frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}$$ when n does not occur in t

If-f
$$\frac{\phi \sim \mathsf{false} \quad u \sim w}{\mathsf{if}\ \phi\ \mathsf{then}\ u\ \mathsf{else}\ v \sim w}$$

If-f
$$\frac{(\mathsf{att}_0() = \mathsf{sk}) \sim \mathsf{false}}{\mathsf{if}\ \mathsf{att}_0() = \mathsf{sk}\ \mathsf{then}\ \mathit{ko}\ \mathsf{else}\ \mathit{ok} \sim \mathit{ok}}$$

Some rules

Refl

$$\frac{}{u \sim u}$$

=ind

$$\frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}} \text{ when n does not occur in t}$$

If-f

$$\frac{\phi \sim \mathsf{false} \quad u \sim w}{\mathsf{if}\ \phi\ \mathsf{then}\ u\ \mathsf{else}\ v \sim w}$$

If-f

$$\frac{(\mathsf{att}_0() = \mathsf{sk}) \sim \mathsf{false} \qquad ok \sim ok}{\mathsf{if}\ \mathsf{att}_0() = \mathsf{sk}\ \mathsf{then}\ ko\ \mathsf{else}\ ok \sim ok}$$

Some rules

$$\textsf{Refl}\ \frac{}{u \sim u} \qquad \textsf{=ind}\ \frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}\ \text{when n does not occur in t} \qquad \textsf{If-f}\ \frac{\phi \sim \mathsf{false} \quad u \sim w}{\text{if } \phi \text{ then } u \text{ else } v \sim w}$$

$$\textsf{If-f}\ \frac{\textsf{=ind}\ \dfrac{}{(\mathsf{att}_0() = \mathsf{sk}) \sim \mathsf{false}} \qquad ok \sim ok}{\text{if } \mathsf{att}_0() = \mathsf{sk} \text{ then } ko \text{ else } ok \sim ok}$$

Some rules

$$\textsf{Refl}\ \frac{}{u \sim u} \qquad \textsf{=ind}\ \frac{}{(t \doteq \mathsf{n}) \sim \mathsf{false}}\ \text{when n does not occur in t} \qquad \textsf{If-f}\ \frac{\phi \sim \mathsf{false} \quad u \sim w}{\mathsf{if}\ \phi\ \mathsf{then}\ u\ \mathsf{else}\ v \sim w}$$

$$\textsf{If-f}\ \frac{\textsf{=ind}\ \frac{}{(\mathsf{att}_0() = \mathsf{sk}) \sim \mathsf{false}} \qquad \textsf{Refl}\ \frac{}{ok \sim ok}}{\mathsf{if}\ \mathsf{att}_0() = \mathsf{sk}\ \mathsf{then}\ ko\ \mathsf{else}\ ok \sim ok}$$

But wait. . .

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

But wait. . .

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathsf{att}_0()$$

But wait. . .

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})$$

But wait...

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk}))$$

But wait...

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathtt{enc}(\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})), \mathsf{r}', \mathsf{sk})$$

But wait. . .

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathtt{enc}(\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})), \mathsf{r}', \mathsf{sk})$$

The logic quantifies over all sets of potential values of att_1 and att_0

$\hookrightarrow$ all possible Turing machines $\mathcal{T}_{\mathsf{att}_0}$ and $\mathcal{T}_{\mathsf{att}_1}$, and thus all attackers.

But wait. . .

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathtt{enc}(\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})), \mathsf{r}', \mathsf{sk})$$

The logic quantifies over all sets of potential values of att_1 and att_0

$\hookrightarrow$ all possible Turing machines $\mathcal{T}_{\mathsf{att}_0}$ and $\mathcal{T}_{\mathsf{att}_1}$, and thus all attackers.

But. . .

- In the real world, we have a stateful interactive probabilistic attacker $\mathcal{A}$.

But wait...

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathtt{enc}(\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})), \mathsf{r}', \mathsf{sk})$$

The logic quantifies over all sets of potential values of att_1 and att_0

$\hookrightarrow$ all possible Turing machines $\mathcal{T}_{\mathsf{att}_0}$ and $\mathcal{T}_{\mathsf{att}_1}$, and thus all attackers.

But...

- In the real world, we have a stateful interactive probabilistic attacker $\mathcal{A}$.
- In the BC world, we have two stateless (because a call to att_i must be pure) and independent deterministic attackers $\mathcal{T}_{\mathsf{att}_i}$ that share a source of randomness.

But wait...

Does this allows to capture real life behaviours?

A protocol where we encrypt two consecutive attacker chosen values:

$$\mathtt{enc}(\mathsf{att}_1(\mathtt{enc}(\mathsf{att}_0(), \mathsf{r}, \mathsf{sk})), \mathsf{r}', \mathsf{sk})$$

The logic quantifies over all sets of potential values of att_1 and att_0

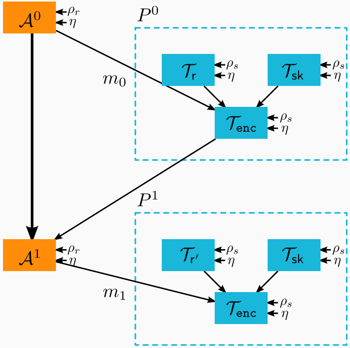
$\hookrightarrow$ all possible Turing machines $\mathcal{T}_{\mathsf{att}_0}$ and $\mathcal{T}_{\mathsf{att}_1}$, and thus all attackers.

But...

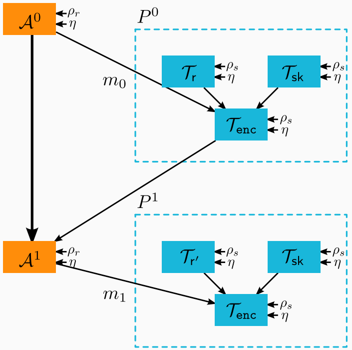
- In the real world, we have a stateful interactive probabilistic attacker $\mathcal{A}$.
- In the BC world, we have two stateless (because a call to att_i must be pure) and independent deterministic attackers $\mathcal{T}_{\mathsf{att}_i}$ that share a source of randomness.

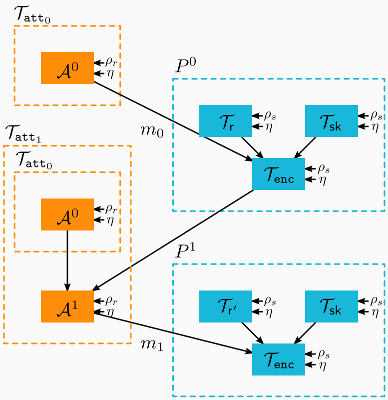
$\hookrightarrow$ Solved by specifying that $\mathcal{T}_{\mathsf{att}_1}$ always starts by recomputing the state of $\mathcal{T}_{\mathsf{att}_0}$

Real world interaction vs BC interaction

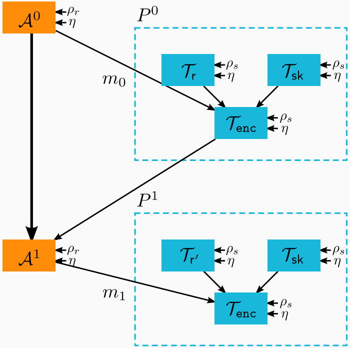


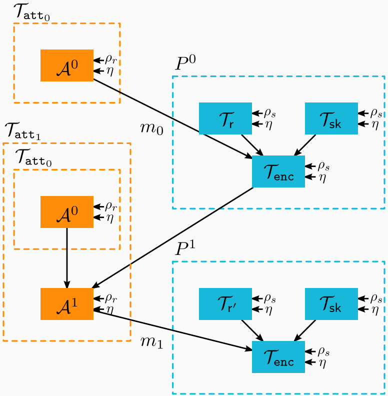
Real world interaction vs BC interaction





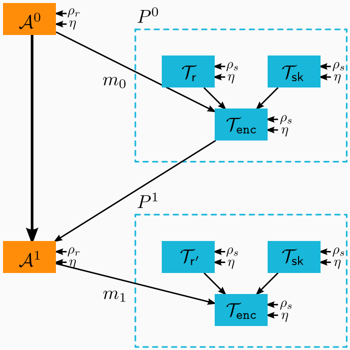
Real world interaction vs BC interaction

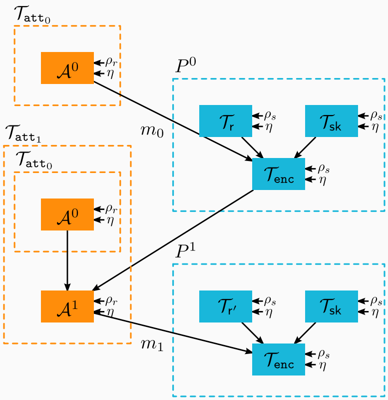




We compute twice $\mathcal{A}_0(\rho_r, \eta)$ to reconstruct its state.

Real world interaction vs BC interaction





We compute twice $\mathcal{A}_0(\rho_r, \eta)$ to reconstruct its state.

Impossible with a quantum attacker

Going post-quantum

First issue

Behind the curtain, the interpretation of terms crucially rely on two facts:

- we can see a probabilistic attacker as some deterministic $\mathcal{A}(1^\eta, \rho_r)$,
- and run it multiple times with the same randomness to reconstruct internal states.

Going post-quantum

First issue

Behind the curtain, the interpretation of terms crucially rely on two facts:

- we can see a probabilistic attacker as some deterministic $\mathcal{A}(1^\eta, \rho_r)$,
- and run it multiple times with the same randomness to reconstruct internal states.

The two impossible operations with a quantum attacker!

Going post-quantum

First issue

Behind the curtain, the interpretation of terms crucially rely on two facts:

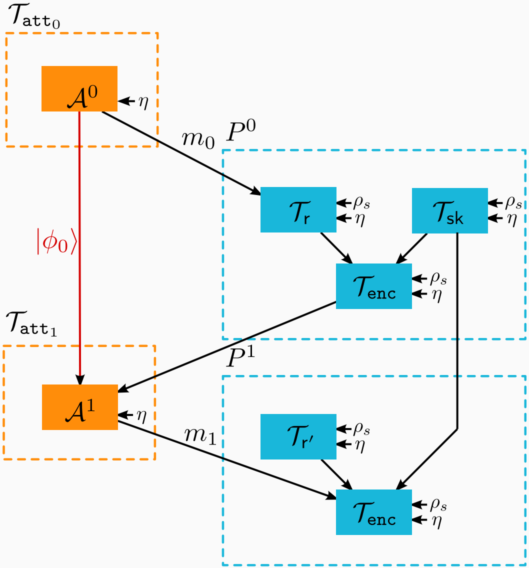
- we can see a probabilistic attacker as some deterministic $\mathcal{A}(1^\eta, \rho_r)$,
- and run it multiple times with the same randomness to reconstruct internal states.

The two impossible operations with a quantum attacker!

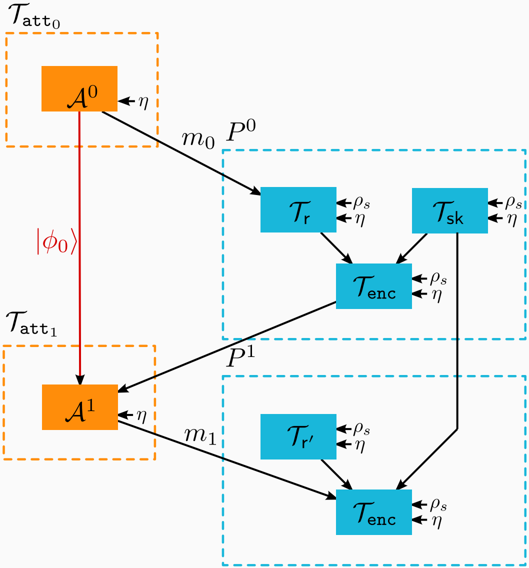
Our main contribution

An interpretation sound for interactive black-box attackers, where the interpretation directly depends a single interactive Turing Machine $\mathcal{T}_{\mathcal{A}}$, instead of many $\mathcal{T}_{\mathsf{att}_i}$.

New (natural) interpretation

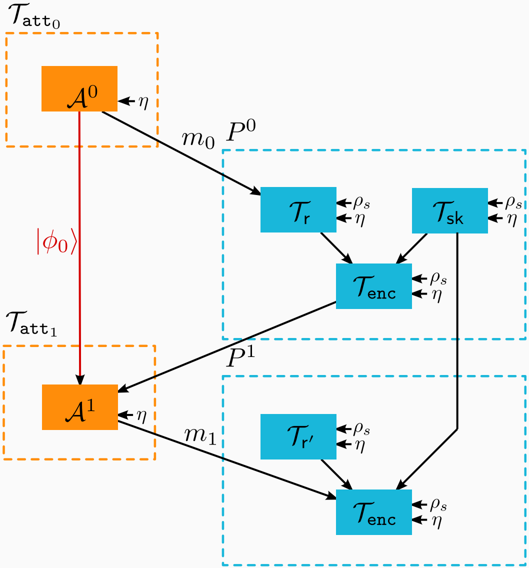


New (natural) interpretation



But the old rules break down. . .

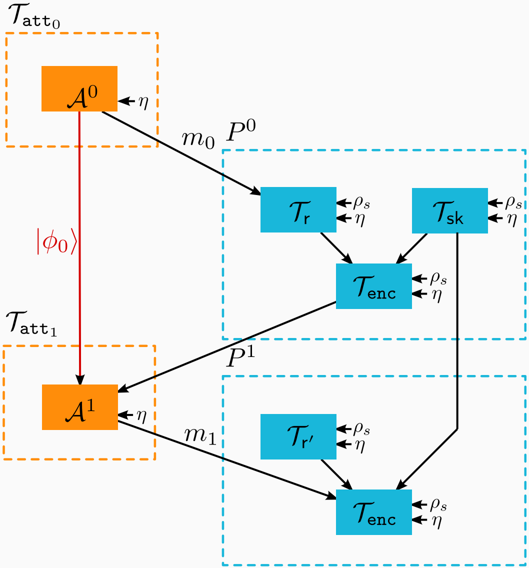
New (natural) interpretation



But the old rules break down. . .

$\mathsf{att}_0(\mathsf{sk})$, if $\mathsf{att}_1() = \mathsf{sk}$ then *ko* else *ok*

New (natural) interpretation



But the old rules break down...

$$\mathsf{att}_0(\mathsf{sk}), \text{if } \mathsf{att}_1() = \mathsf{sk} \text{ then } \mathit{ko} \text{ else } \mathit{ok}$$

$$\sim$$

$$\mathsf{att}_0(\mathsf{sk}), \mathit{ok}$$

If it could have been that simple...

A cascade of changes

If it could have been that simple...

A cascade of changes

- What is the meaning of the sequence $(\mathrm{att}_1(ok), \mathrm{att}_1(ko))$?

If it could have been that simple...

A cascade of changes

- What is the meaning of the sequence $(\mathsf{att}_1(ok), \mathsf{att}_1(ko))$?

  $\hookrightarrow$ we must forbid such things, that model a rewinding

If it could have been that simple...

A cascade of changes

- What is the meaning of the sequence $(\mathrm{att}_1(ok), \mathrm{att}_1(ko))$?

  $\hookrightarrow$ we must forbid such things, that model a rewinding

- What is the meaning of the sequence $(\mathrm{att}_0(ok), \mathrm{att}_1())$?

If it could have been that simple...

A cascade of changes

- What is the meaning of the sequence $(\mathrm{att}_1(ok), \mathrm{att}_1(ko))$?

  $\hookrightarrow$ we must forbid such things, that model a rewinding

- What is the meaning of the sequence $(\mathrm{att}_0(ok), \mathrm{att}_1())$?

  $\hookrightarrow$ att_1 should depend on *ok*, as the machine that will interpret it will have seen it in the first step.

If it could have been that simple...

A cascade of changes

- What is the meaning of the sequence $(\mathsf{att}_1(ok), \mathsf{att}_1(ko))$?

  $\hookrightarrow$ we must forbid such things, that model a rewinding

- What is the meaning of the sequence $(\mathsf{att}_0(ok), \mathsf{att}_1())$?

  $\hookrightarrow$ att_1 should depend on ok, as the machine that will interpret it will have seen it in the first step.

- What is the validity of the formula $(\mathsf{att}_0() \doteq \mathsf{n}) \sim (\mathsf{att}_1(\mathsf{att}_0()) \doteq \mathsf{n})$?

If it could have been that simple...

A cascade of changes

- What is the meaning of the sequence $(\mathsf{att}_1(ok), \mathsf{att}_1(ko))$?

  $\hookrightarrow$ we must forbid such things, that model a rewinding
- What is the meaning of the sequence $(\mathsf{att}_0(ok), \mathsf{att}_1())$?

  $\hookrightarrow$ att_1 should depend on ok, as the machine that will interpret it will have seen it in the first step.
- What is the validity of the formula $(\mathsf{att}_0() \doteq \mathsf{n}) \sim (\mathsf{att}_1(\mathsf{att}_0()) \doteq \mathsf{n})$?

  $\hookrightarrow$ he single interactive attacker will know how many time it was called on both sides!

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

- **Consistency** - all occurrences of a att_i for some i occurs with the same arguments;

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

- **Consistency** - all occurrences of a att_i for some i occurs with the same arguments; $\forall \vec{t}, \mathrm{att}_i(x) \in \vec{t} \land \mathrm{att}_i(y) \in \vec{t} \Rightarrow x = y$

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

- **Consistency** - all occurrences of a att_i for some i occurs with the same arguments;
  $\forall \vec{t}, \mathsf{att}_i(x) \in \vec{t} \wedge \mathsf{att}_i(y) \in \vec{t} \Rightarrow x = y$
- **Monotonicity** - inputs of att_i are a prefix of the inputs of att_j, $j > i$;

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

- **Consistency** - all occurrences of a att_i for some i occurs with the same arguments;
  $\forall \vec{t}, \mathsf{att}_i(x) \in \vec{t} \wedge \mathsf{att}_i(y) \in \vec{t} \Rightarrow x = y$
- **Monotonicity** - inputs of att_i are a prefix of the inputs of att_j, $j > i$;
  $\forall \vec{t}, i < j, \mathsf{att}_i(u_1, \ldots, u_i) \in \vec{t} \wedge \mathsf{att}_j(u'_1, \ldots, u'_j) \in \vec{t} \Rightarrow u_1 = u'_1 \wedge \cdots \wedge u_i = u'_i$

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

- **Consistency** - all occurrences of a att_i for some i occurs with the same arguments;
  $\forall \vec{t}, \mathsf{att}_i(x) \in \vec{t} \wedge \mathsf{att}_i(y) \in \vec{t} \Rightarrow x = y$
- **Monotonicity** - inputs of att_i are a prefix of the inputs of att_j, $j > i$;
  $\forall \vec{t}, i < j, \mathsf{att}_i(u_1, \ldots, u_i) \in \vec{t} \wedge \mathsf{att}_j(u'_1, \ldots, u'_j) \in \vec{t} \Rightarrow u_1 = u'_1 \wedge \cdots \wedge u_i = u'_i$
- **Balance** - Same number of calls to the attacker on both sides of every $\vec{u} \sim \vec{v}$.

Our solution

Syntactic conditions

A set of three simple syntactic conditions over terms and formulas.

- **Consistency** - all occurrences of a att_i for some i occurs with the same arguments; $\forall \vec{t}, \mathsf{att}_i(x) \in \vec{t} \wedge \mathsf{att}_i(y) \in \vec{t} \Rightarrow x = y$
- **Monotonicity** - inputs of att_i are a prefix of the inputs of att_j, $j > i$; $\forall \vec{t}, i < j, \mathsf{att}_i(u_1, \ldots, u_i) \in \vec{t} \wedge \mathsf{att}_j(u'_1, \ldots, u'_j) \in \vec{t} \Rightarrow u_1 = u'_1 \wedge \cdots \wedge u_i = u'_i$
- **Balance** - Same number of calls to the attacker on both sides of every $\vec{u} \sim \vec{v}$. $\forall i, \mathsf{att}_i \in \vec{u} \Leftrightarrow \mathsf{att}_i \in \vec{v}$

Our solution

The conditions

Those specific three conditions were chosen because they are:

Our solution

The conditions

Those specific three conditions were chosen because they are:

- Necessary, otherwise one can write terms that don't have any interpretation in the quantum world;

Our solution

The conditions

Those specific three conditions were chosen because they are:

- Necessary, otherwise one can write terms that don't have any interpretation in the quantum world;
- Sufficient to obtain the soundness of the BC logic;

Our solution

The conditions

Those specific three conditions were chosen because they are:

- Necessary, otherwise one can write terms that don't have any interpretation in the quantum world;
- Sufficient to obtain the soundness of the BC logic;
- Simple and syntactic, so we were able to integrate them inside Squirrel with a few hundred lines of code, only at the cost of a small expressivity loss.

What is Squirrel

In a nut: an interactive prover for the BC logic

- Relies on a meta-logic to allow for mechanized proofs of protocol for an unbounded number of sessions;
- gives computational guarantees;
- appears to be usable, and slowly starting to scale to more and more complex protocols.

What is Squirrel

In a nut: an interactive prover for the BC logic

- Relies on a meta-logic to allow for mechanized proofs of protocol for an unbounded number of sessions;
- gives computational guarantees;
- appears to be usable, and slowly starting to scale to more and more complex protocols.

Some figures

- 5 people core team: David Baelde, Stéphanie Delaune, Charlie J., Adrien Koutsos, Solène Moreau (and expanding)
- 30 000 lines of code and celebrating our 2 000 commit!
- about 15 real life case studies of protocols

Implementation and Case-studies

Protocol	LoC	Assumptions	Security properties
Key exchange protocols			
$\text{IkeV1}_{\text{psk}}$	680	PRF, EUF-CMA	Strong Secrecy & Authentication
$\text{IkeV2}_{\text{psk}}^{\text{sign}}$	300	PRF, EUF-CMA	Strong Secrecy & Authentication
KE_{BCGNP}	355	PRF, IND-CCA, XOR	Strong Secrecy & Implicit Authentication
KE_{FSXY}	660	PRF, IND-CCA, XOR	Strong Secrecy & Implicit Authentication
SC-AKE	650	PRF, IND-CCA, SUF-CMA, XOR	Strong Secrecy & Authentication
Proving post-quantum soundness of Squirrelcase studies			
Basic Hash	100	PRF, EUF-CMA	Authentication & Unlinkability
Hash Lock	130	PRF, EUF-CMA	Authentication & Unlinkability
LAK (with pairs)	250	PRF, EUF-CMA	Authentication & Unlinkability
MW	300	PRF, EUF-CMA, XOR	Authentication & Unlinkability
Feldhofer	270	ENC-KP, INT-CTXT	Authentication & Unlinkability
Private Authentication	100	IND-CCA, ENC-KP	Anonymity

What's next?

Recap

Our contribution

The first interactive protocol prover that also provides post-quantum guarantees.

Recap

Our contribution

The first interactive protocol prover that also provides post-quantum guarantees.

Limitation

Our key-exchange case-study do not cover any complex properties or compromise model, and there are no clear framework to prove key-exchanges in Squirrel.

Recap

Our contribution

The first interactive protocol prover that also provides post-quantum guarantees.

Limitation

Our key-exchange case-study do not cover any complex properties or compromise model, and there are no clear framework to prove key-exchanges in Squirrel.

Natural next step

Foundations for proving Key-Exchanges in Squirrel:

- Define how to express complex properties such as PFS or PCS in Squirrel, and simplified with our composition result.
- Link proofs in Squirrel with existing framework (BR, CK,eCK,. . .)
- Perform an extensive case-study (KEMTLS)

The landscape

What we now have (thanks to 40 years of research!)

Many tools, attacker models and associated proof techniques. For instance:

- Proverif and Tamarin to verify at a high-level full protocol specifications;
- Squirrel to verify precisely the core of a protocol.

The landscape

What we now have (thanks to 40 years of research!)

Many tools, attacker models and associated proof techniques. For instance:

- Proverif and Tamarin to verify at a high-level full protocol specifications;
- Squirrel to verify precisely the core of a protocol.

But...

- the tools are developed by distinct groups, and protocol analysis papers are often at a single level using a single tool;
- so many different approaches makes it difficult to export the tools and attacker models outside the protocol community.

The landscape

What we now have (thanks to 40 years of research!)

Many tools, attacker models and associated proof techniques. For instance:

- Proverif and Tamarin to verify at a high-level full protocol specifications;
- Squirrel to verify precisely the core of a protocol.

But. . .

- the tools are developed by distinct groups, and protocol analysis papers are often at a single level using a single tool;
- so many different approaches makes it difficult to export the tools and attacker models outside the protocol community.

Our goal

Build bridges inside the different groups in the community, as well as outside the community.

Medium-term goal

Issue

The existing protocol analysis paper are often at a single level using a single tool.

Medium-term goal

Issue

The existing protocol analysis paper are often at a single level using a single tool.

Goal

Design a platform that allows to formally combine the guarantees of multiple tool:

Medium-term goal

Issue

The existing protocol analysis paper are often at a single level using a single tool.

Goal

Design a platform that allows to formally combine the guarantees of multiple tool:

- First for Tamarin and Proverif, to combine there complementary strengths;

Medium-term goal

Issue

The existing protocol analysis paper are often at a single level using a single tool.

Goal

Design a platform that allows to formally combine the guarantees of multiple tool:

- First for Tamarin and Proverif, to combine there complementary strengths;
- Then integrate Squirrel;

Medium-term goal

Issue

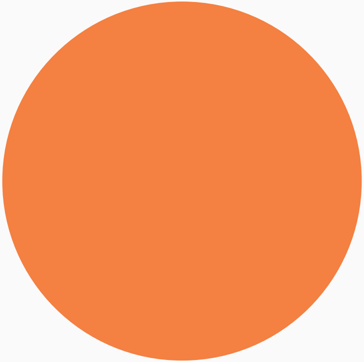
The existing protocol analysis paper are often at a single level using a single tool.

Goal

Design a platform that allows to formally combine the guarantees of multiple tool:

- First for Tamarin and Proverif, to combine there complementary strengths;
- Then integrate Squirrel;
- Make a concrete multi-level analysis.

Long term goal

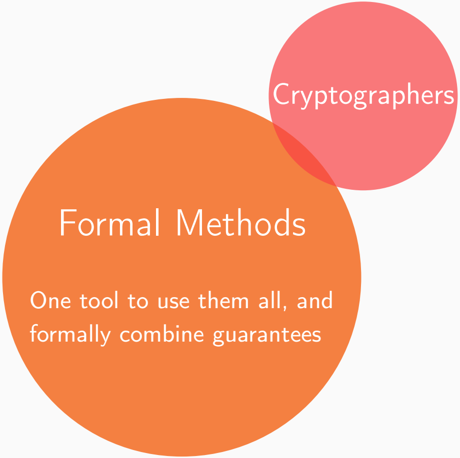


Long term goal

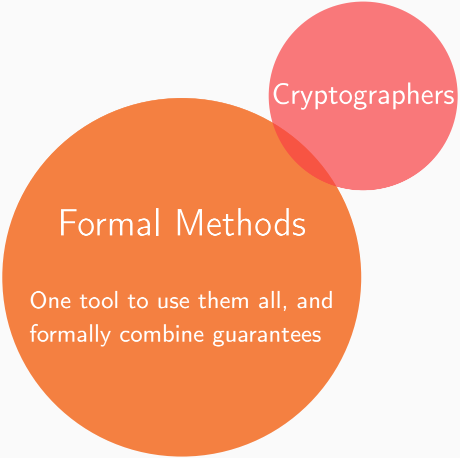
Formal Methods

One tool to use them all, and formally combine guarantees

Long term goal

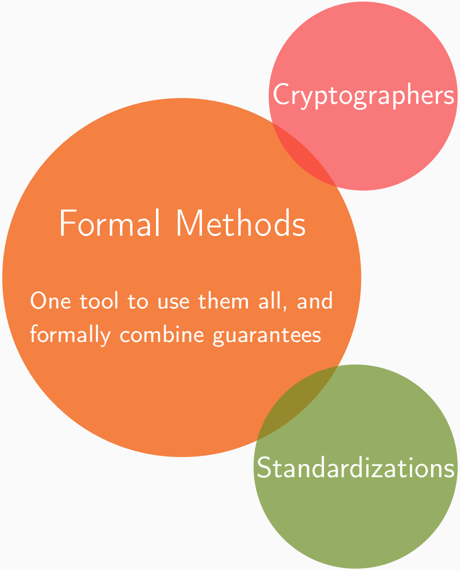


Long term goal



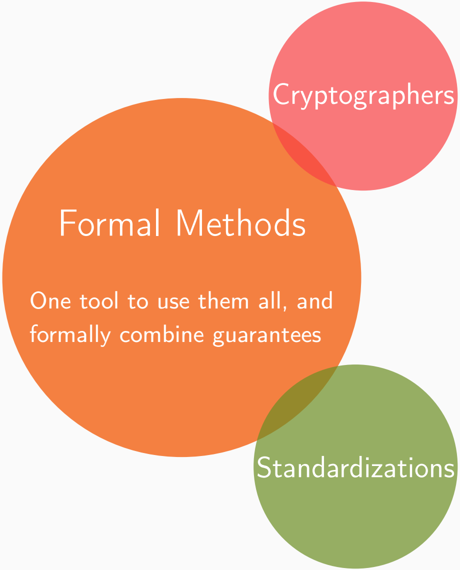
- Use the tools straight away in new protocol designs

Long term goal



- Use the tools straight away in new protocol designs

Long term goal



- Use the tools straight away in new protocol designs

- Provide all standards with formal models
- Participate in the development of new standards

Long term goal

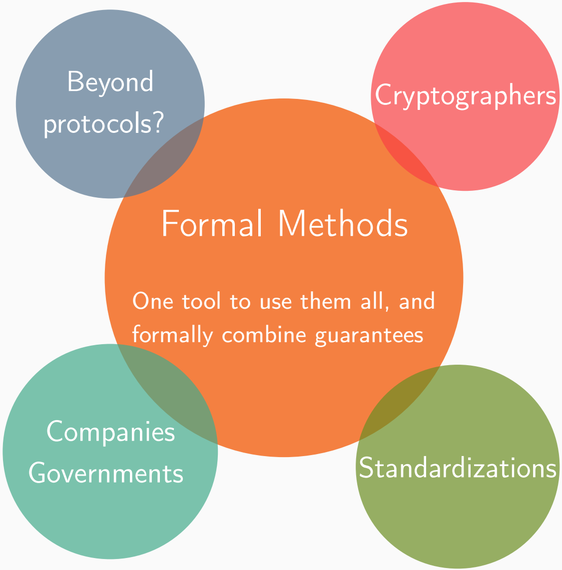
- Use the tools straight away in new protocol designs

- Provide all standards with formal models
- Participate in the development of new standards

Long term goal

- Use the tools straight away in new protocol designs
- Make some techniques available to non-experts
- Provide all standards with formal models
- Participate in the development of new standards

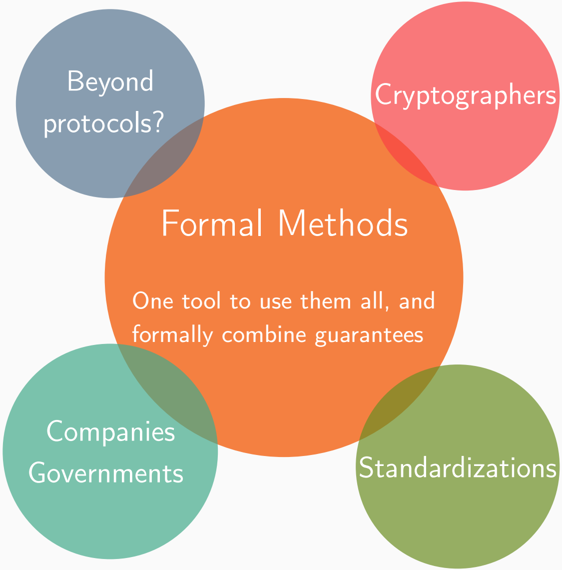
Long term goal



- Use the tools straight away in new protocol designs
- Make some techniques available to non-experts
- Provide all standards with formal models
- Participate in the development of new standards

Long term goal

- attacker models for code level analysis, e.g. for fault-injection
- Make some techniques available to non-experts



- Use the tools straight away in new protocol designs
- Provide all standards with formal models
- Participate in the development of new standards