Introduction à la sécurité informatique - 5
Modèle Dolev-Yao

Frédéric Prost
Frederic.Prost@univ-grenoble-alpes.fr

UGA

8 Février 2023

Plan

Introduction

- Cryptographic primitives are usually only a part of wider schemes.
- In order to study the security of whole protocols it is often advantageous to have an abstract view of cryptographic operations.

Introduction

- Cryptographic primitives are usually only a part of wider schemes.
- In order to study the security of whole protocols it is often advantageous to have an abstract view of cryptographic operations.

  $\Longrightarrow$ the aim is to work with a high-level description of what encryption primitives are supposed to achieve.
- It is similar to high-level programming approach to programming vs circuit design or TM programming...

Introduction

- There are two main ways to “prove” security from an abstract point of view:

Introduction

- There are two main ways to “prove” security from an abstract point of view:
    1. Symbolic approach: cryptographic functions are seen as function on symbolic space. Security properties are formally defined.

Introduction

- There are two main ways to “prove” security from an abstract point of view:
    1. Symbolic approach: cryptographic functions are seen as function on symbolic space. Security properties are formally defined.
    2. Computational approach: cryptographic functions are seen as functions on bit strings. Security properties are defined in terms of probability and complexity.

Introduction

- There are two main ways to “prove” security from an abstract point of view:
  1. Symbolic approach: cryptographic functions are seen as function on symbolic space. Security properties are formally defined.
  2. Computational approach: cryptographic functions are seen as functions on bit strings. Security properties are defined in terms of probability and complexity.
- How do they relate to one another?

Introduction

- There are two main ways to “prove” security from an abstract point of view:
  1. Symbolic approach: cryptographic functions are seen as function on symbolic space. Security properties are formally defined.
  2. Computational approach: cryptographic functions are seen as functions on bit strings. Security properties are defined in terms of probability and complexity.
- How do they relate to one another?

$\Longrightarrow$ Computational soundness.

Symbolic vs Computational approaches

- Computational is more “solid”.

Symbolic vs Computational approaches

- Computational is more “solid”.
- Computational is more “artistic”: for each protocol, cryptographic functions, one has to build a specific proof.

Symbolic vs Computational approaches

- Computational is more "solid".
- Computational is more "artistic": for each protocol, cryptographic functions, one has to build a specific proof.
- Symbolic allows to make more elaborated proofs: protocols are more and more complex and built as subtle combinations of basic cryptographic primitives.

Symbolic vs Computational approaches

- Computational is more “solid”.
- Computational is more “artistic”: for each protocol, cryptographic functions, one has to build a specific proof.
- Symbolic allows to make more elaborated proofs: protocols are more and more complex and built as subtle combinations of basic cryptographic primitives.
- Symbolic allows automated proof and, hopefully, modular proof approaches.

The Formal View

- Cryptographic operations are seen as purely formal: $\{M\}_K$
  M and K are formal expressions, not sequences of bits.

The Formal View

- Cryptographic operations are seen as purely formal: $\{M\}_K$
  M and K are formal expressions, not sequences of bits.
- An algebra among such formal terms can be applied: typically $\{\{M\}_K\}_{\overline{K}} = M$
  All-or-nothing kind of approach (no probability or incomplete leakages).

The Formal View

- Cryptographic operations are seen as purely formal: $\{M\}_K$
  M and K are formal expressions, not sequences of bits.
- An algebra among such formal terms can be applied: typically $\{\{M\}_K\}_{\overline{K}} = M$
  All-or-nothing kind of approach (no probability or incomplete leakages).
  Counter example ? $\{M\}_K = M + K$ and K used twice...

The Formal View

- Cryptographic operations are seen as purely formal: $\{M\}_K$
  M and K are formal expressions, not sequences of bits.
- An algebra among such formal terms can be applied: typically
  $\{\{M\}_K\}_{\overline{K}} = M$
  All-or-nothing kind of approach (no probability or incomplete leakages).
  Counter example ? $\{M\}_K = M + K$ and K used twice...
- Starts with the work of Dolev and Yao [Dolev and Yao, 1983] extensively used to prove the safety of some protocols and also to discover many attacks.
- Leads to the development of effective methods and automatic tools for automated protocol analysis.

$\Longrightarrow$ There is a gap between the ideal representation of encryption in a formal model and its concrete implementation.

The Computational View

- Based on complexity theory.
- A proponent of this approach would say that formal approaches are naïve and disconnected from the reality.
- Here, keys, messages are just srtings of bits. Encryption is just an algorithm. The adversary is a Turing Machine.
- Good protocols are the one in which adversaries cannot do "something bad" too often and efficiently enough.

  $\Longrightarrow$ the notion of advantage gained.

Plan

Plan

The Dolev-Yao Formal Model of Security[Dolev and Yao, 1983]

- In the Needham-Schroeder protocol [Needham and Schroeder, 1978] of identification flaws were found after the publication of the paper. It triggered the interest for formal security protocol analysis tools.
- The Dolev-Yao model is the first formal method proposal.
- The original model is very constrained and does not allow to describe many interesting protocols. Still it is interesting because:
    - First proposition of formal model.

The Dolev-Yao Formal Model of Security[Dolev and Yao, 1983]

- In the Needham-Schroeder protocol [Needham and Schroeder, 1978] of identification flaws were found after the publication of the paper. It triggered the interest for formal security protocol analysis tools.
- The Dolev-Yao model is the first formal method proposal.
- The original model is very constrained and does not allow to describe many interesting protocols. Still it is interesting because:
  - First proposition of formal model.
  - Restriction are mostly on the honest protocol participants and security goal. Adversaries are quite general.

The Dolev-Yao Formal Model of Security[Dolev and Yao, 1983]

- In the Needham-Schroeder protocol [Needham and Schroeder, 1978] of identification flaws were found after the publication of the paper. It triggered the interest for formal security protocol analysis tools.
- The Dolev-Yao model is the first formal method proposal.
- The original model is very constrained and does not allow to describe many interesting protocols. Still it is interesting because:
  - First proposition of formal model.
  - Restriction are mostly on the honest protocol participants and security goal. Adversaries are quite general.
  - Restricting the class of target protocols allows interesting results like: security is decidable in polynomial time, it can be automated.

What is Looked for in a Formal Model?

If you want to unambiguously answer to the question: is this protocol secure or not ? What do you need ?

What is Looked for in a Formal Model?

If you want to unambiguously answer to the question: is this protocol secure or not ? What do you need ?

1. Precise language for descriptions of protocols.

What is Looked for in a Formal Model?

If you want to unambiguously answer to the question: is this protocol secure or not ? What do you need ?

1. Precise language for descriptions of protocols.
2. Formal execution model (kind of operational semantics of the protocol), possibly in the presence of an adversary. It includes a descrption of adversary's capabilities: typically, starting the execution of an arbitrary instances of the protocol among anyone (honnest players and adversary).

What is Looked for in a Formal Model?

If you want to unambiguously answer to the question: is this protocol secure or not ? What do you need ?

1. Precise language for descriptions of protocols.
2. Formal execution model (kind of operational semantics of the protocol), possibly in the presence of an adversary. It includes a descrption of adversary's capabilities: typically, starting the execution of an arbitrary instances of the protocol among anyone (honnest players and adversary).
3. A formal language for specifying desired security protocols.

Dolev-Yao Model

- The focus is on two party protocols and secrecy properties.

Dolev-Yao Model

- The focus is on two party protocols and secrecy properties.
- Important features of DY model:
  1. **Secrecy properties:** The only goal is to send M in a secret way.

Dolev-Yao Model

- The focus is on two party protocols and secrecy properties.

- Important features of DY model:

  1. **Secrecy properties:** The only goal is to send M in a secret way.
  2. **Stateless parties:** The messages transmitted by a party at every step of the protocol are a function of their initial knowledge and the message they just received. In particular, parties cannot use information collected from previous messages.

Dolev-Yao Model

- The focus is on two party protocols and secrecy properties.
- Important features of DY model:
  1. **Secrecy properties:** The only goal is to send M in a secret way.
  2. **Stateless parties:** The messages transmitted by a party at every step of the protocol are a function of their initial knowledge and the message they just received. In particular, parties cannot use information collected from previous messages.
  3. **Concurrent execution:** The adversary can start an arbitrary number of protocol executions, involving different sets of parties, where each player can participate in several concurrent executions.

Dolev-Yao Model

- The focus is on two party protocols and secrecy properties.
- Important features of DY model:
  1. **Secrecy properties:** The only goal is to send M in a secret way.
  2. **Stateless parties:** The messages transmitted by a party at every step of the protocol are a function of their initial knowledge and the message they just received. In particular, parties cannot use information collected from previous messages.
  3. **Concurrent execution:** The adversary can start an arbitrary number of protocol executions, involving different sets of parties, where each player can participate in several concurrent executions.
  4. **Public Key cryptography and infrastructure:** It is assumed that a public table (X, E_X) containing the name and public key of every user is publicly available. The initial knowledge of each user consists of this table, plus the user secret decryption key D_X.

Example 1:

- Alice sends an encrypted message to Bob and waits for an echo in acknowledgment:

$$\begin{array}{ll} 1. & A \rightarrow B : \quad \{M\}_B \\ 2. & B \rightarrow A : \quad \{M\}_A \end{array}$$

Example 1:

- Alice sends an encrypted message to Bob and waits for an echo in acknowledgment:

$$\begin{array}{lll} 1. & A \rightarrow B: & \{M\}_B \\ 2. & B \rightarrow A: & \{M\}_A \end{array}$$

- This protocol is insecure. Formal attack goes like this:

$$\begin{array}{llll} 1. & A \rightarrow Z: & \{M\}_B & Z \text{ intercepts the message} \end{array}$$

Example 1:

- Alice sends an encrypted message to Bob and waits for an echo in acknowledgment:

$$\begin{array}{lll} 1. & A \rightarrow B: & \{M\}_B \\ 2. & B \rightarrow A: & \{M\}_A \end{array}$$

- This protocol is insecure. Formal attack goes like this:

$$\begin{array}{llll} 1. & A \rightarrow Z: & \{M\}_B & Z \text{ intercepts the message} \\ 2. & Z \rightarrow B: & \{M\}_B & \end{array}$$

Example 1:

- Alice sends an encrypted message to Bob and waits for an echo in acknowledgment:

$$\begin{array}{lll} 1. & A \to B : & \{M\}_B \\ 2. & B \to A : & \{M\}_A \end{array}$$

- This protocol is insecure. Formal attack goes like this:

$$\begin{array}{llll} 1. & A \to Z : & \{M\}_B & Z \text{ intercepts the message} \\ 2. & Z \to B : & \{M\}_B & \\ 3. & B \to Z : & \{M\}_Z & \text{since } B \text{ follows the protocol, } Z \text{ can recover } M \end{array}$$

Example 1:

- Alice sends an encrypted message to Bob and waits for an echo in acknowledgment:

$$\begin{array}{ll} 1. & A \rightarrow B : \quad \{M\}_B \\ 2. & B \rightarrow A : \quad \{M\}_A \end{array}$$

- This protocol is insecure. Formal attack goes like this:

$$\begin{array}{llll} 1. & A \rightarrow Z : & \{M\}_B & Z \text{ intercepts the message} \\ 2. & Z \rightarrow B : & \{M\}_B & \\ 3. & B \rightarrow Z : & \{M\}_Z & \text{since } B \text{ follows the protocol, } Z \text{ can recover } M \\ 4. & Z \rightarrow A : & \{M\}_A & \text{optional so that even } A \\ & & & \text{does'nt notice the protocol has been broken} \end{array}$$

Example 2:

- Let's try to fix the protocol by adding the name and an extra layer of encryption:

$$\begin{array}{lll} 1. & A \rightarrow B: & \{\{M\}_B; A\}_B \\ 2. & B \rightarrow A: & \{\{M\}_A; B\}_A \end{array}$$

Example 2:

- Let's try to fix the protocol by adding the name and an extra layer of encryption:

$$\begin{array}{lll} 1. & A \to B : & \{\{M\}_B; A\}_B \\ 2. & B \to A : & \{\{M\}_A; B\}_A \end{array}$$

  is it secure (from DY point of view) ?

Example 2:

- Let's try to fix the protocol by adding the name and an extra layer of encryption:

$$\begin{array}{lll} 1. & A \rightarrow B: & \{\{M\}_B; A\}_B \\ 2. & B \rightarrow A: & \{\{M\}_A; B\}_A \end{array}$$

  is it secure (from DY point of view) ?

- No! Here is a (formal) attack.

Example 2:

1. Z intercepts a protocol execution between A and B with message M, and intercepts the last message $\{M'\}_A$ where $M' = \{M\}_A; B$ (as before).

Example 2:

1. Z intercepts a protocol execution between A and B with message M, and intercepts the last message $\{M'\}_A$ where $M' = \{M\}_A; B$ (as before).
2. Z starts another protocol between Z and A with message M', using its knowledge of $\{M'\}_A$:

$$\begin{array}{lll} 1. & Z \to A: & \{\{M'\}_A; Z\}_A \\ 2. & A \to Z: & \{\{M'\}_Z; A\}_Z \end{array}$$

Now Z can decrypt and recover $M' = \{M\}_A; B$. Dropping the last B, this gives $\{M\}_A$.

Example 2:

1. Z intercepts a protocol execution between A and B with message M, and intercepts the last message $\{M'\}_A$ where $M' = \{M\}_A; B$ (as before).
2. Z starts another protocol between Z and A with message M', using its knowledge of $\{M'\}_A$:

$$\begin{array}{lll} 1. & Z \rightarrow A: & \{\{M'\}_A; Z\}_A \\ 2. & A \rightarrow Z: & \{\{M'\}_Z; A\}_Z \end{array}$$

   Now Z can decrypt and recover $M' = \{M\}_A; B$. Dropping the last B, this gives $\{M\}_A$.
3. Z starts another interaction with A:

$$\begin{array}{lll} 1. & Z \rightarrow A: & \{\{M\}_A; Z\}_A \\ 2. & A \rightarrow Z: & \{\{M\}_Z; A\}_Z \end{array}$$

   At this point, Z can decrypt and recover the original message M which was intented for B only

DY Model: Protocols considered

The DY model considered focuses on 2 party protocols, executed concurrently in a network with an arbitrary number of participants.

- The protocol involves two parties: S (the sender) and R (the receiver) $S(M, R)$ takes an input message M, and an identity R of the party S wants to send the message M to.
- The receiver is ready to engage in a protocol execution with any sender.
- Each protocol step is modeled as a function mapping the last received message to a new message to be transmitted. These functions can be the composition of any number of basic functions chosen from a given set F_X of basic functions available to user X.

DY Model: Basic Operations

DY considers two kinds of protocols (corresponding to two sets of basic functions) called cascade protocols and namestamp protocols. The latter is a generalization of the first one, so we concentrate on namestamp protocols. The basic operations available to party X are:

- D_X (decryption under X's secret key)
- E_Y (encryption under any user Y's public key)
- i_y (append identifier y to the message)
- d_y (delete identifier y from the end of the message). If input message does not end in y, then abort.
- d (delete identifier at the end of the message)

DY model: Formal Description of a Protocol

A two party protocol is formally described as a sequence of strings $f[1], f[2], ..., f[k]$ where for any i, $f[2i+1]$ is a string over the function symbols available to S, and $f[2i]$ is a string over the function symbols available to R.

DY model: Formal Description of a Protocol

A two party protocol is formally described as a sequence of strings $f[1], f[2], ..., f[k]$ where for any i, $f[2i+1]$ is a string over the function symbols available to S, and $f[2i]$ is a string over the function symbols available to R.

- $f[1]$ is the function applied by the sender to the input message M to determine the first message sent to R.
- $f[2i]$ is the function applied by R to the *ith* received message to determine the next message to be transmitted to S.
- $f[2i+1]$ is the function applied by S to the *ith* received message to determine the next message to be transmitted to R.

S and R in the above description are two generic party names, and the protocol can be instantiated replacing S and R with any other pair of parties. Replacing S and R in $f[i]$ with A and B is denoted $f[i]\{A, B\}$.

DY model: Composition and Cancellation Rules

- For any i, let $F[i](M) = f[i](f[i-1](\ldots f[2](f[1](M))\ldots)$ be the composition of the first i functions.

DY model: Composition and Cancellation Rules

- For any i, let $F[i](M) = f[i](f[i-1](...f[2](f[1](M))...)$ be the composition of the first i functions.

  The sequence of message transmitted during the execution of protocol on input M are $F[1](M), ...F[k](M)$.

DY model: Composition and Cancellation Rules

- For any i, let $F[i](M) = f[i](f[i-1](\ldots f[2](f[1](M))\ldots)$ be the composition of the first i functions.

  The sequence of message transmitted during the execution of protocol on input M are $F[1](M), \ldots F[k](M)$.

- Strings of function symbols are interpreted modulo the following cancellation rules:

$$
\begin{array}{lcl}
D_x E_x & = & \epsilon \\
E_x D_x & = & \epsilon \\
d_x i_x & = & \epsilon \\
d i_x & = & \epsilon
\end{array}
$$

  where ϵ is the empty string, representing the identity function.

DY model: Composition and Cancellation Rules

- This set of operations can be easily generalized. E.g., strings are taken to represent functions, and in particular, the set of cancellation rules should satisfy the property that if $fw = gw$ for any string w, then f and g are the same function (symbol).

- The above rules satisfy these properties.

  An immediate consequence is that if f has both a left and right inverse $lf = fr = id$, then $l = r$ and this inverse is unique.

Examples

- Example 1:

$$\begin{array}{lll} 1. & S \rightarrow R: & \{M\}_R \\ 2. & R \rightarrow S: & \{M\}_S \end{array}$$

  is modeled by the sequence of strings:

Examples

- Example 1:

$$\begin{array}{lll} 1. & S \rightarrow R : & \{M\}_R \\ 2. & R \rightarrow S : & \{M\}_S \end{array}$$

is modeled by the sequence of strings:

$$\begin{array}{ll} 1. & E_R \\ 2. & E_S D_R \end{array}$$

Examples

- Example 1:

$$\begin{array}{lll} 1. & S \rightarrow R: & \{M\}_R \\ 2. & R \rightarrow S: & \{M\}_S \end{array}$$

is modeled by the sequence of strings:

$$\begin{array}{ll} 1. & E_R \\ 2. & E_S D_R \end{array}$$

- Example 2:

$$\begin{array}{lll} 1. & S \rightarrow R: & \{\{M\}_R; S\}_R \\ 2. & R \rightarrow S: & \{\{M\}_S; R\}_S \end{array}$$

is modeled by the sequence of strings:

Examples

- Example 1:

$$\begin{array}{lll} 1. & S \rightarrow R: & \{M\}_R \\ 2. & R \rightarrow S: & \{M\}_S \end{array}$$

  is modeled by the sequence of strings:

$$\begin{array}{ll} 1. & E_R \\ 2. & E_S D_R \end{array}$$

- Example 2:

$$\begin{array}{lll} 1. & S \rightarrow R: & \{\{M\}_R; S\}_R \\ 2. & R \rightarrow S: & \{\{M\}_S; R\}_S \end{array}$$

  is modeled by the sequence of strings:

$$\begin{array}{ll} 1. & E_R i_S E_R \\ 2. & E_S i_R E_S D_R d_S D_R \end{array}$$

Formal Execution Model

DY considers a model where an active attacker can interfere with the concurrent execution of an arbitrary number of protocol executions.

- Let U be a potentially infinite pool of user names. Some of the users in U are honest (H) and some are corrupted (C). The attacker can start an arbitrary number of protocol executions between parties in U, honest and dishonest ones.
- The goal of the adversary is to recover the message M underlying a protocol execution between two honest parties A and B.
- The attacker is assumed to have total control of the network: in other words, the adversary **IS** the network.

Formal Excution Model: Adversary Functions

Under the DY execution model the adversary has access to the following functions:

- $f[i]$ where $i \geq 1$ and S, R are replaced by any pair of distinct parties in U.
- E_X, i_X, d_X and d for any party X in U.
- D_X for any dishonest party X in C

Moreover, the adversary can obtain the values $F[i]\{A, B\}(M)$ for any i and honest parties A, B.

Formal Excution Model: Secure Protocol Definition

The goal of the adversary is to recover M.

Formal Excution Model: Secure Protocol Definition

The goal of the adversary is to recover M.

Equivalently, the adversary's goal is to find a sequence of functions $[g_1, ..., g_k]$ such that $g_k \circ ... g_2 \circ g_1 \circ f[1]\{A, B\} = id$ for some honest parties A and B. Hence the definion:

Formal Excution Model: Secure Protocol Definition

The goal of the adversary is to recover M.

Equivalently, the adversary's goal is to find a sequence of functions $[g_1, ..., g_k]$ such that $g_k \circ ...g_2 \circ g_1 \circ f[1]\{A, B\} = id$for some honest parties A and B. Hence the definion:

Definition

Let $f[1], ..., f[r]$ be a two party protocol between a sender S and receiver R. The protocol is insecure if and only if for some honest parties A, B, the adversary has access to a sequence of functions $g_1, ..., g_k$ such that $g_k \circ ...g_2 \circ g_1 \circ f[1]\{A, B\} = id$.

Formal Excution Model: Secure Protocol Definition

The goal of the adversary is to recover M.

Equivalently, the adversary's goal is to find a sequence of functions $[g_1, ..., g_k]$ such that $g_k \circ ...g_2 \circ g_1 \circ f[1]\{A, B\} = id$for some honest parties A and B. Hence the definion:

Definition

Let $f[1], ..., f[r]$ be a two party protocol between a sender S and receiver R. The protocol is insecure if and only if for some honest parties A, B, the adversary has access to a sequence of functions $g_1, ..., g_k$ such that $g_k \circ ...g_2 \circ g_1 \circ f[1]\{A, B\} = id$.

The remaining question: Can such a security property be decided? Efficiently decided?

Formal Excution Model: Reduction Theorem

The answer is yes but there are problems due to the unbounded number of participants. One has to show that we can always restrict the number of parties to 3: 2 honest parties A, B and the adversary Z.

Formal Excution Model: Reduction Theorem

The answer is yes but there are problems due to the unbounded number of participants. One has to show that we can always restrict the number of parties to 3: 2 honest parties A, B and the adversary Z.

Theorem

Let $f[1], ..., f[r]$ be a DY protocol. If the protocol is insecure, then there is a sequence of functions $[g_1, ..., g_k]$ and pair of parties A, B demonstrating the insecurity, where all the parties involved in the functions are from A, B and Z.

Formal Excution Model: Reduction Theorem

Proof sketch:
Assume $g_k \circ ...g_2 \circ g_1 \circ f[1]\{A, B\} = id$ is an attack. We obtain an attack involving only A, B and Z by replacing all identifiers different from A and B with Z. Since the substitution can only give more cancellations, we still have $g'_k \circ ... \circ g'_1 \circ f[1]\{A, B\} = id$.

Formal Excution Model: Reduction Theorem

Proof sketch:
Assume $g_k \circ ... g_2 \circ g_1 \circ f[1]\{A, B\} = id$ is an attack. We obtain an attack involving only A, B and Z by replacing all identifiers different from A and B with Z. Since the substitution can only give more cancellations, we still have $g'_k \circ ... \circ g'_1 \circ f[1]\{A, B\} = id$.

- If g_k is D_X, E_X, i_X, d_X or d, for some X different from A, B, then the resulting function is D_Z, E_Z, i_Z, d_Z, d and adversary Z is allowed to use this function.

Formal Excution Model: Reduction Theorem

Proof sketch:

Assume $g_k \circ ...g_2 \circ g_1 \circ f[1]\{A, B\} = id$ is an attack. We obtain an attack involving only A, B and Z by replacing all identifiers different from A and B with Z. Since the substitution can only give more cancellations, we still have $g'_k \circ ... \circ g'_1 \circ f[1]\{A, B\} = id$.

- If g_k is D_X, E_X, i_X, d_X or d, for some X different from A, B, then the resulting function is D_Z, E_Z, i_Z, d_Z, d and adversary Z is allowed to use this function.
- If g_k is $f[i]\{A, B\}$ or $f[i]\{B, A\}$, then $g'_k = g_k$ is an allowed function

Formal Excution Model: Reduction Theorem

Proof sketch:
Assume $g_k \circ ... g_2 \circ g_1 \circ f[1]\{A, B\} = id$ is an attack. We obtain an attack involving only A, B and Z by replacing all identifiers different from A and B with Z. Since the substitution can only give more cancellations, we still have $g'_k \circ ... \circ g'_1 \circ f[1]\{A, B\} = id$.

- If g_k is D_X, E_X, i_X, d_X or d, for some X different from A, B, then the resulting function is D_Z, E_Z, i_Z, d_Z, d and adversary Z is allowed to use this function.
- If g_k is $f[i]\{A, B\}$ or $f[i]\{B, A\}$, then $g'_k = g_k$ is an allowed function
- If g_k is $f[i]\{A, C\}, f[i]\{B, C\}, f[i]\{C, A\}$ or $f[i]\{C, B\}$ for some C different from A and B, then the new function g'_k is identical to g_k, except for replacing C with Z.

Formal Excution Model: Reduction Theorem

Proof sketch:

Assume $g_k \circ ... g_2 \circ g_1 \circ f[1]\{A, B\} = id$ is an attack. We obtain an attack involving only A, B and Z by replacing all identifiers different from A and B with Z. Since the substitution can only give more cancellations, we still have $g'_k \circ ... \circ g'_1 \circ f[1]\{A, B\} = id$.

- If g_k is D_X, E_X, i_X, d_X or d, for some X different from A, B, then the resulting function is D_Z, E_Z, i_Z, d_Z, d and adversary Z is allowed to use this function.
- If g_k is $f[i]\{A, B\}$ or $f[i]\{B, A\}$, then $g'_k = g_k$ is an allowed function
- If g_k is $f[i]\{A, C\}, f[i]\{B, C\}, f[i]\{C, A\}$ or $f[i]\{C, B\}$ for some C different from A and B, then the new function g'_k is identical to g_k, except for replacing C with Z.
- If g_k is $f[i]\{C, D\}$ for two parties C, D not in $\{A, B\}$, then $g'_k = f[i]\{Z, Z\}$ is the composition of functions of the form D_Z, E_Z, i_Z, d_Z, d, which are all allowed.

Decidability of Formal Execution Model

- We use the Theorem to reduce the problem of testing the security of a protocol to a special case of the same problem where the number of parties is bounded by 3 and the adversary has access only to a finite number of functions.

Decidability of Formal Execution Model

- We use the Theorem to reduce the problem of testing the security of a protocol to a special case of the same problem where the number of parties is bounded by 3 and the adversary has access only to a finite number of functions.
  Though the length of the attack is still potentially unbounded!
  $\Longrightarrow$ so it is not clear if the problem can be solved algorithmically.

Decidability of Formal Execution Model

- We use the Theorem to reduce the problem of testing the security of a protocol to a special case of the same problem where the number of parties is bounded by 3 and the adversary has access only to a finite number of functions.
  Though the length of the attack is still potentially unbounded!
  $\Longrightarrow$ so it is not clear if the problem can be solved algorithmically.
- DY shows that the problem is indeed decidable, and moreover, there is an efficient (polynomial time) decision procedure.
- The running time of the decision procedure of DY is n^3.

Decidability Procedure of Formal Execution Model

1. Consider the set of all words over the alphabet $\{E_A, E_B, E_Z, D_A, D_B, D_Z, i_A, i_B, i_Z, d_A, d_B, d_Z, d\}$ that simplify to the empty string using the cancellation rules $D_X E_X = E_X D_X = d_X i_X = d i_X = \epsilon$.
2. This set of words is context free and can be generated by a context free grammar with rules $S \to \epsilon | D_X S E_X S | \ldots$ and so on for all cancellation rules.

Decidability Procedure of Formal Execution Model

1. Consider the set of all words over the alphabet $\{E_A, E_B, E_Z, D_A, D_B, D_Z, i_A, i_B, i_Z, d_A, d_B, d_Z, d\}$ that simplify to the empty string using the cancellation rules $D_X E_X = E_X D_X = d_X i_X = di_X = \epsilon$.
2. This set of words is context free and can be generated by a context free grammar with rules $S \to \epsilon | D_X S E_X S | \dots$ and so on for all cancellation rules.
   The grammar can be easily converted into an equivalent Push Down Automaton. Notice that the size of this automaton is constant because it does not depend on the protocol.

Decidability Procedure of Formal Execution Model

1. Consider the set of all words over the alphabet $\{E_A, E_B, E_Z, D_A, D_B, D_Z, i_A, i_B, i_Z, d_A, d_B, d_Z, d\}$ that simplify to the empty string using the cancellation rules $D_X E_X = E_X D_X = d_X i_X = d i_X = \epsilon$.
2. This set of words is context free and can be generated by a context free grammar with rules $S \to \epsilon | D_X S E_X S | \ldots$ and so on for all cancellation rules.
   The grammar can be easily converted into an equivalent Push Down Automaton. Notice that the size of this automaton is constant because it does not depend on the protocol.
3. Next we build a nondeterministic finite automaton accepting all the strings of the form $g_k \circ \ldots \circ g_1 \circ f[1]A, B$ where each g_i is one of the finitely many functions the adversary has access to.

Decidability Procedure of Formal Execution Model

4. The resulting automaton has a number of states proportional to n.

Decidability Procedure of Formal Execution Model

4. The resulting automaton has a number of states proportional to n.

5. Finally, we combine the PDA and NFA using a cartesian product construction to obtain a new PDA that accepts the intersection of the two languages.

   At this point we are left with the problem of deciding if the language of a PDA is empty or not.

Decidability Procedure of Formal Execution Model

4. The resulting automaton has a number of states proportional to n.

5. Finally, we combine the PDA and NFA using a cartesian product construction to obtain a new PDA that accepts the intersection of the two languages.

   At this point we are left with the problem of deciding if the language of a PDA is empty or not.

   $\Longrightarrow$ This can be done in $O(n^3)$.

Dolev-Yao Model conclusion

- DY model looks very simplistic: the scope of functions and protocols is very narrow.

Dolev-Yao Model conclusion

- DY model looks very simplistic: the scope of functions and protocols is very narrow.
- DY model, on the other hand, puts no restriction on the strength of the opponent.

Dolev-Yao Model conclusion

- DY model looks very simplistic: the scope of functions and protocols is very narrow.
- DY model, on the other hand, puts no restriction on the strength of the opponent.
- Many extensions have been proposed:
  - Multi party protocols. (much harder)
  - Homomorphic encryptions.
  - Hash functions.
  - Zero-Knowledge proofs.
  - etc.

Plan

1. Introduction
2. Dolev-Yao Model
   - The basic Dolev-Yao Model
   - Soundness Results for Dolev-Yao like Models

Reality Check of DY-Models

- It is not obvious that a proof in a Dolev-Yao model implies security with respect to real cryptographic definitions.

Reality Check of DY-Models

- It is not obvious that a proof in a Dolev-Yao model implies security with respect to real cryptographic definitions.
- We are looking to "soundness" results for DY Models.

Reality Check of DY-Models

- It is not obvious that a proof in a Dolev-Yao model implies security with respect to real cryptographic definitions.
- We are looking to “soundness” results for DY Models.
- Raw question : how does the formal models relate to computational security models ?

Reality Check of DY-Models

- It is not obvious that a proof in a Dolev-Yao model implies security with respect to real cryptographic definitions.
- We are looking to "soundness" results for DY Models.
- Raw question : how does the formal models relate to computational security models ?
- Early results on passive attackers [Abadi and Rogaway, 2000].

Reality Check of DY-Models

- It is not obvious that a proof in a Dolev-Yao model implies security with respect to real cryptographic definitions.
- We are looking to “soundness” results for DY Models.
- Raw question : how does the formal models relate to computational security models ?
- Early results on passive attackers [Abadi and Rogaway, 2000].
- Results on active attackers more complex [Backes et al., 2003].

Reality Check of DY-Models

- It is not obvious that a proof in a Dolev-Yao model implies security with respect to real cryptographic definitions.
- We are looking to “soundness” results for DY Models.
- Raw question : how does the formal models relate to computational security models ?
- Early results on passive attackers [Abadi and Rogaway, 2000].
- Results on active attackers more complex [Backes et al., 2003].
- Impossibility results as well [Backes et al., 2006].

Bibliography I

Abadi, M. and Fournet, C. (2001).
Mobile values, new names, and secure communication.
In *Conference Record of POPL 2001: The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001*, pages 104–115. ACM.

Abadi, M. and Rogaway, P. (2000).
Reconciling two views of cryptography (the computational soundness of formal encryption).
In *Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000, Sendai, Japan, August 17-19, 2000, Proceedings*, pages 3–22.

Backes, M., Pfitzmann, B., and Waidner, M. (2003).
A universally composable cryptographic library.
IACR Cryptology ePrint Archive, 2003:15.

Bibliography II

Backes, M., Pfitzmann, B., and Waidner, M. (2006).
Soundness Limits of Dolev-Yao Models.
In et Steve Kremer, V. C., editor, *Workshop on Formal and Computational Cryptography (FCC 2006)*, Venice/Italy. Véronique Cortier et Steve Kremer.

Backes, M. and Unruh, D. (2010).
Computational soundness of symbolic zero-knowledge proofs.
Journal of Computer Security, 18(6):1077–1155.

Bellare, M. and Rogaway, P. (1993).
Random oracles are practical: A paradigm for designing efficient protocols.
In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM.

Bibliography III

Bohli, J., Müller-Quade, J., and Röhrich, S. (2007).
Bingo voting: Secure and coercion-free voting using a trusted random number generator.
In *E-Voting and Identity, First International Conference, VOTE-ID 2007, Bochum, Germany, October 4-5, 2007, Revised Selected Papers*, volume 4896 of *Lecture Notes in Computer Science*, pages 111–124. Springer.

Boyen, X., Mei, Q., and Waters, B. (2005).
Direct chosen ciphertext security from identity-based techniques.
In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 320–329, New York, NY, USA. ACM.

Bibliography IV

Camenisch, J., Neven, G., and Shelat, A. (2007).
Simulatable adaptive oblivious transfer.
In Naor, M., editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 573–590. Springer.

Canetti, R. and Fischlin, M. (2001).
Universally composable commitments.
In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer.

Bibliography V

Canetti, R., Goldreich, O., and Halevi, S. (1998).
The random oracle methodology, revisited (preliminary version).
In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 209–218. ACM.

Chevallier-Mames, B., Fouque, P., Pointcheval, D., Stern, J., and Traoré, J. (2010).
On some incompatible properties of voting schemes.
In *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 191–199. Springer.

Bibliography VI

Cortier, V. and Smyth, B. (2011).
Attacking and fixing helios: An analysis of ballot secrecy.
In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 297–311. IEEE Computer Society.

Dolev, D. and Yao, A. C. (1983).
On the security of public key protocols.
IEEE Transactions on Information Theory, 29(2):198–207.

Dreier, J., Lafourcade, P., and Lakhnech, Y. (2012).
A formal taxonomy of privacy in voting protocols.
In *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*, pages 6710–6715. IEEE.

Bibliography VII

Fujioka, A., Okamoto, T., and Ohta, K. (1992).
A practical secret voting scheme for large scale elections.
In *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer.

Gennaro, R., Halevi, S., and Rabin, T. (1999).
Secure hash-and-sign signatures without the random oracle.
In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer.

Bibliography VIII

Gentry, C. (2006).
Practical identity-based encryption without random oracles.
In Vaudenay, S., editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer.

Goldwasser, S. and Micali, S. (1984).
Probabilistic encryption.
Journal of Computer and System Sciences, 28(2):270–299.

Bibliography IX

Goldwasser, S., Micali, S., and Rivest, R. L. (1984).
A "paradoxical"' solution to the signature problem (abstract).
In Blakley, G. R. and Chaum, D., editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, page 467. Springer.

Jacquemard, F., Rusinowitch, M., and Vigneron, L. (2000).
Compiling and verifying security protocols.
In *Logic for Programming and Automated Reasoning, 7th International Conference, LPAR 2000, Reunion Island, France, November 11-12, 2000, Proceedings*, volume 1955 of *Lecture Notes in Computer Science*, pages 131–160. Springer.

Bibliography X

Juels, A., Catalano, D., and Jakobsson, M. (2010).
Coercion-resistant electronic elections.
In *Towards Trustworthy Elections, New Directions in Electronic Voting*, volume 6000 of *Lecture Notes in Computer Science*, pages 37–63. Springer.

Koblitz, N. and Menezes, A. J. (2015).
The random oracle model: a twenty-year retrospective.
Des. Codes Cryptography, 77(2-3):587–610.

Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., and Yoo, S. (2003).
Providing receipt-freeness in mixnet-based voting protocols.
In *Information Security and Cryptology - ICISC 2003, 6th International Conference, Seoul, Korea, November 27-28, 2003, Revised Papers*, volume 2971 of *Lecture Notes in Computer Science*, pages 245–258. Springer.

Bibliography XI

Needham, R. M. and Schroeder, M. D. (1978).
Using encryption for authentication in large networks of computers.
Commun. ACM, 21(12):993–999.

Okamoto, T. (1996).
An electronic voting scheme.
In *IFIP World Conference on IT Tools*, pages 21–30.

Pfitzmann, B. and Waidner, M. (2001).
A model for asynchronous reactive systems and its application to secure message transmission.
In *2001 IEEE Symposium on Security and Privacy, Oakland, California, USA May 14-16, 2001*, pages 184–200. IEEE Computer Society.