Zero-Knowledge Proofs

Joost van Amersfoort

University of Amsterdam
Teacher: Christian Schaffner
TA: Malvin Gattinger

October 22, 2014

Introduction

- Zero-knowledge proofs are proofs that yield nothing beyond the validity of the assertion

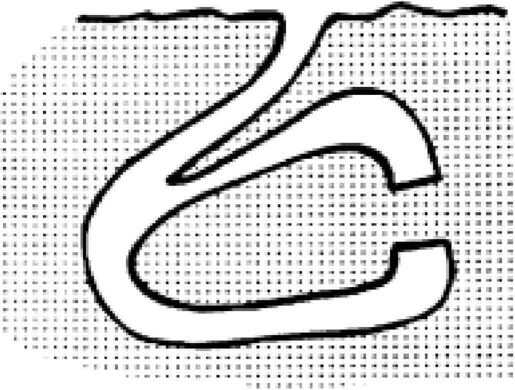


Figure: The special cave [4]

Interactive proofs

- Zero-knowledge proofs are a special case of interactive proofs
- Interactive proofs have two parties: the Prover (P) and the Verifier (V)
- Verifier is a PPT machine, Prover is unbounded and both are able to communicate
- The prover claims a certain statement is true
- If (P,V) accept this statement (completeness) and rejects false statements (soundness), then it is an interactive proof system

Formal Definition Zero-Knowledge Proofs

Fix an interactive machine (the Prover) look at what can be computed by an arbitrary adversary (the Verifier). Now an interactive proof A is zero-knowledge on the set S, if for every feasible strategy B^*, there exists a feasible computation C^*, s.t. the following two probability ensembles are computationally indistinguishable:

Formal Definition Zero-Knowledge Proofs

Fix an interactive machine (the Prover) look at what can be computed by an arbitrary adversary (the Verifier). Now an interactive proof A is zero-knowledge on the set S, if for every feasible strategy B^*, there exists a feasible computation C^*, s.t. the following two probability ensembles are computationally indistinguishable:

- $\{(A, B^*)(x)\}_{x \in S} \stackrel{def}{=}$ the output of B^* after interacting with A on common input $x \in S$; and
- $\{(C^*)(x)\}_{x \in S} \stackrel{def}{=}$ the output of C^* on input $x \in S$

Formal Definition Zero-Knowledge Proofs

Fix an interactive machine (the Prover) look at what can be computed by an arbitrary adversary (the Verifier). Now an interactive proof A is zero-knowledge on the set S, if for every feasible strategy B^*, there exists a feasible computation C^*, s.t. the following two probability ensembles are computationally indistinguishable:

- $\{(A, B^*)(x)\}_{x \in S} \stackrel{def}{=}$ the output of B^* after interacting with A on common input $x \in S$; and
- $\{(C^*)(x)\}_{x \in S} \stackrel{def}{=}$ the output of C^* on input $x \in S$

The first ensemble is the execution of an interactive protocol, the second represents a stand-alone procedure. This means that anything that could be extracted from A, was also already in C. So nothing was gained from the interaction. [2]

Popquiz

- Imagine a scheme where a prover (P) wants to prove to be the owner of a public/private key pair to a verifier (V).

Popquiz

- Imagine a scheme where a prover (P) wants to prove to be the owner of a public/private key pair to a verifier (V).
- V chooses a random message M, encrypts it using the public key and sends the resulting ciphertext to P. P decrypts this message and sends the result M' back.

Popquiz

- Imagine a scheme where a prover (P) wants to prove to be the owner of a public/private key pair to a verifier (V).
- V chooses a random message M, encrypts it using the public key and sends the resulting ciphertext to P. P decrypts this message and sends the result M' back.
- If $M = M'$ then V accepts P's proof.

Popquiz

- Imagine a scheme where a prover (P) wants to prove to be the owner of a public/private key pair to a verifier (V).
- V chooses a random message M, encrypts it using the public key and sends the resulting ciphertext to P. P decrypts this message and sends the result M' back.
- If $M = M'$ then V accepts P's proof.
- Whats could go wrong in this scheme?

Commitment Schemes

- In order to solve the problem of a misbehaving verifier, it is necessary to introduce commitment schemes

Commitment Schemes

- In order to solve the problem of a misbehaving verifier, it is necessary to introduce commitment schemes
- In a commitment scheme, a player is able to choose a value from some set and commit to his choice such that he can no longer change his mind

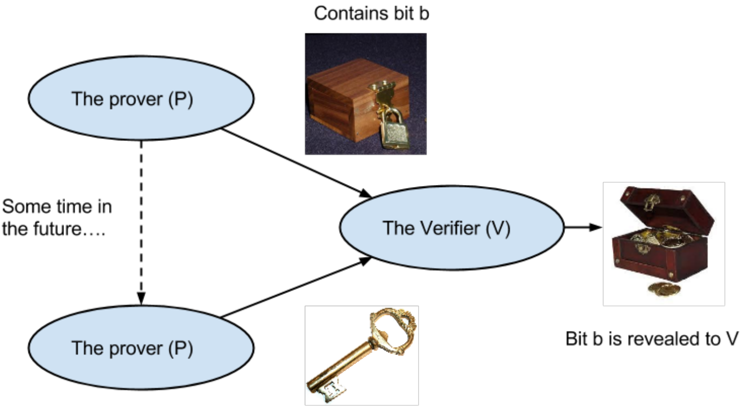
Commitment Schemes

- In order to solve the problem of a misbehaving verifier, it is necessary to introduce commitment schemes
- In a commitment scheme, a player is able to choose a value from some set and commit to his choice such that he can no longer change his mind
- Example: a game with two players P and V, where P wants to commit to a bit b. He writes b down on a piece of paper, puts it in a box and locks it using a padlock. He then passes the box to V

Commitment Schemes

- In order to solve the problem of a misbehaving verifier, it is necessary to introduce commitment schemes
- In a commitment scheme, a player is able to choose a value from some set and commit to his choice such that he can no longer change his mind
- Example: a game with two players P and V, where P wants to commit to a bit b. He writes b down on a piece of paper, puts it in a box and locks it using a padlock. He then passes the box to V
- Whenever P decides to he passes the key to V to open the padlock. In this way P is bound to his original choice and his choice is hidden until he decided to give the key [1]

Commitment Scheme visualized



The Solution

- Remember in the old scheme P just decrypts C and sends M' back

The Solution

- Remember in the old scheme P just decrypts C and sends M' back
- New scheme: instead of sending back M', P sends a commitment message with M'.

The Solution

- Remember in the old scheme P just decrypts C and sends M' back
- New scheme: instead of sending back M', P sends a commitment message with M'.
- He then receives the original message M (forcing the verifier to know M). If M = M', he opens the commitment by sending the key to the V.

The Solution

- Remember in the old scheme P just decrypts C and sends M' back
- New scheme: instead of sending back M', P sends a commitment message with M'.
- He then receives the original message M (forcing the verifier to know M). If M = M', he opens the commitment by sending the key to the V.
- Now the verifier accepts the identity of the prover iff the commitment can be correctly opened and M' = M.

Theoretical Applications

- In the last example the Verifier was forced to behave according to protocol

Theoretical Applications

- In the last example the Verifier was forced to behave according to protocol
- It has been shown that using zero-knowledge protocols as sub-protocols it is possible to transform any protocol that assumes players follow the rules into one that is secure even if players deviate from the protocol [3]

Practical Applications

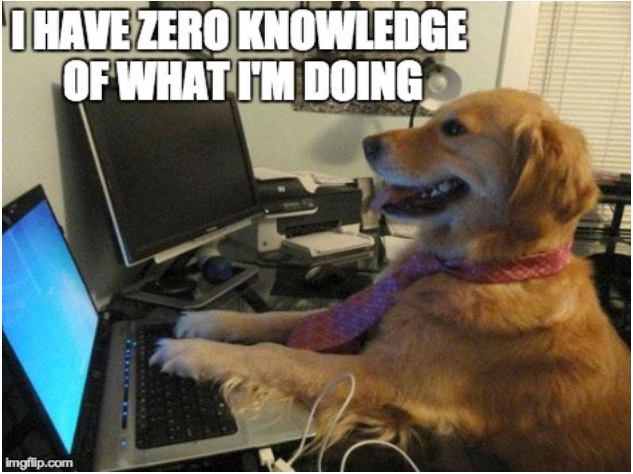
- Biggest impact of zero-knowledge is in design of efficient protocols for specific problems

Practical Applications

- Biggest impact of zero-knowledge is in design of efficient protocols for specific problems
- Example: give the user the solution to a hard problem and the user identifies himself by providing a zero-knowledge proof that he knows this solution

Practical Applications

- Biggest impact of zero-knowledge is in design of efficient protocols for specific problems
- Example: give the user the solution to a hard problem and the user identifies himself by providing a zero-knowledge proof that he knows this solution
- This works on smartcards (OV-Chipkaart) [5], where computation is very limited



Thank you for your attention!

Time left for a game?
Sudoku Zero-Knowledge

Bibliography

Ivan Damgård, *Commitment schemes and zero-knowledge protocols*, Lectures on Data Security, Springer, 1999, pp. 63–86.

Oded Goldreich, *Zero-knowledge twenty years after its invention.*, IACR Cryptology ePrint Archive **2002** (2002), 186.

Oded Goldreich, Silvio Micali, and Avi Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, FOCS, vol. 86, 1986, pp. 174–187.

Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou, *How to explain zero-knowledge protocols to your children*, Advances in Cryptology CRYPTOâĂŹ89 Proceedings, Springer, 1990, pp. 628–631.

Claus-Peter Schnorr, *Efficient signature generation by smart cards*, Journal of cryptology **4** (1991), no. 3, 161–174.