Formal Methods and Cryptography

Lecture 25

Formal Methods

Formal Methods

- Logical foundations of computer science

Formal Methods

- Logical foundations of computer science
  - A language that "machines can understand"

Formal Methods

- Logical foundations of computer science
  - A language that “machines can understand”
  - To specify a computational procedure fully formally

Formal Methods

- Logical foundations of computer science
  - A language that "machines can understand"
  - To specify a computational procedure fully formally
  - Don't always need a computer: language abstracts away details not relevant to properties of interest

Formal Methods

- Logical foundations of computer science
  - A language that “machines can understand”
  - To specify a computational procedure fully formally
  - Don’t always need a computer: language abstracts away details not relevant to properties of interest
- Widely applied in practice

Formal Methods

- Logical foundations of computer science
  - A language that "machines can understand"
  - To specify a computational procedure fully formally
  - Don't always need a computer: language abstracts away details not relevant to properties of interest
- Widely applied in practice
  - Ensures that the procedures designed/analyzed and those implemented are the same

Formal Methods

- Logical foundations of computer science
  - A language that "machines can understand"
  - To specify a computational procedure fully formally
  - Don't always need a computer: language abstracts away details not relevant to properties of interest
- Widely applied in practice
  - Ensures that the procedures designed/analyzed and those implemented are the same
  - Can automate analysis of the designed procedures

Formal Methods in Cryptography

Formal Methods in Cryptography

- Motivation: security bugs even in simple protocols, if system is under-specified; exhaustive analysis by hand is error-prone

Formal Methods in Cryptography

- Motivation: security bugs even in simple protocols, if system is under-specified; exhaustive analysis by hand is error-prone
- A language to unambiguously specify cryptographic protocols and the whole system (in terms of basic building blocks)

Formal Methods in Cryptography

- Motivation: security bugs even in simple protocols, if system is under-specified; exhaustive analysis by hand is error-prone
- A language to unambiguously specify cryptographic protocols and the whole system (in terms of basic building blocks)
- Automated analysis

Formal Methods in Cryptography

- Motivation: security bugs even in simple protocols, if system is under-specified; exhaustive analysis by hand is error-prone
- A language to unambiguously specify cryptographic protocols and the whole system (in terms of basic building blocks)
- Automated analysis
    - Security definitions for various tasks are (were) often a list of intuitive high-level properties that must hold in adversarial environments

Formal Methods in Cryptography

- Motivation: security bugs even in simple protocols, if system is under-specified; exhaustive analysis by hand is error-prone
- A language to unambiguously specify cryptographic protocols and the whole system (in terms of basic building blocks)
- Automated analysis
  - Security definitions for various tasks are (were) often a list of intuitive high-level properties that must hold in adversarial environments
  - Formal methods Goal: to be able to analyze any given protocol and see if it satisfies these properties

Formal Methods in Cryptography

- Motivation: security bugs even in simple protocols, if system is under-specified; exhaustive analysis by hand is error-prone
- A language to unambiguously specify cryptographic protocols and the whole system (in terms of basic building blocks)
- Automated analysis
  - Security definitions for various tasks are (were) often a list of intuitive high-level properties that must hold in adversarial environments
  - Formal methods Goal: to be able to analyze any given protocol and see if it satisfies these properties
    - As opposed to finding one protocol (by hand) that satisfies the properties

Formal Methods in Cryptography

Formal Methods in Cryptography

- Outline:

Formal Methods in Cryptography

- Outline:
  - Develop a formal language for modeling the entire system (protocol, adversary, environment) and its evolution

Formal Methods in Cryptography

- Outline:
  - Develop a formal language for modeling the entire system (protocol, adversary, environment) and its evolution
    - Use abstractions of cryptographic primitives like encryption

Formal Methods in Cryptography

- Outline:
  - Develop a formal language for modeling the entire system (protocol, adversary, environment) and its evolution
    - Use abstractions of cryptographic primitives like encryption
  - Define security properties in this language

Formal Methods in Cryptography

- Outline:
  - Develop a formal language for modeling the entire system (protocol, adversary, environment) and its evolution
    - Use abstractions of cryptographic primitives like encryption
  - Define security properties in this language
  - Given any concrete protocol, map it to the formal language, and use standard formal method tools to automatically analyze it for the security properties

Formal Methods in Cryptography

- Outline:
  - Develop a formal language for modeling the entire system (protocol, adversary, environment) and its evolution
    - Use abstractions of cryptographic primitives like encryption
  - Define security properties in this language
  - Given any concrete protocol, map it to the formal language, and use standard formal method tools to automatically analyze it for the security properties
  - Ensure that security/insecurity in the formal model has useful implications in a more realistic model

Modeling

Modeling

- Typically, adversary controls the network

Modeling

- Typically, adversary controls the network
- A "process algebra" or a logic framework to describe what can happen in the system

Modeling

- Typically, adversary controls the network
- A "process algebra" or a logic framework to describe what can happen in the system
  - Dolev-Yao algebra: Parties can use keys to "encrypt" messages to get opaque symbols that can be operated on only if key is also provided. Deterministic encryption.

Modeling

- Typically, adversary controls the network
- A "process algebra" or a logic framework to describe what can happen in the system
  - Dolev-Yao algebra: Parties can use keys to "encrypt" messages to get opaque symbols that can be operated on only if key is also provided. Deterministic encryption.
  - BAN logic [Burrows-Abadi-Needham]: principals (parties) can "say" or "see" messages, and "believe" statements like "A said M" or "A believes B said M". Includes a notion of symmetric keys and public/private keys used for "encryption" (or rather, signcryption)

Modeling

- Typically, adversary controls the network
- A "process algebra" or a logic framework to describe what can happen in the system
  - Dolev-Yao algebra: Parties can use keys to "encrypt" messages to get opaque symbols that can be operated on only if key is also provided. Deterministic encryption.
  - BAN logic [Burrows-Abadi-Needham]: principals (parties) can "say" or "see" messages, and "believe" statements like "A said M" or "A believes B said M". Includes a notion of symmetric keys and public/private keys used for "encryption" (or rather, signcryption)
  - spi calculus: incorporates an "encryption" primitive into pi calculus which is used to model concurrent, communicating systems

Modeling

Modeling

- e.g. Dolev-Yao

Modeling

- e.g. Dolev-Yao
  - Term-rewriting algebra: operations that can lead to new events are defined by rules for writing new terms

Modeling

- e.g. Dolev-Yao
  - Term-rewriting algebra: operations that can lead to new events are defined by rules for writing new terms
  - Operations: send/receive terms; pick "nonces"; pair/separate; "encrypt"/"decrypt"

Modeling

- e.g. Dolev-Yao
  - Term-rewriting algebra: operations that can lead to new events are defined by rules for writing new terms
  - Operations: send/receive terms; pick "nonces"; pair/separate; "encrypt"/"decrypt"
    - For each user X, public operation E_X and private operation D_X

Modeling

- e.g. Dolev-Yao
  - Term-rewriting algebra: operations that can lead to new events are defined by rules for writing new terms
  - Operations: send/receive terms; pick "nonces"; pair/separate; "encrypt"/"decrypt"
    - For each user X, public operation E_X and private operation D_X
    - $D_X(E_X(m))$ can be rewritten as m

Modeling

- e.g. Dolev-Yao
  - Term-rewriting algebra: operations that can lead to new events are defined by rules for writing new terms
  - Operations: send/receive terms; pick "nonces"; pair/separate; "encrypt"/"decrypt"
    - For each user X, public operation E_X and private operation D_X
    - $D_X(E_X(m))$ can be rewritten as m
    - Separate(Pair(a,b)) gives a,b

Modeling

- e.g. Dolev-Yao
  - Term-rewriting algebra: operations that can lead to new events are defined by rules for writing new terms
  - Operations: send/receive terms; pick "nonces"; pair/separate; "encrypt"/"decrypt"
    - For each user X, public operation E_X and private operation D_X
    - $D_X(E_X(m))$ can be rewritten as m
    - Separate(Pair(a,b)) gives a,b
    - No other rewritings; each party can use terms it received and rewrite them (according to the protocol); adversary can obtain the closure of all terms sent out in the network

Security Properties - 1

Security Properties - 1

- Valid trace of a system: a sequence of events possible in the system (for the given protocol and an arbitrary adversary)

Security Properties - 1

- Valid trace of a system: a sequence of events possible in the system (for the given protocol and an arbitrary adversary)
  - Event: input/output/communication by parties or adversary

Security Properties - 1

- Valid trace of a system: a sequence of events possible in the system (for the given protocol and an arbitrary adversary)
  - Event: input/output/communication by parties or adversary
- Security property is defined for a trace, and a protocol is called secure if all valid traces satisfy the security property

Security Properties - 1

- Valid trace of a system: a sequence of events possible in the system (for the given protocol and an arbitrary adversary)
  - Event: input/output/communication by parties or adversary
- Security property is defined for a trace, and a protocol is called secure if all valid traces satisfy the security property
  - e.g.: For a key-agreement protocol, a trace is insecure if it has Alice outputting a nonce R (i.e., event [Alice:(output,R)]) and the adversary also outputting R (event [Eve:(output,R)])

Security Properties - 1

- Valid trace of a system: a sequence of events possible in the system (for the given protocol and an arbitrary adversary)
    - Event: input/output/communication by parties or adversary
- Security property is defined for a trace, and a protocol is called secure if all valid traces satisfy the security property
    - e.g.: For a key-agreement protocol, a trace is insecure if it has Alice outputting a nonce R (i.e., event [Alice:(output,R)]) and the adversary also outputting R (event [Eve:(output,R)])
    - e.g.: (in BAN logic) "(A believes B said X) at some point ⇒ (B said X) before that point"

Security Properties - 2

Security Properties - 2

- Security in spi calculus (inherited from pi calculus) essentially same as simulation-based security

Security Properties - 2

- Security in spi calculus (inherited from pi calculus) essentially same as simulation-based security
  - Observational Equivalence: Two systems P, Q are observationally equivalent if for all systems (environments) Z, the systems (Z|P) and (Z|Q) produce the same outputs

Security Properties - 2

- Security in spi calculus (inherited from pi calculus) essentially same as simulation-based security
  - Observational Equivalence: Two systems P, Q are observationally equivalent if for all systems (environments) Z, the systems (Z|P) and (Z|Q) produce the same outputs
  - To define security of a protocol, define an ideal protocol (think as ideal functionality, combined with a simulator for the "dummy adversary") and require that the two systems are observationally equivalent

Security Properties - 2

- Security in spi calculus (inherited from pi calculus) essentially same as simulation-based security
  - Observational Equivalence: Two systems P, Q are observationally equivalent if for all systems (environments) Z, the systems (Z|P) and (Z|Q) produce the same outputs
  - To define security of a protocol, define an ideal protocol (think as ideal functionality, combined with a simulator for the "dummy adversary") and require that the two systems are observationally equivalent
- Limitation: original spi calculus incorporated an ideal shared-key encryption and no other cryptographic features; extensions typically limited to secure communication tasks

An Example

An Example

- Needham-Schroeder-Lowe (public-key) protocol

An Example

- Needham-Schroeder-Lowe (public-key) protocol
  - For "mutual authentication"

An Example

- Needham-Schroeder-Lowe (public-key) protocol
  - For "mutual authentication"
  - Or, for "key agreement"

An Example

- Needham-Schroeder-Lowe (public-key) protocol
  - For "mutual authentication"
  - Or, for "key agreement"
- Uses an ideal encryption (or signcryption) to let two parties exchange nonces so that each should know that the nonce came from the other party (whose public-key it already has)

An Example

- Needham-Schroeder-Lowe (public-key) protocol
  - For "mutual authentication"
  - Or, for "key agreement"
- Uses an ideal encryption (or signcryption) to let two parties exchange nonces so that each should know that the nonce came from the other party (whose public-key it already has)
  - The nonce should be useful as a secret shared-key

An Example

- Needham-Schroeder-Lowe (public-key) protocol
  - For "mutual authentication"
  - Or, for "key agreement"
- Uses an ideal encryption (or signcryption) to let two parties exchange nonces so that each should know that the nonce came from the other party (whose public-key it already has)
  - The nonce should be useful as a secret shared-key
- Most formal frameworks use this example, to show that they can find the bug in the original Needham-Schroeder protocol (1978)

An Example

- Needham-Schroeder-Lowe (public-key) protocol
  - For "mutual authentication"
  - Or, for "key agreement"
- Uses an ideal encryption (or signcryption) to let two parties exchange nonces so that each should know that the nonce came from the other party (whose public-key it already has)
  - The nonce should be useful as a secret shared-key
- Most formal frameworks use this example, to show that they can find the bug in the original Needham-Schroeder protocol (1978)
  - Or new bugs in extended settings

Initiator ($\mathtt{M}_{\mathtt{init}}$)**:**

```
initialize(self, other);
newrandom(na);
pair(self, na, a_na);
encrypt(other, a_na, a_na_enc);
send(a_na_enc);
receive(b_na_nb_enc);
decrypt(self, b_na_nb_enc, b_na_nb);
separate(b_na_nb, b, na_nb);
test(b == other);
separate(na_nb, na2, nb);
test(na == na2);
encrypt(other, nb, nb_enc);
send(nb_enc);
pair(self, other, a_b);
pair(a_b, [x], a_b_x);
pair(Finished, a_b_x, out);
output(out);
done;
```

Responder ($\mathtt{M}_{resp}$)**:**

```
initialize(self, other);
receive(a_na_enc);
decrypt(self, a_na_enc, a_na);
separate(a_na, a, na);
test(a == other);
newrandom(nb);
pair(other, na, b_na);
pair(b_na, nb, b_na_nb);
encrypt(other, b_na_nb, b_na_nb_enc);
send(b_na_nb_enc);
receive(nb_enc);
decrypt(self, nb_enc, nb2);
test(nb == nb2);
pair(self, [x], b_a_x);
pair(Finished, b_a_x, out);
output(out);
done;
```

Version 1: x=na (Initiator's nonce output as secret key)
Version 2: x=nb (Responder's nonce output as secret key)

[NSL protocol, from Canetti-Herzog 2006]

Automated Analysis

Automated Analysis

- Not necessarily very efficient

Automated Analysis

- Not necessarily very efficient
  - Often NP-hard (or even P-SPACE hard). Typical algorithms are exponential in the size of the system

Automated Analysis

- Not necessarily very efficient
  - Often NP-hard (or even P-SPACE hard). Typical algorithms are exponential in the size of the system
  - Typically undecidable when allowing an unbounded number of concurrent sessions

Automated Analysis

- Not necessarily very efficient
  - Often NP-hard (or even P-SPACE hard). Typical algorithms are exponential in the size of the system
  - Typically undecidable when allowing an unbounded number of concurrent sessions
- Popular models (Dolev-Yao, BAN logic, spi calculus) have reasonably efficient algorithms for analyzing a variety of security properties, if the system is small (e.g., single session)

Automated Analysis

- Not necessarily very efficient
  - Often NP-hard (or even P-SPACE hard). Typical algorithms are exponential in the size of the system
  - Typically undecidable when allowing an unbounded number of concurrent sessions
- Popular models (Dolev-Yao, BAN logic, spi calculus) have reasonably efficient algorithms for analyzing a variety of security properties, if the system is small (e.g., single session)
  - Sometimes state-exploration (using model-checking tools) can be used to discover (some) flaws, but does not prove security

What does Security in a Formal Model mean?

What does Security in a Formal Model mean?

- "Encryption" as proposed in most of the formal models attributes message secrecy, key-anonymity, non-malleability, circular-encryption security, MAC/signature properties and much more (while requiring it to be deterministic)

What does Security in a Formal Model mean?

- "Encryption" as proposed in most of the formal models attributes message secrecy, key-anonymity, non-malleability, circular-encryption security, MAC/signature properties and much more (while requiring it to be deterministic)
  - Possibly achievable in random-oracle model or generic-group model

What does Security in a Formal Model mean?

- "Encryption" as proposed in most of the formal models attributes message secrecy, key-anonymity, non-malleability, circular-encryption security, MAC/signature properties and much more (while requiring it to be deterministic)
  - Possibly achievable in random-oracle model or generic-group model
- Security guarantee similar in spirit to these heuristic models

What does Security in a Formal Model mean?

- “Encryption” as proposed in most of the formal models attributes message secrecy, key-anonymity, non-malleability, circular-encryption security, MAC/signature properties and much more (while requiring it to be deterministic)
  - Possibly achievable in random-oracle model or generic-group model
- Security guarantee similar in spirit to these heuristic models
  - Security against adversaries who use only operations permitted by the formal model

What does Security in a Formal Model mean?

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?
  - Not quite, but maybe strong enough to translate the formal-model guarantees to security guarantees in the computational model

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?
  - Not quite, but maybe strong enough to translate the formal-model guarantees to security guarantees in the computational model
  - A formal model is "sound" if we can do the following:

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?
  - Not quite, but maybe strong enough to translate the formal-model guarantees to security guarantees in the computational model
  - A formal model is "sound" if we can do the following:
    - Translate protocol in computational model to formal model. Get security guarantee for it in formal model. This should imply security of the original protocol in the computational model

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?
  - Not quite, but maybe strong enough to translate the formal-model guarantees to security guarantees in the computational model
  - A formal model is "sound" if we can do the following:
    - Translate protocol in computational model to formal model. Get security guarantee for it in formal model. This should imply security of the original protocol in the computational model

In a specific format, using only specific primitives

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?
    - Not quite, but maybe strong enough to translate the formal-model guarantees to security guarantees in the computational model
- A formal model is "sound" if we can do the following:
    - Translate protocol in computational model to formal model. Get security guarantee for it in formal model. This should imply security of the original protocol in the computational model

In a specific format, using only specific primitives

If primitives satisfy certain security definitions

What does Security in a Formal Model mean?

- Can we develop strong underlying crypto primitives to implement the "encryption" as used in these formal models?
  - Not quite, but maybe strong enough to translate the formal-model guarantees to security guarantees in the computational model
- A formal model is "sound" if we can do the following:
  - Translate protocol in computational model to formal model. Get security guarantee for it in formal model. This should imply security of the original protocol in the computational model
  - Soundness of the formal model and formal security property for the computational task and primitive used

In a specific format, using only specific primitives

If primitives satisfy certain security definitions

Soundness of Formal Models

Soundness of Formal Models

- Initiated by Abadi-Rogaway (2001)

Soundness of Formal Models

- Initiated by Abadi-Rogaway (2001)
  - Shows soundness for a class of protocols/tasks: protocol secure for the task, if formal protocol has a certain security property in the formal model, and protocol uses CCA secure encryption in place of ideal encryptions in the formal model

Soundness of Formal Models

- Initiated by Abadi-Rogaway (2001)
  - Shows soundness for a class of protocols/tasks: protocol secure for the task, if formal protocol has a certain security property in the formal model, and protocol uses CCA secure encryption in place of ideal encryptions in the formal model
- Since then extended to various authentication/key-agreement-like tasks (and some computation tasks), against passive and active adversaries, using different formal models (algebras, spi-calculus)

Soundness of Formal Models

- Initiated by Abadi-Rogaway (2001)
  - Shows soundness for a class of protocols/tasks: protocol secure for the task, if formal protocol has a certain security property in the formal model, and protocol uses CCA secure encryption in place of ideal encryptions in the formal model
- Since then extended to various authentication/key-agreement-like tasks (and some computation tasks), against passive and active adversaries, using different formal models (algebras, spi-calculus)
- Recent works incorporate signatures, NIZK proofs etc.

Soundness of Formal Models

- Initiated by Abadi-Rogaway (2001)
  - Shows soundness for a class of protocols/tasks: protocol secure for the task, if formal protocol has a certain security property in the formal model, and protocol uses CCA secure encryption in place of ideal encryptions in the formal model
- Since then extended to various authentication/key-agreement-like tasks (and some computation tasks), against passive and active adversaries, using different formal models (algebras, spi-calculus)
- Recent works incorporate signatures, NIZK proofs etc.
- Typically each work considers a specific task, develops a security criterion in a specific formal model, and establishes soundness for protocols using specific crypto primitives (like CCA2 encryption)

Soundness of Formal Models

- Initiated by Abadi-Rogaway (2001)
  - Shows soundness for a class of protocols/tasks: protocol secure for the task, if formal protocol has a certain security property in the formal model, and protocol uses CCA secure encryption in place of ideal encryptions in the formal model
- Since then extended to various authentication/key-agreement-like tasks (and some computation tasks), against passive and active adversaries, using different formal models (algebras, spi-calculus)
- Recent works incorporate signatures, NIZK proofs etc.
- Typically each work considers a specific task, develops a security criterion in a specific formal model, and establishes soundness for protocols using specific crypto primitives (like CCA2 encryption)
  - Somewhat general frameworks: e.g., Backes et al. (CCS 2009)

Soundness of Formal Models

Soundness of Formal Models

- Several challenges

Soundness of Formal Models

- Several challenges
  - Traditional models usually deterministic (except for picking nonces, and possibly within the encryption operation), but for many interesting tasks cryptographic protocols typically use more randomness

Soundness of Formal Models

- Several challenges
  - Traditional models usually deterministic (except for picking nonces, and possibly within the encryption operation), but for many interesting tasks cryptographic protocols typically use more randomness
  - If model is too general, becomes hard/intractable to automatically reason

Soundness of Formal Models

- Several challenges
  - Traditional models usually deterministic (except for picking nonces, and possibly within the encryption operation), but for many interesting tasks cryptographic protocols typically use more randomness
  - If model is too general, becomes hard/intractable to automatically reason
    - Promising approach: Universal Composition -- require stronger per-session security that will allow decomposing the analysis to be per-session

Soundness of Formal Models

- Several challenges
  - Traditional models usually deterministic (except for picking nonces, and possibly within the encryption operation), but for many interesting tasks cryptographic protocols typically use more randomness
  - If model is too general, becomes hard/intractable to automatically reason
    - Promising approach: Universal Composition -- require stronger per-session security that will allow decomposing the analysis to be per-session
  - Only a few security properties have been considered (related to authentication and secure communication). Need to identify automatically verifiable (and sufficient) criteria for each new task

Universal Composition

Universal Composition

- Recall: security guarantee (in computational model) in terms of an ideal functionality (can be used in a formal model)

Universal Composition

- Recall: security guarantee (in computational model) in terms of an ideal functionality (can be used in a formal model)
  - From [GMW'87]. Used by [Pfitzmann-Waidner'01] and [Canetti'01]

Universal Composition

- Recall: security guarantee (in computational model) in terms of an ideal functionality (can be used in a formal model)
    - From [GMW'87]. Used by [Pfitzmann-Waidner'01] and [Canetti'01]
- UC Security [Canetti'01]: security is defined for one session of the protocol, in the presence of an arbitrary environment

Universal Composition

- Recall: security guarantee (in computational model) in terms of an ideal functionality (can be used in a formal model)
  - From [GMW'87]. Used by [Pfitzmann-Waidner'01] and [Canetti'01]
- UC Security [Canetti'01]: security is defined for one session of the protocol, in the presence of an arbitrary environment
- Composition Theorem: UC security of individual sessions automatically implies UC security of multiple concurrent sessions

Universal Composition

- Recall: security guarantee (in computational model) in terms of an ideal functionality (can be used in a formal model)
    - From [GMW'87]. Used by [Pfitzmann-Waidner'01] and [Canetti'01]
- UC Security [Canetti'01]: security is defined for one session of the protocol, in the presence of an arbitrary environment
- Composition Theorem: UC security of individual sessions automatically implies UC security of multiple concurrent sessions
    - Drawback: a strong security requirement that is more "expensive" to realize

Universal Composition

- Recall: security guarantee (in computational model) in terms of an ideal functionality (can be used in a formal model)
  - From [GMW'87]. Used by [Pfitzmann-Waidner'01] and [Canetti'01]
- UC Security [Canetti'01]: security is defined for one session of the protocol, in the presence of an arbitrary environment
- Composition Theorem: UC security of individual sessions automatically implies UC security of multiple concurrent sessions
  - Drawback: a strong security requirement that is more "expensive" to realize
  - Advantages: 1. Security for concurrent sessions. 2. Easy to use as a sub-module in higher level protocols and analyze security. Analysis of higher level protocols often "automatable"

Composition Logic

Composition Logic

- Ongoing research

Composition Logic

- Ongoing research
- Protocol Composition Logic of Mitchell et al.

Composition Logic

- Ongoing research
- Protocol Composition Logic of Mitchell et al.
- Formal model and soundness theorems by Canetti-Herzog

Composition Logic

- Ongoing research
- Protocol Composition Logic of Mitchell et al.
- Formal model and soundness theorems by Canetti-Herzog
- Task-Structured Probabilistic I/O Automata

Composition Logic

- Ongoing research
- Protocol Composition Logic of Mitchell et al.
- Formal model and soundness theorems by Canetti-Herzog
- Task-Structured Probabilistic I/O Automata
- ...

Secure Computation?

Secure Computation?

- Most tasks formally analyzed relate to secure communication

Secure Computation?

- Most tasks formally analyzed relate to secure communication
- UC framework in principle allows arbitrary functionalities

Secure Computation?

- Most tasks formally analyzed relate to secure communication
- UC framework in principle allows arbitrary functionalities
- Also, possibility of modeling certain homomorphic encryption schemes algebraically (and in a sound manner) if implemented using "non-malleable" homomorphic encryption

Secure Computation?

- Most tasks formally analyzed relate to secure communication
- UC framework in principle allows arbitrary functionalities
- Also, possibility of modeling certain homomorphic encryption schemes algebraically (and in a sound manner) if implemented using "non-malleable" homomorphic encryption
- Challenge: Efficient automated analysis in the resulting formal model

More Automation?

More Automation?

- Formal models are used to analyze higher level protocols, reducing their security to the security of underlying cryptographic primitives

More Automation?

- Formal models are used to analyze higher level protocols, reducing their security to the security of underlying cryptographic primitives
- Crypto primitives themselves designed and security reduced to computational complexity assumptions by hand

More Automation?

- Formal models are used to analyze higher level protocols, reducing their security to the security of underlying cryptographic primitives
- Crypto primitives themselves designed and security reduced to computational complexity assumptions by hand
- Can this be automated?

More Automation?

- Formal models are used to analyze higher level protocols, reducing their security to the security of underlying cryptographic primitives
- Crypto primitives themselves designed and security reduced to computational complexity assumptions by hand
- Can this be automated?
  - Plausible, if a formal model of complexity assumptions

More Automation?

- Formal models are used to analyze higher level protocols, reducing their security to the security of underlying cryptographic primitives
- Crypto primitives themselves designed and security reduced to computational complexity assumptions by hand
- Can this be automated?
  - Plausible, if a formal model of complexity assumptions
  - Likely, for generic group model (which is a formal model)

More Automation?

- Formal models are used to analyze higher level protocols, reducing their security to the security of underlying cryptographic primitives
- Crypto primitives themselves designed and security reduced to computational complexity assumptions by hand
- Can this be automated?
  - Plausible, if a formal model of complexity assumptions
  - Likely, for generic group model (which is a formal model)
  - Recent developments in machine verifiable, machine-assisted proofs: EasyCrypt/CertiCrypt

Today

Today

- Use of formal methods in cryptography

Today

- Use of formal methods in cryptography
  - Prior to 2000 (or Abadi-Rogaway), separate communities

Today

- Use of formal methods in cryptography
    - Prior to 2000 (or Abadi-Rogaway), separate communities
        - Dolev-Yao, spi calculus, BAN logic

Today

- Use of formal methods in cryptography
  - Prior to 2000 (or Abadi-Rogaway), separate communities
    - Dolev-Yao, spi calculus, BAN logic
    - Security in formal model had little bearing as a security guarantee in the computational model (but attacks in the formal model give real attacks)

Today

- Use of formal methods in cryptography
  - Prior to 2000 (or Abadi-Rogaway), separate communities
    - Dolev-Yao, spi calculus, BAN logic
    - Security in formal model had little bearing as a security guarantee in the computational model (but attacks in the formal model give real attacks)
  - Soundness guarantees

Today

- Use of formal methods in cryptography
  - Prior to 2000 (or Abadi-Rogaway), separate communities
    - Dolev-Yao, spi calculus, BAN logic
    - Security in formal model had little bearing as a security guarantee in the computational model (but attacks in the formal model give real attacks)
  - Soundness guarantees
    - Security in formal models that can be translated to security in computational models

Today

- Use of formal methods in cryptography
  - Prior to 2000 (or Abadi-Rogaway), separate communities
    - Dolev-Yao, spi calculus, BAN logic
    - Security in formal model had little bearing as a security guarantee in the computational model (but attacks in the formal model give real attacks)
  - Soundness guarantees
    - Security in formal models that can be translated to security in computational models
  - Composition: to make analysis of complex protocols feasible; also to obtain security in arbitrary environments

Today

- Use of formal methods in cryptography
  - Prior to 2000 (or Abadi-Rogaway), separate communities
    - Dolev-Yao, spi calculus, BAN logic
    - Security in formal model had little bearing as a security guarantee in the computational model (but attacks in the formal model give real attacks)
  - Soundness guarantees
    - Security in formal models that can be translated to security in computational models
  - Composition: to make analysis of complex protocols feasible; also to obtain security in arbitrary environments
  - Ongoing work: Probabilistic models (e.g. Task PIOA), more tasks, more tools for formal analysis