Proofs of Knowledge

Stéphanie Delaune

November 28, 2006

Proofs of knowledge

Proof of knowledge are often used to

- prove one's identity (*e.g.* authentication protocol)
- prove one's belonging to a group
- prove that one has done something correctly (*e.g.* mix net)

Example

- Alice knows the product of two prime numbers, (*e.g.* $p_1 \times p_2$),
- Alice knows also the pair (p_1, p_2).

Now, assume that Bob knows only the product $p_1 \times p_2$

- He is not able to retieve the pair (p_1, p_2) of Alice
  $\longrightarrow$ factorisation in prime numbers is a very hard problem
- If Alice gives him p_1 and p_2 he is convinced that she knows the result.

Proofs of knowledge

Proof of knowledge are often used to

- prove one's identity (*e.g.* authentication protocol)
- prove one's belonging to a group
- prove that one has done something correctly (*e.g.* mix net)

Example

- Alice knows the product of two prime numbers, (*e.g.* $p_1 \times p_2$),
- Alice knows also the pair (p_1, p_2).

Now, assume that Bob knows only the product $p_1 \times p_2$

- He is not able to retieve the pair (p_1, p_2) of Alice
  $\longrightarrow$ factorisation in prime numbers is a very hard problem
- If Alice gives him p_1 and p_2 he is convinced that she knows the result.

Proofs of knowledge

Proof of knowledge are often used to

- prove one's identity (*e.g.* authentication protocol)
- prove one's belonging to a group
- prove that one has done something correctly (*e.g.* mix net)

Example

- Alice knows the product of two prime numbers, (*e.g.* $p_1 \times p_2$),
- Alice knows also the pair (p_1, p_2).

Now, assume that Bob knows only the product $p_1 \times p_2$

- He is not able to retieve the pair (p_1, p_2) of Alice
  $\longrightarrow$ factorisation in prime numbers is a very hard problem
- If Alice gives him p_1 and p_2 he is convinced that she knows the result.

Two kinds of proofs of knowledge

First Solution: (*e.g.* password mechanism)

- the verifier learns (or even already knows) the password,
- an eavesdropper learns the password

Second Solution: zero-knowledge proof

- an eavesdropper will not learn the solution,
- the verifier will not learn the solution.

Two kinds of proofs of knowledge

First Solution: (*e.g.* password mechanism)

- the verifier learns (or even already knows) the password,
- an eavesdropper learns the password

Second Solution: zero-knowledge proof

- an eavesdropper will not learn the solution,
- the verifier will not learn the solution.

Two kinds of proofs of knowledge

First Solution: (*e.g.* password mechanism)

- the verifier learns (or even already knows) the password,
- an eavesdropper learns the password

Second Solution: zero-knowledge proof

- an eavesdropper will not learn the solution,
- the verifier will not learn the solution.

> *"Zero-knowledge proofs are fascinating and extremely useful constructs. They are both convincing and yet yield nothing beyond the validity of the assertion being proved."* O. Goldreich

Outline of the lecture

1. Introduction
2. Proofs of knowledge
3. Zero-knowledge proofs
4. Conclusion

Outline of the lecture

What is it ?

A proof of knowledge is a method for one party (the prover) to prove to another (the verifier) that he knows some statement.

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.

Example: credit card payment

- The client CI puts his credit card C in the terminal T.
- The merchant enters the amount M of the sale.
- The terminal authenticates the credit card.
- The client enters his PIN.
  If $M \geq$ €100, then in 20% of cases,
  - The terminal contacts the bank B.
  - The banks gives its authorisation.

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\{hash(Data)\}_{\mathrm{priv}(B)}$
- a secret key shared with the bank – K_{CB}

Terminal

- the public key of the bank – $\mathrm{pub}(B)$

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathrm{priv}(B)$
- a public key to verify a signature – $\mathrm{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some $Data$: name of the cardholder, expiry date ...
- a signature of the $Data$ – $\{hash(Data)\}_{\mathrm{priv}(B)}$
- a secret key shared with the bank – K_{CB}

Terminal

- the public key of the bank – $\mathrm{pub}(B)$

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\mathsf{priv}(B)$
- a public key to verify a signature – $\mathsf{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\{hash(Data)\}_{\mathsf{priv}(B)}$
- a secret key shared with the bank – K_{CB}

Terminal

- the public key of the bank – $\mathsf{pub}(B)$

More details

the Bank B , the Client Cl, the Credit Card C and the Terminal T

Bank

- a private signature key – $\text{priv}(B)$
- a public key to verify a signature – $\text{pub}(B)$
- a secret key shared with the credit card – K_{CB}

Credit Card

- some *Data*: name of the cardholder, expiry date ...
- a signature of the *Data* – $\{hash(Data)\}_{\text{priv}(B)}$
- a secret key shared with the bank – K_{CB}

Terminal

- the public key of the bank – $\text{pub}(B)$

Payment protocol

the terminal T reads the credit card C:

$$1. \quad C \rightarrow T : \quad Data, \{hash(Data)\}_{\mathsf{priv}(B)}$$

the terminal T asks the code:

$$\begin{array}{llll} 2. & T \rightarrow & CI : & code? \\ 3. & CI \rightarrow & C : & 1234 \\ 4. & C \rightarrow & T : & ok \end{array}$$

the terminal T requests authorisation the bank B:

$$\begin{array}{llll} 5. & T \rightarrow & B : & auth? \\ 6. & B \rightarrow & T : & 4528965874123 \\ 7. & T \rightarrow & C : & 4528965874123 \\ 8. & C \rightarrow & T : & \{4528965874123\}_{K_{CB}} \\ 9. & T \rightarrow & B : & \{4528965874123\}_{K_{CB}} \\ 10. & B \rightarrow & T : & ok \end{array}$$

Payment protocol

the terminal T reads the credit card C:

$$1. \quad C \rightarrow T : \quad Data, \{hash(Data)\}_{\mathsf{priv}(B)}$$

the terminal T asks the code:

$$\begin{array}{llcll} 2. & T & \rightarrow & CI : & code? \\ 3. & CI & \rightarrow & C : & 1234 \\ 4. & C & \rightarrow & T : & ok \end{array}$$

the terminal T requests authorisation the bank B:

$$\begin{array}{llcll} 5. & T & \rightarrow & B : & auth? \\ 6. & B & \rightarrow & T : & 4528965874123 \\ 7. & T & \rightarrow & C : & 4528965874123 \\ 8. & C & \rightarrow & T : & \{4528965874123\}_{K_{CB}} \\ 9. & T & \rightarrow & B : & \{4528965874123\}_{K_{CB}} \\ 10. & B & \rightarrow & T : & ok \end{array}$$

Payment protocol

the terminal T reads the credit card C:

$$1. \quad C \rightarrow T : \quad Data, \{hash(Data)\}_{\mathrm{priv}(B)}$$

the terminal T asks the code:

$$\begin{array}{llcll} 2. & T & \rightarrow & CI : & code? \\ 3. & CI & \rightarrow & C : & 1234 \\ 4. & C & \rightarrow & T : & ok \end{array}$$

the terminal T requests authorisation the bank B:

$$\begin{array}{llcll} 5. & T & \rightarrow & B : & auth? \\ 6. & B & \rightarrow & T : & 4528965874123 \\ 7. & T & \rightarrow & C : & 4528965874123 \\ 8. & C & \rightarrow & T : & \{4528965874123\}_{K_{CB}} \\ 9. & T & \rightarrow & B : & \{4528965874123\}_{K_{CB}} \\ 10. & B & \rightarrow & T : & ok \end{array}$$

Authentication of the credit card

the terminal T asks the code:

$$
\begin{array}{llcll}
2. & T & \longrightarrow & CI : & code? \\
3. & CI & \longrightarrow & C : & 1234 \\
4. & C & \longrightarrow & T : & ok
\end{array}
$$

Discussion

- the secret code is revealed to the verifer
- the secret code is revealed to any eavesdropper
- the verifier of the proof is the credit card itself
  $\longrightarrow$ Yes Card – (Serge Humpich case)

Authentication of the credit card

the terminal T asks the code:

$$\begin{array}{llcll} 2. & T & \rightarrow & CI: & \mathit{code?} \\ 3. & CI & \rightarrow & C: & 1234 \\ 4. & C & \rightarrow & T: & \mathit{ok} \end{array}$$

Discussion

- the secret code is revealed to the verifer
- the secret code is revealed to any eavesdropper
- the verifier of the proof is the credit card itself
  $\longrightarrow$ Yes Card – (Serge Humpich case)

Requesting authorisation to the bank

the terminal T requests authorisation to the bank B:

$$
\begin{array}{llcll}
5. & T & \rightarrow & B: & \mathit{auth?} \\
6. & B & \rightarrow & T: & 4528965874123 \\
7. & T & \rightarrow & C: & 4528965874123 \\
8. & C & \rightarrow & T: & \{4528965874123\}_{K_{CB}} \\
9. & T & \rightarrow & B: & \{4528965874123\}_{K_{CB}} \\
10. & B & \rightarrow & T: & \mathit{ok}
\end{array}
$$

Discussion

- the secret code is already known by the verifier
- challenge mecanism: the prover C proves to the verifier B that he knows the secret key K_{CB}
- an eavesdropper does not learn the secret K_{CB} but he learns something about it

Requesting authorisation to the bank

the terminal T requests authorisation to the bank B:

$$
\begin{array}{llcll}
5. & T & \rightarrow & B: & auth? \\
6. & B & \rightarrow & T: & 4528965874123 \\
7. & T & \rightarrow & C: & 4528965874123 \\
8. & C & \rightarrow & T: & \{4528965874123\}_{K_{CB}} \\
9. & T & \rightarrow & B: & \{4528965874123\}_{K_{CB}} \\
10. & B & \rightarrow & T: & ok
\end{array}
$$

Discussion

- the secret code is already known by the verifier
- challenge mecanism: the prover C proves to the verifier B that he knows the secret key K_{CB}
- an eavesdropper does not learn the secret K_{CB} but he learns something about it

Outline of the talk

1. Introduction

2. Proofs of knowledge

3. Zero-knowledge proofs

4. Conclusion

What is it ?

Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

$\longrightarrow$ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.
- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

The definitions given above seem to be contradictory.

$\longrightarrow$ Does zero-knowledge proofs really exist?

What is it ?

Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

⟶ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.
- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

The definitions given above seem to be contradictory.

⟶ Does zero-knowledge proofs really exist?

What is it ?

Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

⟶ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.
- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

The definitions given above seem to be contradictory.

⟶ Does zero-knowledge proofs really exist?

What is it ?

Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proved.

$\longrightarrow$ introduced 20 years ago by Goldwasser, Micali and Rackoff [1985]

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.
- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.
- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

The definitions given above seem to be contradictory.

$\longrightarrow$ Does zero-knowledge proofs really exist?

Applications

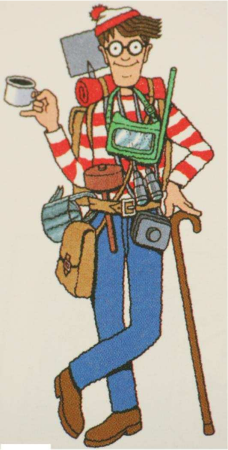
Authentication properties

- Credit card payment
  $\longrightarrow$ to prove that you know the secret code without revealing it
- prove your identity
- prove that you belongs to a group without revealing who you are
  $\longrightarrow$ to ensure privacy

Other properties

- to enforce honest behavior
  $\longrightarrow$ *e.g.* mix net in electronic voting protocols,
- ....

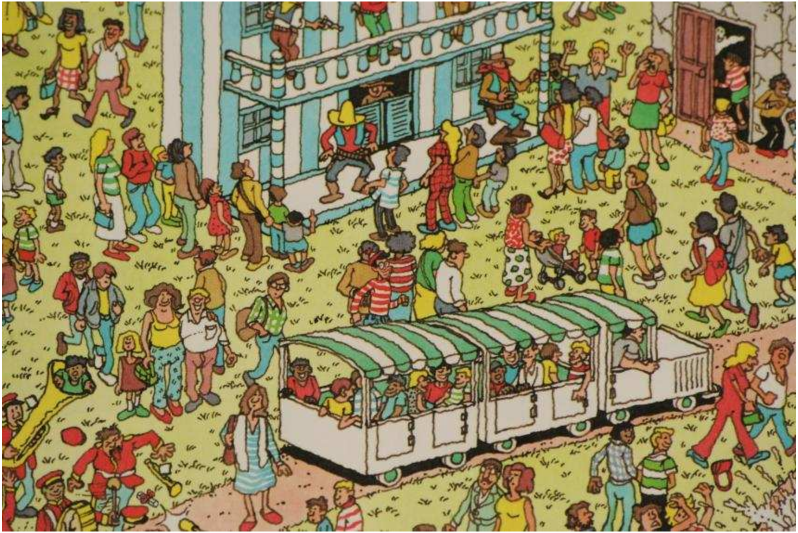
Example: Where is Charlie?



Goal:

1. find the reporter Charlie in a big picture,
2. convince the verifier (me) that you have the solution without revealing it (neither to me, nor to the others).

Example: Where is Charlie?



Example: Where is Charlie?

How can you prove that you know where is Charlie without saying nothing about where he is?

Solutions:

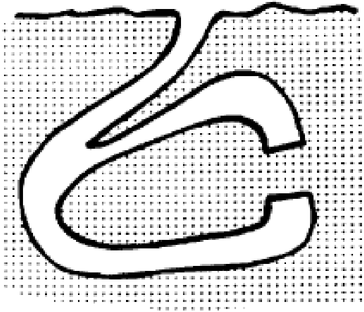
Example: Where is Charlie?

How can you prove that you know where is Charlie without saying nothing about where he is?

Solutions:

1. get a copy of the picture, cut out Charlie and show it to me.
2. put a big mask with a window having the shape of Charlie and show me Charlie through the window.

Example: The strange cave of Ali Baba



- a cave shaped like a circle, with entrance on one side and the magic door blocking the opposite side
- the door can be opened by saying some magic words "....".

Goal:

Ali Baba wants to convince me that he knows the secret without revealing it.

How can Ali Baba proceed?

Example: The strange cave of Ali Baba

Ali Baba wants to convince me that he knows the magic words.

Ali Baba hides inside the cave

I ask him to exit on the right side or on the left side
$\longrightarrow$ I choose

Ali Baba exits from the side I just asked.

... and we repeat this procedure several times.

Example: The strange cave of Ali Baba

Ali Baba wants to convince me that he knows the magic words.

Ali Baba hides inside the cave

I ask him to exit on the right side or on the left side
$\longrightarrow$ I choose

Ali Baba exits from the side I just asked.

... and we repeat this procedure several times.

Example: The strange cave of Ali Baba

I can be convinced that Ali Baba knows the magic words.

Why?

- If Ali Baba does not know the magic word, then he can only return by the same path. Since, I randomly choose the path, he has 50% chance of guessing correctly.
- By repeating this trick many times, say 20 times, his chance of succesfully anticipating all my requests becomes very small.

Moreover,

- I learn nothing about the magic word beyond the fact this word allows Ali Baba to open the magic door, and
- I am not able to prove to someone else that I know the magic words.

Example: The strange cave of Ali Baba

I can be convinced that Ali Baba knows the magic words.

Why?

- If Ali Baba does not know the magic word, then he can only return by the same path. Since, I randomly choose the path, he has 50% chance of guessing correctly.
- By repeating this trick many times, say 20 times, his chance of succesfully anticipating all my requests becomes very small.

Moreover,

- I learn nothing about the magic word beyond the fact this word allows Ali Baba to open the magic door, and
- I am not able to prove to someone else that I know the magic words.

Example: The strange cave of Ali Baba

The End
– of Ali Baba story –

To know the magic word:

How to explain Zero-Knowledge Protocols to Your Children.
Jean-Jacques Quisquater and Louis Guillou.

Graph 3-coloring

Definition (3-coloring)

A 3-coloring of a graph is an assignment of colors in $\{\bullet, \bullet, \bullet\}$ to vertices such that no pair of adjacent vertices are assigned to the same color.

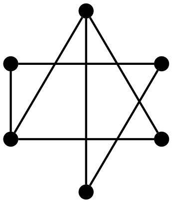
Example

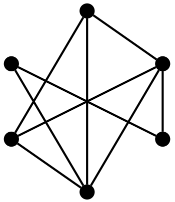
Graph 3-coloring

Definition (3-coloring)

A 3-coloring of a graph is an assignment of colors in $\{\bullet, \bullet, \bullet\}$ to vertices such that no pair of adjacent vertices are assigned to the same color.

Example





Graph 3-coloring

Definition (3-coloring)

A 3-coloring of a graph is an assignment of colors in $\{\bullet, \bullet, \bullet\}$ to vertices such that no pair of adjacent vertices are assigned to the same color.

Example

Graph 3-coloring

Definition (3-coloring)

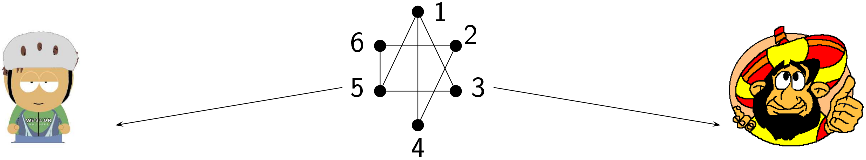
A 3-coloring of a graph is an assignment of colors in {•, •, •} to vertices such that no pair of adjacent vertices are assigned to the same color.

Example

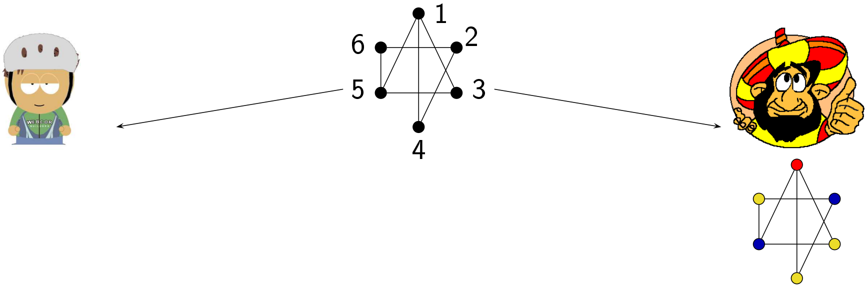
3-coloring problem

Given graph G, the problems of deciding if the graph G is 3-colorable is a very hard problem. It is also very hard to find a 3-coloring of a large graph.

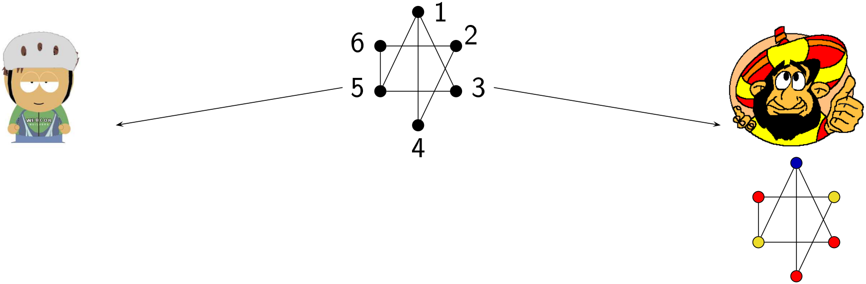
Protocol based on the 3-coloring problem

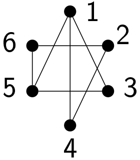


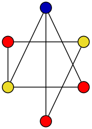
Protocol based on the 3-coloring problem



Protocol based on the 3-coloring problem

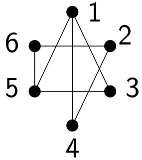


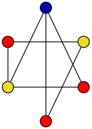




$\{\bullet\}_{k_1}, \{\bullet\}_{k_2}, \{\bullet\}_{k_3}, \{\bullet\}_{k_4}, \{\bullet\}_{k_5}, \{\bullet\}_{k_6}$

Protocol based on the 3-coloring problem



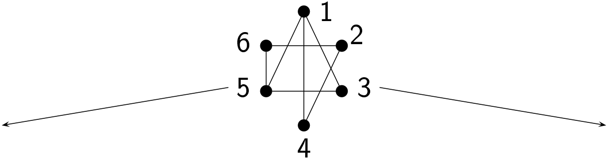


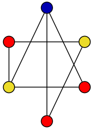
$\{\bullet\}_{k_1}, \{\bullet\}_{k_2}, \{\bullet\}_{k_3}, \{\bullet\}_{k_4}, \{\bullet\}_{k_5}, \{\bullet\}_{k_6}$

choose an edge

$(1, 4)$

Protocol based on the 3-coloring problem





$\{\bullet\}_{k_1}, \{\bullet\}_{k_2}, \{\bullet\}_{k_3}, \{\bullet\}_{k_4}, \{\bullet\}_{k_5}, \{\bullet\}_{k_6}$ ←

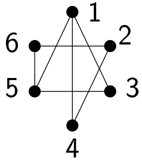
choose an edge

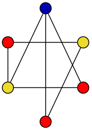
$(1, 4)$ →

send keys

k_1 and k_4 ←

Protocol based on the 3-coloring problem





$\{\bullet\}_{k_1}, \{\bullet\}_{k_2}, \{\bullet\}_{k_3}, \{\bullet\}_{k_4}, \{\bullet\}_{k_5}, \{\bullet\}_{k_6}$

choose an edge

$(1, 4)$

send keys

k_1 and k_4

decrypt $\{\bullet\}_{k_1}$ with k_1
decrypt $\{\bullet\}_{k_4}$ with k_4

accept since $\bullet \neq \bullet$

... repeat the procedure several times

Discussion on the protocol

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

  $\longrightarrow$ if Ali Baba knows the 3-coloring of the graph, then the verifier will accept his proof.

- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.

  $\longrightarrow$ if Ali Baba does not know a 3-coloring of the graph, then Bob rejects with probability $\frac{1}{\#edges}$.

- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

  $\longrightarrow$ Bob just sees two random colors. Hence, he learns nothing about the 3-coloring of the graph.

Discussion on the protocol

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

  $\longrightarrow$ if Ali Baba knows the 3-coloring of the graph, then the verifier will accept his proof.

- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.

  $\longrightarrow$ if Ali Baba does not know a 3-coloring of the graph, then Bob rejects with probability $\frac{1}{\#edges}$.

- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

  $\longrightarrow$ Bob just sees two random colors. Hence, he learns nothing about the 3-coloring of the graph.

Discussion on the protocol

- Completeness: if the statement is true, the honest verifier will be convinced of this fact by an honest prover.

  $\longrightarrow$ if Ali Baba knows the 3-coloring of the graph, then the verifier will accept his proof.

- Soundness: if the statement is false, no cheating prover can convince the honest verifier that it is true.

  $\longrightarrow$ if Ali Baba does not know a 3-coloring of the graph, then Bob rejects with probability $\frac{1}{\#edges}$.

- Zero-knowledge: If the statement is true, no cheating verifier learns anything other than this fact.

  $\longrightarrow$ Bob just sees two random colors. Hence, he learns nothing about the 3-coloring of the graph.

Composability

This allows us to use the same protocol several times.

Sequential composition

Each invocation follows the termination of the previous one.

$\longrightarrow$ Generally, sequential composition is safe. Note that otherwise, the applicability of the protocol is highly limited.

Parallel composition

Many instances of the protocol are invoked at the same time and proceed at the same pace (synchronous model of communication)

$\longrightarrow$ Generally, parallel composition is not safe.

Concurrent composition

This generalizes both sequential and parallel composition. Many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace.

Composability

This allows us to use the same protocol several times.

Sequential composition

Each invocation follows the termination of the previous one.

$\longrightarrow$ Generally, sequential composition is safe. Note that otherwise, the applicability of the protocol is highly limited.

Parallel composition

Many instances of the protocol are invoked at the same time and proceed at the same pace (synchronous model of communication)

$\longrightarrow$ Generally, parallel composition is not safe.

Concurrent composition

This generalizes both sequential and parallel composition. Many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace.

Composability

This allows us to use the same protocol several times.

Sequential composition

Each invocation follows the termination of the previous one.

$\longrightarrow$ Generally, sequential composition is safe. Note that otherwise, the applicability of the protocol is highly limited.

Parallel composition

Many instances of the protocol are invoked at the same time and proceed at the same pace (synchronous model of communication)

$\longrightarrow$ Generally, parallel composition is not safe.

Concurrent composition

This generalizes both sequential and parallel composition. Many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace.

Composability

This allows us to use the same protocol several times.

Sequential composition

Each invocation follows the termination of the previous one.

$\longrightarrow$ Generally, sequential composition is safe. Note that otherwise, the applicability of the protocol is highly limited.

Parallel composition

Many instances of the protocol are invoked at the same time and proceed at the same pace (synchronous model of communication)

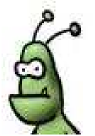
$\longrightarrow$ Generally, parallel composition is not safe.

Concurrent composition

This generalizes both sequential and parallel composition. Many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace.

Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.

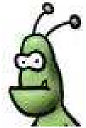


$\xrightarrow{\text{I know the secret}}$

How can he do this?

Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.



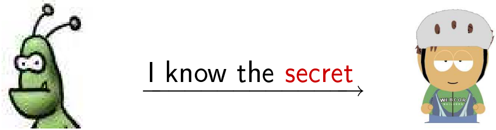
I know the secret $\longrightarrow$

How can he do this?

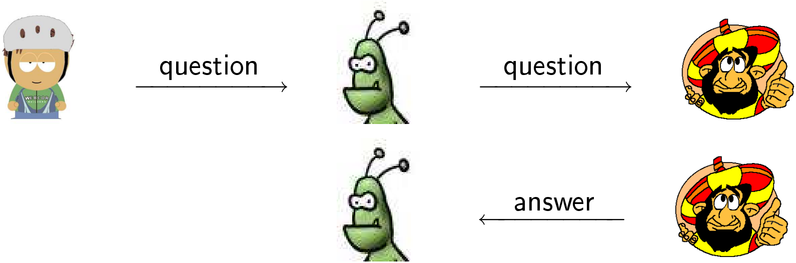
question $\longrightarrow$

Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.

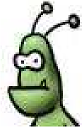


How can he do this?



Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.



I know the secret →

How can he do this?

question →

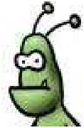
question →

← answer

← answer

Parallel composition – Man in the middle attack

The intruder wants to convince Bob that he knows the secret.



I know the secret $\longrightarrow$

How can he do this?

question $\longrightarrow$

question $\longrightarrow$

$\longleftarrow$ answer

$\longleftarrow$ answer

$\longrightarrow$ This kind of attack often succeeds on Zero-knowledge protocols.

Conclusion and Further Reading

Zero-knowledge proofs are fascinating due to their seemingly contradictory definitions. Nevertheless, such kind of proofs really exist.

It turns out that in an Internet-like setting, where multiple protocols may be executed concurrently, building zero-knowledge proofs is more challenging.

Bibiliography

1. *How to explain Zero-Knowledge Protocols to Your Children.* Jean-Jacques Quisquater and Louis Guillou.
2. Wikipedia web site `http://en.wikipedia.org/wiki/Zero-knowledge_proof`
3. *Zero-Knowledge twenty years after its invention.* Oded Goldreich.

Conclusion and Further Reading

Zero-knowledge proofs are fascinating due to their seemingly contradictory definitions. Nevertheless, such kind of proofs really exist.

It turns out that in an Internet-like setting, where multiple protocols may be executed concurrently, building zero-knowledge proofs is more challenging.

Bibiliography

1. *How to explain Zero-Knowledge Protocols to Your Children.* Jean-Jacques Quisquater and Louis Guillou.
2. Wikipedia web site `http://en.wikipedia.org/wiki/Zero-knowledge_proof`
3. *Zero-Knowledge twenty years after its invention.* Oded Goldreich.