Symbolic Verification of Complexity-Theoretic Properties of Cryptographic Protocols and Attack Discovery Using First Order Logic

Gergei Bana

University of Luxembourg

with

Hubert Comon-Lundh (ENS Cachan), Mitsuhiro Okada (Keio University)

Symbolic Verification of Complexity-Theoretic Properties of Cryptographic Protocols and Attack Discovery Using First Order Logic

Gergei Bana
University of Luxembourg

with
Hubert Comon-Lundh (ENS Cachan), Mitsuhiro Okada (Keio University)

and
Pedro Adão (Lisbon), Koji Hasebe (Tsukuba), Hideki Sakurada (NTT), Rohit Chadha (University of Missouri), Guillaume Scerri (ENS Cachan, Bristol), Adrien Koutsos (ENS Cachan)

Computationally Sound Cryptographic Protocol Verification Project

Computationally Sound Cryptographic Protocol Verification Project

- Participants:
  - Mitsuhiro Okada (Keio University)
  - Hubert Comon (École Normale Supérieure de Cachan)
  - Gergei Bana (University of Luxembourg)

Computationally Sound Cryptographic Protocol Verification Project

- Participants:
  - Mitsuhiro Okada (Keio University)
  - Hubert Comon (École Normale Supérieure de Cachan)
  - Gergei Bana (University of Luxembourg)
- “Computationally Complete Symbolic Attacker”
  - Project for automated verification/attack finding of complexity theoretic properties (provable security) of security protocols
  - First Order Logic
  - “Unconditionally” computationally sound

What is an Attacker of a Crypto Protocol

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

- Symbolic Attacker (Dolev, Yao 1983)
  - Symbolic operations $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

- Symbolic Attacker (Dolev, Yao 1983)
  - Symbolic operations $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Computational attacker (Micali, Goldwasser, etc. 1980's)
  - Complexity theory
  - PPT algorithms, asymptotic properties

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

- Symbolic Attacker (Dolev, Yao 1983)
  - Symbolic operations $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Computational attacker (Micali, Goldwasser, etc. 1980's)
  - Complexity theory
  - PPT algorithms, asymptotic properties
- Explicit probabilities

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

- Symbolic Attacker (Dolev, Yao 1983)
  - Symbolic operations $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Computational attacker (Micali, Goldwasser, etc. 1980’s)
  - Complexity theory
  - PPT algorithms, asymptotic properties
- Explicit probabilities
- Add side-channel attacks

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

- Symbolic Attacker (Dolev, Yao 1983)
  - Symbolic operations $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Computational attacker (Micali, Goldwasser, etc. 1980's)
  - Complexity theory
  - PPT algorithms, asymptotic properties
- Explicit probabilities
- Add side-channel attacks
- etc.

What is an Attacker of a Crypto Protocol

To **prove** there is no attack, we need to model attacks:

more precise but more complex

- Symbolic Attacker (Dolev, Yao 1983)
  - Symbolic operations $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Computational attacker (Micali, Goldwasser, etc. 1980’s)
  - Complexity theory
  - PPT algorithms, asymptotic properties
- Explicit probabilities
- Add side-channel attacks
- etc.

Computationally Complete Symbolic Attacker

Computationally Complete Symbolic Attacker

- G. Bana - H. Comon: POST'2012

Computationally Complete Symbolic Attacker

- G. Bana - H. Comon: POST'2012
- Modern cryptography: security properties are defined in terms of complexity theory - "**computational model**"
  - Agents, adversary are modeled by probabilistic polynomial-time algorithms
  - Security relies on hardness assumptions (discrete log problem etc)

Computationally Complete Symbolic Attacker

- G. Bana - H. Comon: POST'2012
- Modern cryptography: security properties are defined in terms of complexity theory - "**computational model**"
  - Agents, adversary are modeled by probabilistic polynomial-time algorithms
  - Security relies on hardness assumptions (discrete log problem etc)
- For automated proof we need **symbolic techniques**

Computationally Complete Symbolic Attacker

- G. Bana - H. Comon: POST'2012
- Modern cryptography: security properties are defined in terms of complexity theory - "**computational model**"
  - Agents, adversary are modeled by probabilistic polynomial-time algorithms
  - Security relies on hardness assumptions (discrete log problem etc)
- For automated proof we need **symbolic techniques**
- **Computationally sound** symbolic verification: no symbolic adversary —> No PPT adversary

Computationally Complete Symbolic Attacker

- G. Bana - H. Comon: POST'2012
- Modern cryptography: security properties are defined in terms of complexity theory - "**computational model**"
  - Agents, adversary are modeled by probabilistic polynomial-time algorithms
  - Security relies on hardness assumptions (discrete log problem etc)
- For automated proof we need **symbolic techniques**
- **Computationally sound** symbolic verification: no symbolic adversary —> No PPT adversary
- Computationally complete symbolic attacker: covers all PPT adversaries symbolically

Attempts to Automatically Verify Computational Properties

Attempts to Automatically Verify Computational Properties

- Relating attacker models: “Dolev-Yao Computational Soundness” (Tamarin, ProVerif etc tools,   Backes, Cortier, Warinschi etc)
    - Dolev-Yao symbolic security sometimes means Computational Security
    - Relied on strong restrictions on computational model, not good for algebraic operations

Attempts to Automatically Verify Computational Properties

- Relating attacker models: “Dolev-Yao Computational Soundness” (Tamarin, ProVerif etc tools,   Backes, Cortier, Warinschi etc)
    - Dolev-Yao symbolic security sometimes means Computational Security
    - Relied on strong restrictions on computational model, not good for algebraic operations
- Proofs (partly automated) in the computational model: No symbolic attacker, security property derived directly
    - CryptoVerif (Bruno Blanchet, INRIA)
    - F7 (Fournet et al. Microsoft, INRIA)
    - EasyCrypt (Barthe et al. IMDEA)

Attempts to Automatically Verify Computational Properties

- Relating attacker models: “Dolev-Yao Computational Soundness” (Tamarin, ProVerif etc tools, Backes, Cortier, Warinschi etc)
  - Dolev-Yao symbolic security sometimes means Computational Security
  - Relied on strong restrictions on computational model, not good for algebraic operations
- Proofs (partly automated) in the computational model: No symbolic attacker, security property derived directly
  - CryptoVerif (Bruno Blanchet, INRIA)
  - F7 (Fournet et al. Microsoft, INRIA)
  - EasyCrypt (Barthe et al. IMDEA)
  - Many good results, but for either tools:
    - No attack is found, incomplete proof may mean weak tool, weak user, or insecure protocol
    - Hard to use effectively for others than developers
    - Various hidden assumptions

Our Aim

Our Aim

- Symbolic method convenient for automation

Our Aim

- Symbolic method convenient for automation
- Computational (complexity theoretic) guarantees

Our Aim

- Symbolic method convenient for automation
- Computational (complexity theoretic) guarantees
  - Symbolic verification $\Rightarrow$ No computational attack

Our Aim

- Symbolic method convenient for automation
- Computational (complexity theoretic) guarantees
  - Symbolic verification $\Rightarrow$ No computational attack
- Based on a simple language - Intuitive, easy to use

Our Aim

- Symbolic method convenient for automation
- Computational (complexity theoretic) guarantees
  - Symbolic verification $\Rightarrow$ No computational attack
- Based on a simple language - Intuitive, easy to use
- Transparent - no hidden assumptions

Our Aim

- Symbolic method convenient for automation
- Computational (complexity theoretic) guarantees
  - Symbolic verification $\Rightarrow$ No computational attack
- Based on a simple language - Intuitive, easy to use
- Transparent - no hidden assumptions
- Failure of proof delivers attack

Our Aim

- Symbolic method convenient for automation
- Computational (complexity theoretic) guarantees
  - Symbolic verification $\Rightarrow$ No computational attack
- Based on a simple language - Intuitive, easy to use
- Transparent - no hidden assumptions
- Failure of proof delivers attack
- Algebraic operations, Standard hardness assumptions (DDH, etc) and security notions of primitives (CPA, CCA, etc) are easy to formalize

Our View of Attacker

Our View of Attacker

- Keep the notion of symbolic attacker

Our View of Attacker

- Keep the notion of symbolic attacker
- No explicit symbolic attacker computation
  $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$

Our View of Attacker

- Keep the notion of symbolic attacker

- No explicit symbolic attacker computation
  $K\,,\,x \vdash \{x\}_K$; $K\,,\,\{x\}_K \vdash x$; $x\,,\,y \vdash \langle x,y \rangle$

- Instead: attacker computations represented by function symbols: f_0 f_1 $f_2 \ldots$

Our View of Attacker

- Keep the notion of symbolic attacker
- No explicit symbolic attacker computation
  $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Instead: attacker computations represented by function symbols: f_0 f_1 f_2 …
- No rules about what the attacker **can** do

Our View of Attacker

- Keep the notion of symbolic attacker
- No explicit symbolic attacker computation
  $K\ ,\ x \vdash \{x\}_K\ ;\quad K\ ,\ \{x\}_K \vdash x\ ;\quad x\ ,\ y \vdash \langle x,y \rangle$
- Instead: attacker computations represented by function symbols: f_0 f_1 $f_2 \ldots$
- No rules about what the attacker **can** do
- Instead: (**first-order logic**) **rules** on the messages that **attacker cannot violate**: axiom

Our View of Attacker

- Keep the notion of symbolic attacker
- No explicit symbolic attacker computation
  $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Instead: attacker computations represented by function symbols: f_0 f_1 $f_2 \ldots$
- No rules about what the attacker **can** do
- Instead: (**first-order logic**) **rules** on the messages that **attacker cannot violate**: axiom
- Attackers are allowed to complete everything (f_0 f_1 $f_2 \ldots$ can satisfy anything) except what is forbidden

Our View of Attacker

- Keep the notion of symbolic attacker
- No explicit symbolic attacker computation
  $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Instead: attacker computations represented by function symbols: f_0 f_1 f_2 …
- No rules about what the attacker **can** do
- Instead: (**first-order logic**) **rules** on the messages that **attacker cannot violate**: axiom
- Attackers are allowed to complete everything (f_0 f_1 f_2 … can satisfy anything) except what is forbidden
- Security proofs: security property is derived from the axioms using FOL

Our View of Attacker

- Keep the notion of symbolic attacker
- No explicit symbolic attacker computation
  $K , x \vdash \{x\}_K$; $K , \{x\}_K \vdash x$; $x , y \vdash \langle x,y \rangle$
- Instead: attacker computations represented by function symbols: f_0 f_1 f_2 …
- No rules about what the attacker **can** do
- Instead: (**first-order logic**) **rules** on the messages that **attacker cannot violate**: axiom
- Attackers are allowed to complete everything (f_0 f_1 f_2 … can satisfy anything) except what is forbidden
- Security proofs: security property is derived from the axioms using FOL
- Attack model: Negation of security property consistent with axioms

Proof Method

Proof Method

- First order language on the terms produced by the protocol execution. (Actions in the protocol execution(s) are not part of the first order language)

- First-order logic with a single predicate

  - $t_1,...,t_n \sim u_1,...,u_n$; semantics: computational indistinguishability of PPT algorithms $t_1,...,t_n$ from $u_1,...,u_n$

Proof Method

- First order language on the terms produced by the protocol execution. (Actions in the protocol execution(s) are not part of the first order language)

- First-order logic with a single predicate

    - $t_1,...,t_n \sim u_1,...,u_n$; semantics: computational indistinguishability of PPT algorithms $t_1,...,t_n$ from $u_1,...,u_n$

- Attempt to first-order derive security property of the protocol from:

    - Computationally sound **core Axioms**, independent of primitives

    - Computationally sound **axioms for the primitives or hardness assumptions**: E.g. CCA2, Discrete Logarithm, etc

Proof Method

- First order language on the terms produced by the protocol execution. (Actions in the protocol execution(s) are not part of the first order language)

- First-order logic with a single predicate

  - $t_1,\ldots,t_n \sim u_1,\ldots,u_n$; semantics: computational indistinguishability of PPT algorithms $t_1,\ldots,t_n$ from $u_1,\ldots,u_n$

- Attempt to first-order derive security property of the protocol from:

  - Computationally sound **core Axioms**, independent of primitives

  - Computationally sound **axioms for the primitives or hardness assumptions**: E.g. CCA2, Discrete Logarithm, etc

- If axioms not enough: attack is given by a branch of the proof tree not leading to axiom (concrete model of function symbols)

Proof Method

- First order language on the terms produced by the protocol execution. (Actions in the protocol execution(s) are not part of the first order language)

- First-order logic with a single predicate

  - $t_1,\ldots,t_n \sim u_1,\ldots,u_n$; semantics: computational indistinguishability of PPT algorithms $t_1,\ldots,t_n$ from $u_1,\ldots,u_n$

- Attempt to first-order derive security property of the protocol from:

  - Computationally sound **core Axioms**, independent of primitives

  - Computationally sound **axioms for the primitives or hardness assumptions**: E.g. CCA2, Discrete Logarithm, etc

- If axioms not enough: attack is given by a branch of the proof tree not leading to axiom (concrete model of function symbols)

- Add properties needed for the implementation until you avoid all attacks

Proof Method

- First order language on the terms produced by the protocol execution. (Actions in the protocol execution(s) are not part of the first order language)
- First-order logic with a single predicate
  - $t_1,...,t_n \sim u_1,...,u_n$; semantics: computational indistinguishability of PPT algorithms $t_1,...,t_n$ from $u_1,...,u_n$
- Attempt to first-order derive security property of the protocol from:
  - Computationally sound **core Axioms**, independent of primitives
  - Computationally sound **axioms for the primitives or hardness assumptions**: E.g. CCA2, Discrete Logarithm, etc
- If axioms not enough: attack is given by a branch of the proof tree not leading to axiom (concrete model of function symbols)
- Add properties needed for the implementation until you avoid all attacks
- At the end: A list of properties that if satisfied by the implementation, then secure.

More Than Computational

More Than Computational

- A verification of protocol **P** results a set **S** of

More Than Computational

- A verification of protocol **P** results a set **S** of
  - axioms
  - additional necessary properties

More Than Computational

- A verification of protocol **P** results a set **S** of
  - axioms
  - additional necessary properties
- For any model **M**, not necessarily computational,

More Than Computational

- A verification of protocol **P** results a set **S** of
  - axioms
  - additional necessary properties

- For any model **M**, not necessarily computational,
  - if $\mathbf{M} \models \mathbf{S}$,

More Than Computational

- A verification of protocol **P** results a set **S** of
  - axioms
  - additional necessary properties

- For any model **M**, not necessarily computational,
  - if $\mathbf{M} \models \mathbf{S}$,
  - then the protocol **P** is secure in **M**.

More Than Computational

- A verification of protocol **P** results a set **S** of
  - axioms
  - additional necessary properties
- For any model **M**, not necessarily computational,
  - if **M** $\vDash$ **S**,
  - then the protocol **P** is secure in **M**.
- I.e. implementation should satisfy the properties in S

Core Axioms 1

Axioms for indistinguishability.

REFL: $\vec{x} \sim \vec{x}$

SYM: $\vec{x} \sim \vec{y} \longrightarrow \vec{y} \sim \vec{x}$

TRANS: $\vec{x} \sim \vec{y} \wedge \vec{y} \sim \vec{z} \longrightarrow \vec{x} \sim \vec{z}$

RESTR: If p projects and permutes onto a sublist,
$\vec{x} \sim \vec{y} \longrightarrow p(\vec{x}) \sim p(\vec{y})$

FUNCAPP: for any $\vec{f}$: $\mathsf{message}^n \rightarrow \mathsf{message}^m$, $\vec{f} \in \mathcal{F} \cup \mathcal{G}$,
$\vec{x} \sim \vec{y} \longrightarrow \vec{x}, \vec{f}(\vec{x}) \sim \vec{y}, \vec{f}(\vec{y})$

TFDIST: $\neg\,(\boldsymbol{true} \sim \boldsymbol{false})$

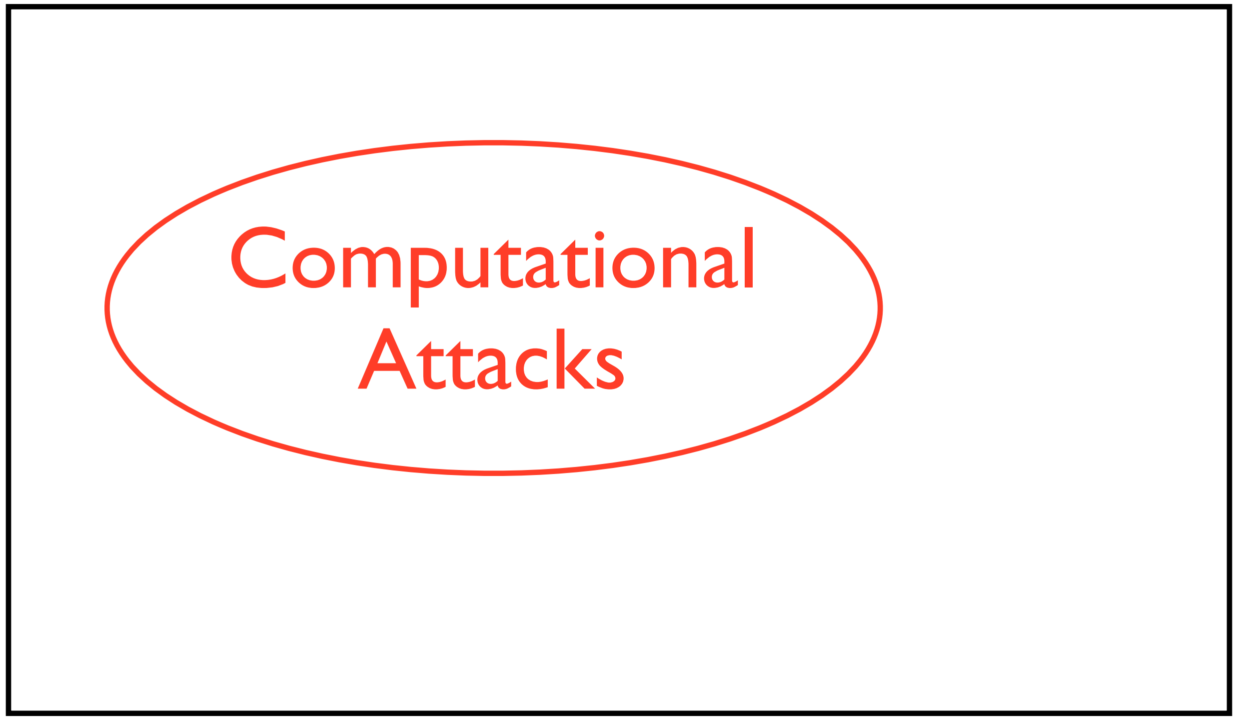
Axioms for equality.

EQREFL: $x = x$

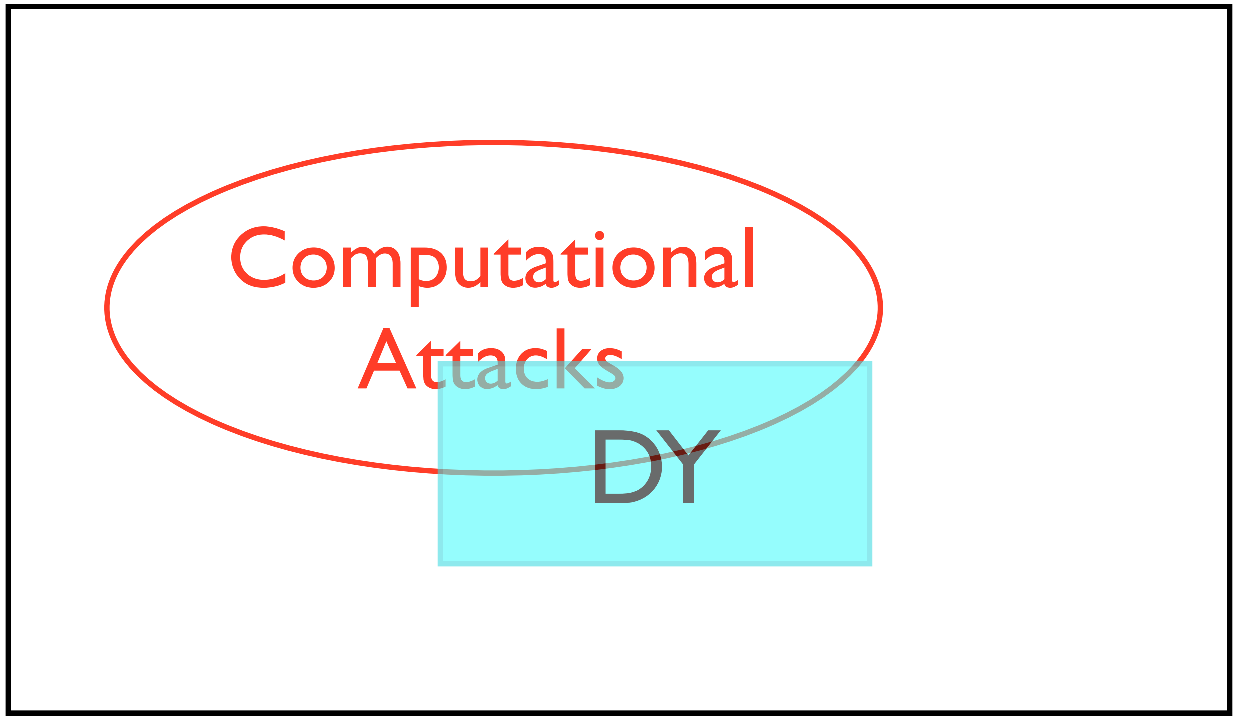
EQCONG: $=$ is a congruence relation with respect to the current syntax.

EQTHEO: $=$ preserves the equational theory of functions

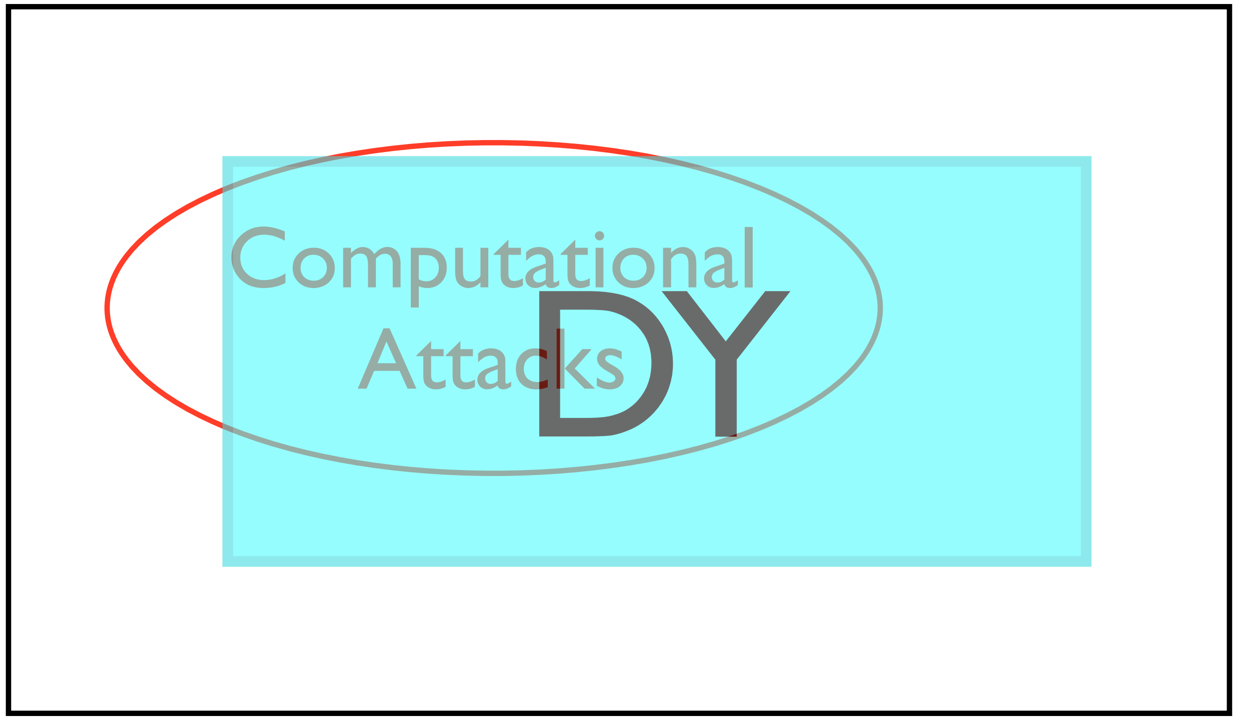
Axioms Limit the Attacker



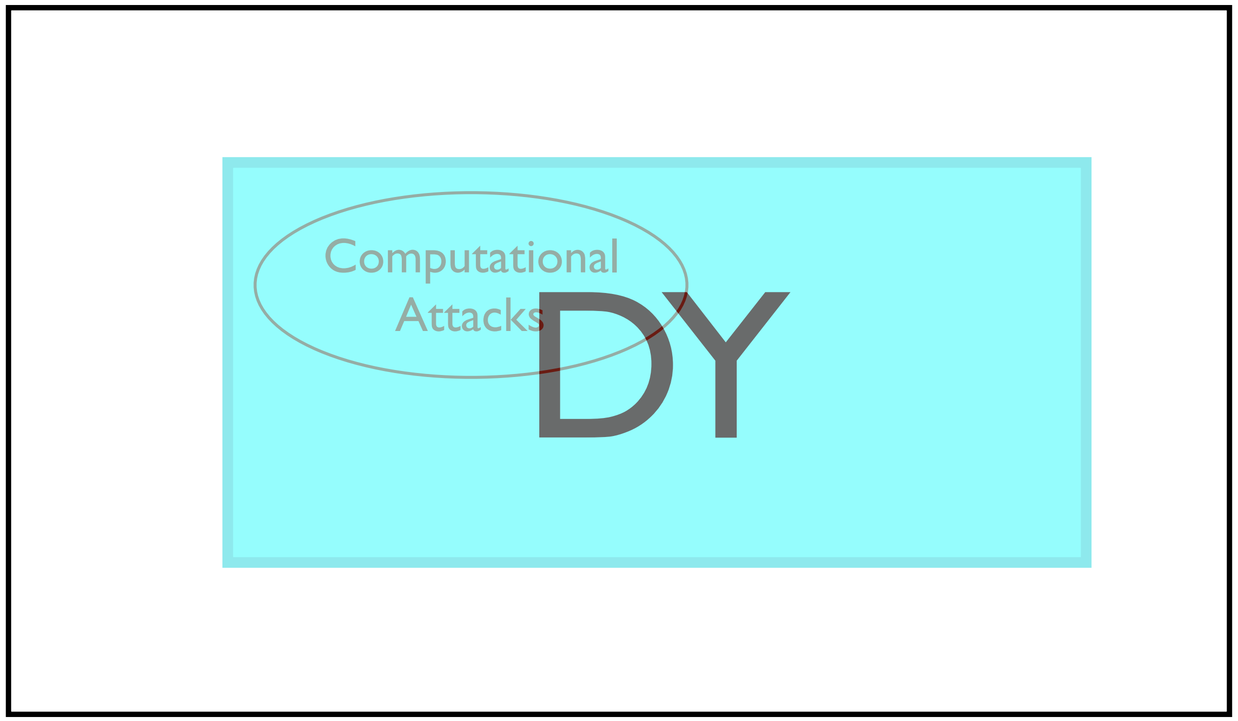
Axioms Limit the Attacker



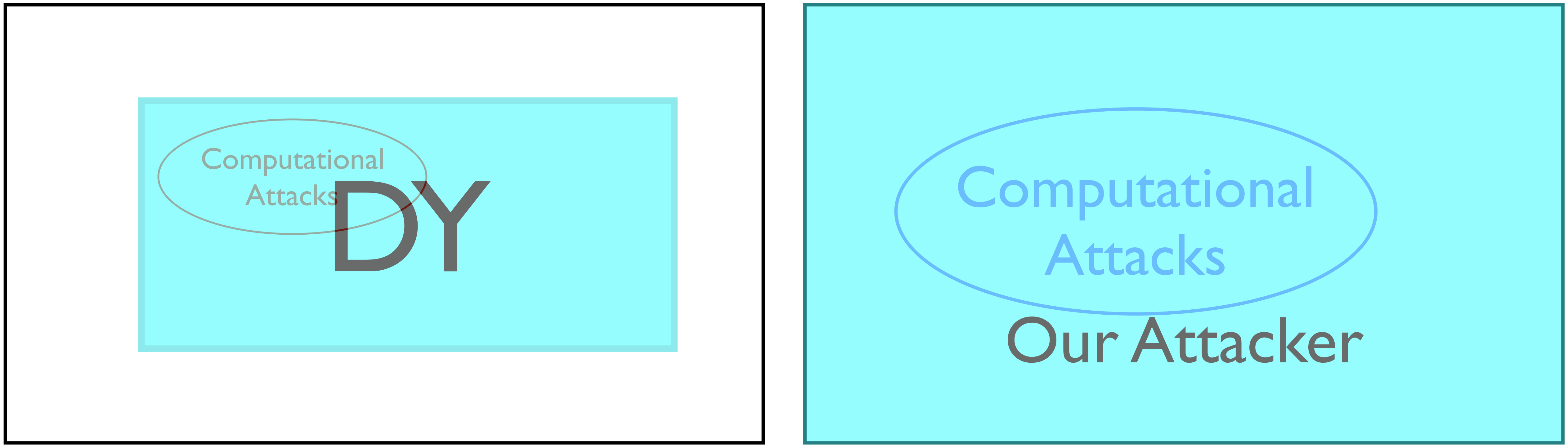
Axioms Limit the Attacker



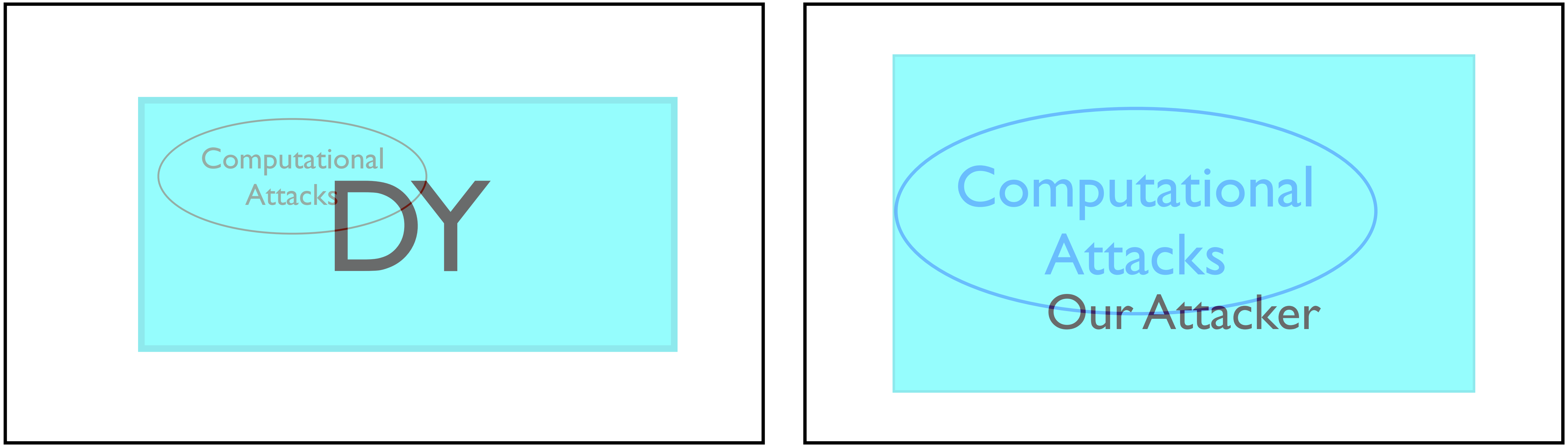
Axioms Limit the Attacker



Axioms Limit the Attacker

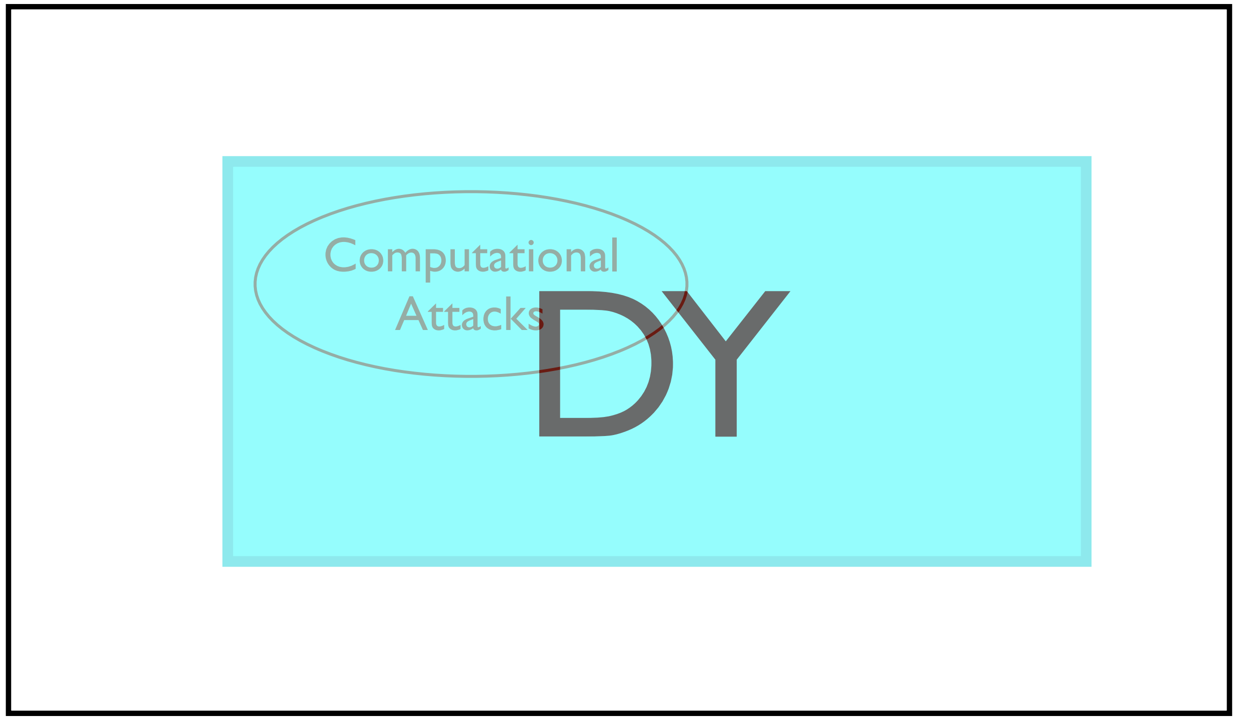


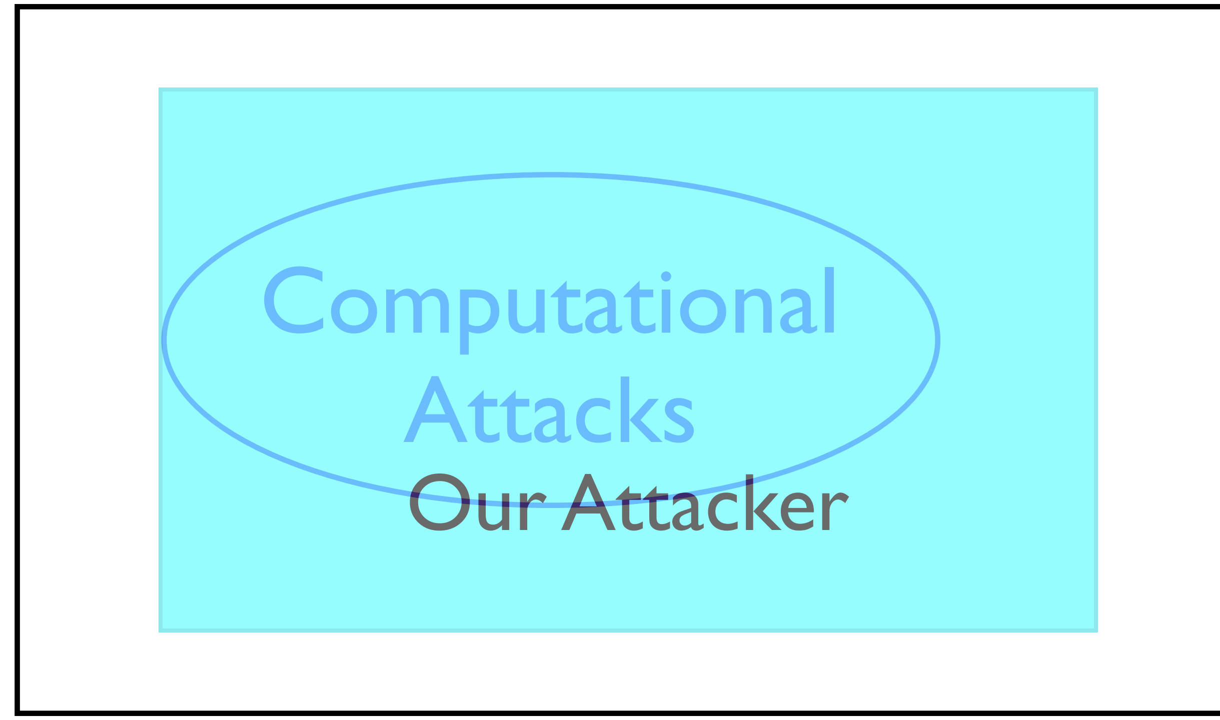
Axioms Limit the Attacker



- Computationally sound axioms

Axioms Limit the Attacker





- Computationally sound axioms
- Idea works for other than computational
- Why computational:
  - Lot of work on computational model
  - We want to compare

Decisional Diffie-Hellman Assumption

$$(\mathsf{G}(n), \mathsf{g}(n), \mathsf{g}(n)^{\mathtt{r}(n_1)}, \mathsf{g}(n)^{\mathtt{r}(n_2)}, \mathsf{g}(n)^{\mathtt{r}(n_1)\mathtt{r}(n_2)})$$
$$\sim$$
$$(\mathsf{G}(n), \mathsf{g}(n), \mathsf{g}(n)^{\mathtt{r}(n_1)}, \mathsf{g}(n)^{\mathtt{r}(n_2)}, \mathsf{g}(n)^{\mathtt{r}(n_3)})$$

Encryptions - CPA, CCA1, CCA2

$$\vec{t}[\text{if } \mathbf{EQ}(\mathtt{L}(u), \mathtt{L}(u')) \text{ then } \{u\}_{\mathsf{pk}(n_1)}^{\mathsf{r}(n_2)} \text{ else } \mathbf{0}\,]$$

$$\sim$$

$$\vec{t}[\text{if } \mathbf{EQ}(\mathtt{L}(u), \mathtt{L}(u')) \text{ then } \{u'\}_{\mathsf{pk}(n_1)}^{\mathsf{r}(n_3)} \text{ else } \mathbf{0}\,]$$

- L is length.
- Depending on conditions on **u**, **u'**, **t'**, it covers various standard notions of security:
  - Secure against Chosen Plaintext Attack
  - Secure against Chosen Ciphertext Attack 1
  - Secure against Chosen Ciphertext Attack 2

History

- Initial version G. Bana - K. Hasebe - M. Okada 2008 (Franco-Japanese): Derive security with **FOL**
- New momentum G. Bana - H. Comon POST'12: Computationally complete symbolic attacker (**trace properties**)
    - Followup: G. Bana - P. Adao - H. Sakurada FSTTCS'12, Bana-Hasebe-Okada CCS'13
        - Library of axioms, analyzed several protocols for **agreement**, **authentication** for arbitrary number of sessions by hand, found **new attacks**
    - Hubert Comon student Guillaume Scerri's PhD thesis - simple verification **tool** Scary
- **Indistinguishability** G. Bana - H. Comon CCS'14: - basics with simple **anonymity**
    - Followup: G. Bana - R. Chadha (Univ. of Missouri) eprint'15
        - Library of axioms **digital signatures**, **CPA**, **CCA security**, **exponentiation**, **DDH assumption**, various versions of **Diffie-Hellman key exchange** for arbitrary number of sessions by hand, **NSL** protocol, **real-or random secrecy**, **anonymity**, **agreement**, **authentication**
- G. Scerri - R. Stanley-Oakes (Bristol) CCS'16: Security of **Key Wrapping API's**
- H. Comon - A. Koutsos CSF'17: proving **unlinkability and authentication of RFID protocols** (XOR)
- Recently completed with student of R. Chadha: Various versions of DH key exchange, Station-to-Station protocol proofs in **COQ**

Current Main Focuses

Current Main Focuses

- With Hubert Comon:

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand
- Hubert's student Adrien Koutsos works on automation

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand
- Hubert's student Adrien Koutsos works on automation
- With Mitsu Okada:

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand
- Hubert's student Adrien Koutsos works on automation
- With Mitsu Okada:
  - Completeness theorems - found attacks are real?

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand
- Hubert's student Adrien Koutsos works on automation
- With Mitsu Okada:
  - Completeness theorems - found attacks are real?
  - Exploring the connection with Fitting

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand
- Hubert's student Adrien Koutsos works on automation
- With Mitsu Okada:
  - Completeness theorems - found attacks are real?
  - Exploring the connection with Fitting
- At University of Luxembourg (with Peter Ryan)

Current Main Focuses

- With Hubert Comon:
  - efficient decision procedures
  - further examples by hand
- Hubert's student Adrien Koutsos works on automation
- With Mitsu Okada:
  - Completeness theorems - found attacks are real?
  - Exploring the connection with Fitting
- At University of Luxembourg (with Peter Ryan)
  - e-voting