Zero-Knowledge and Sigma Protocol

LIU Yi

Zero knowledge

Zero knowledge

- I know that *Statement* is true, and I want to convince you of that. I try to present all the facts I know and the inferences from the facts that imply *Statement* is true.

Zero knowledge

- I know that *Statement* is true, and I want to convince you of that. I try to present all the facts I know and the inferences from the facts that imply *Statement* is true.

- **Example**: P : 26781 is not a prime since $26781 = 113 \times 237$.

Zero knowledge

- I know that *Statement* is true, and I want to convince you of that. I try to present all the facts I know and the inferences from the facts that imply *Statement* is true.

- **Example**: P : 26781 is not a prime since $26781 = 113 \times 237$.

- Given this factorization, other than that you are convinced that *Statement* is true, you gained some knowledge (the factorization).

Zero knowledge

- Informally, in a Zero Knowledge Proof, Alice will prove to Bob that a statement S is true. Bob will be completely convinced that S is true, but will not learn anything as a result of this process. That is, Bob will gain zero knowledge.

Zero knowledge

- S. Goldwasser, S. Micali, C. Rackoff, STOC' 85

The Knowledge Complexity of Interactive Proof-Systems

(Extended Abstract)

Shafi Goldwasser, MIT
Silvio Micali, MIT
Charles Rackoff, University of Toronto

Zero knowledge

- S. Goldwasser, S. Micali, C. Rackoff, STOC'85

The Knowledge Complexity of Interactive Proof-Systems

(Extended Abstract)

Shafi Goldwasser — MIT
Silvio Micali — MIT
Charles Rackoff — University of Toronto

Shafi, with Micali (and later Rackoff) [6], had been thinking for a while about expanding the traditional notion of "proof" to an interactive process in which a "prover" can convince a probabilistic "verifier" of the correctness of a mathematical proposition with overwhelming probability if and only if the proposition is correct. They called this interactive process an "interactive proof" (a name suggested by Mike Sipser). They wondered if one could prove some non-trivial statement (for example, membership of a string in a hard language) without giving away any knowledge whatsoever about why it was true. They defined that the verifier receives no knowledge from the prover if the verifier could simulate on his own the probability distribution that he obtains in interacting with the prover.The idea that "no knowledge" means simulatability was a very important contribution. They also gave the first example of these "zero knowledge interactive proofs" using quadratic residuosity. This paper won the first ACM SIGACT Gödel Prize. This zero-knowledge work led to a huge research program in the community that continues to this day, including results showing that (subject to an assumption such as the existence of one-way functions) a group of distrusting parties can compute a function of all their inputs without learning any knowledge about other people's inputs beyond that which follows from the value of the function.

Zero knowledge

- S. Goldwasser, S. Micali, C. Rackoff, STOC' 85

The Knowledge Complexity of Interactive Proof-Systems

(Extended Abstract)

Shafi Goldwasser	Silvio Micali	Charles Rackoff
MIT	MIT	University of Toronto

Shafi, with Micali (and later Rackoff) [6], had been thinking for a while about expanding the traditional notion of "proof" to an interactive process in which a "prover" can convince a probabilistic "verifier" of the correctness of a mathematical proposition with overwhelming probability if and only if the proposition is correct. They called this interactive process an "interactive proof" (a name suggested by Mike Sipser). They wondered if one could prove some non-trivial statement (for example, membership of a string in a hard language) without giving away any knowledge whatsoever about why it was true. They defined that the verifier receives no knowledge from the prover if the verifier could simulate on his own the probability distribution that he obtains in interacting with the prover.The idea that "no knowledge" means simulatability was a very important contribution. They also gave the first example of these "zero knowledge interactive proofs" using quadratic residuosity. This paper won the first ACM SIGACT Gödel Prize. This zero-knowledge work led to a huge research program in the community that continues to this day, including results showing that (subject to an assumption such as the existence of one-way functions) a group of distrusting parties can compute a function of all their inputs without learning any knowledge about other people's inputs beyond that which follows from the value of the function.

https://amturing.acm.org/award_winners/goldwasser_8627889.cfm

Zero knowledge

- Together with a paper by László Babai and Shlomo Moran, this landmark paper invented interactive proof systems, for which all five authors won the first Gödel Prize in 1993.

Recipients [edit]

Year	Name(s)	Notes	Publication year
1993	László Babai, Shafi Goldwasser, Silvio Micali, Shlomo Moran, and Charles Rackoff	for the development of interactive proof systems	1988,[paper 1] 1989[paper 2]
1994	Johan Håstad	for an exponential lower bound on the size of constant-depth Boolean circuits (for the parity function).	1989[paper 3]
1995	Neil Immerman and Róbert Szelepcsényi	for the Immerman–Szelepcsényi theorem regarding nondeterministic space complexity	1988,[paper 4] 1988[paper 5]
1996	Mark Jerrum and Alistair Sinclair	for work on Markov chains and the approximation of the permanent of a matrix	1989,[paper 6] 1989[paper 7]
1997	Joseph Halpern and Yoram Moses	for defining a formal notion of "knowledge" in distributed environments	1990[paper 8]
1998	Seinosuke Toda	for Toda's theorem which showed a connection between counting solutions (PP) and alternation of quantifiers (PH)	1991[paper 9]

Applications of ZKPs

- **Protocol design.** A protocol is an algorithm for interactive parties to achieve a certain goal.

Applications of ZKPs

- **Protocol design.** A protocol is an algorithm for interactive parties to achieve a certain goal.

- However, in crypto, we often want to design protocols that should achieve security even when one of the parties is "cheating" (dishonest).

Applications of ZKPs

- **Protocol design.** A protocol is an algorithm for interactive parties to achieve a certain goal.

- However, in crypto, we often want to design protocols that should achieve security even when one of the parties is "cheating" (dishonest). Alice can prove in zero knowledge that she followed the instructions.

Applications of ZKPs

- **Protocol design.** A protocol is an algorithm for interactive parties to achieve a certain goal.

- However, in crypto, we often want to design protocols that should achieve security even when one of the parties is “cheating” (dishonest). Alice can prove in zero knowledge that she followed the instructions.

Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design

(Extended Abstract)

Oded Goldreich
Dept. of Computer Sc.
Technion
Haifa, Israel

Silvio Micali
Lab. for Computer Sc.
MIT
Cambridge, MA 02139

Avi Wigderson
Inst. of Math. and CS
Hebrew University
Jerusalem, Israel

Applications of ZKPs

- **Authentication systems**
- **Blockchains**
- **…**

Zero knowledge Proof

- Prover *P*, verifier *V*, language *L*

Zero knowledge Proof

- Prover P, verifier V, language L
- P proves that $x \in L$ without revealing anything

Zero knowledge Proof

- Prover *P*, verifier *V*, language *L*
- *P* proves that $x \in L$ without revealing anything
  - **Completeness**: *V* always accepts when honest *P* and *V* interact.

Zero knowledge Proof

- Prover *P*, verifier *V*, language *L*
- *P* proves that $x \in L$ without revealing anything
  - **Completeness**: *V* always accepts when honest *P* and *V* interact.
  - **Soundness**: *V* accepts with negligible probability when *x not in L*, for any *P**

Zero knowledge Proof

- Prover *P*, verifier *V*, language *L*
- *P* proves that $x \in L$ without revealing anything
  - **Completeness**: *V* always accepts when honest *P* and *V* interact.
  - **Soundness**: *V* accepts with negligible probability when *x not in L*, for any *P**
    - *Computational soundness: only holds when P* is polynomial-time*

Zero knowledge Proof

- Prover *P*, verifier *V*, language *L*
- *P* proves that $x \in L$ without revealing anything
  - **Completeness**: *V* always accepts when honest *P* and *V* interact.
  - **Soundness**: *V* accepts with negligible probability when *x not in L*, for any *P**
    - *Computational soundness: only holds when P* is polynomial-time*

Zero-knowledge: There exists a simulator S such that S(x) is indistinguishable from a real proof execution

Zero knowledge Proof

- Prover *P*, verifier *V*, language *L*
- *P* proves that $x \in L$ without revealing anything
  - **Completeness**: *V* always accepts when honest *P* and *V* interact.
  - **Soundness**: *V* accepts with negligible probability when *x not in L*, for any *P**
    - *Computational soundness: only holds when P* is polynomial-time*

Zero-knowledge: There exists a simulator S such that S(x) is indistinguishable from a real proof execution

The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero-knowledge.

Zero knowledge Proof

- If proving a statement requires that the prover possess some secret information, then the verifier will not be able to prove the statement to anyone else without possessing the secret information.

Zero knowledge Proof

- If proving a statement requires that the prover possess some secret information, then the verifier will not be able to prove the statement to anyone else without possessing the secret information.

- Informally, a zero-knowledge proof of knowledge is a special case when the statement consists only of the fact that the prover possesses the secret information.

ZK Proof of Knowledge

- Prover P, verifier V, relation R

ZK Proof of Knowledge

- Prover *P*, verifier *V*, relation *R*
- *P* proves that it knows a witness w for which $(x,w) \in R$ without revealing anything.

ZK Proof of Knowledge

- Prover *P*, verifier *V*, relation *R*
- *P* proves that it knows a witness w for which $(x,w) \in R$ without revealing anything.
  - The proof is zero knowledge as before

ZK Proof of Knowledge

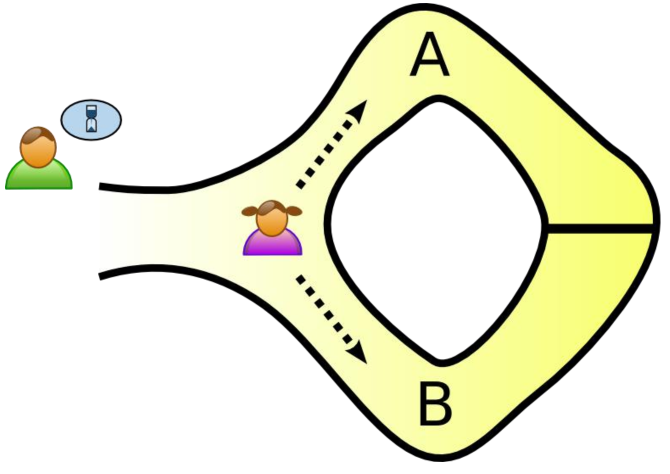
- Prover *P*, verifier *V*, relation *R*
- *P* proves that it knows a witness w for which (x,w) $\in$ R without revealing anything.
  - The proof is zero knowledge as before
  - There exists an extractor *K* that obtains w such that (x,w) $\in$ R from any *P** with the same probability that *P** convinces *V*

ZK Proof of Knowledge

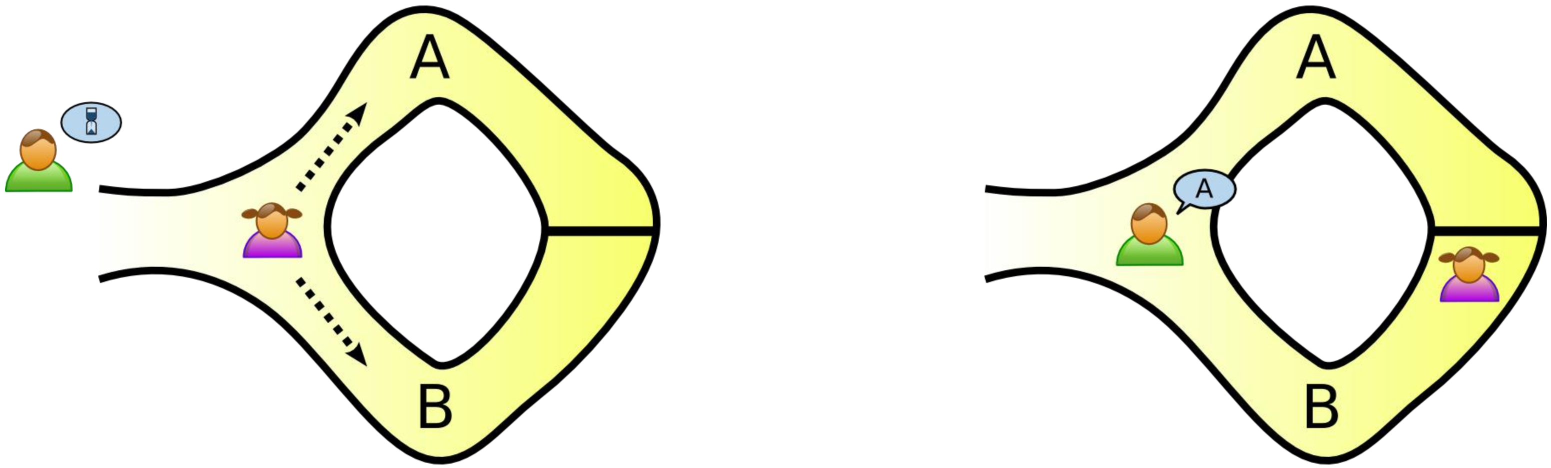
- Prover *P*, verifier *V*, relation *R*
- *P* proves that it knows a witness w for which $(x,w) \in R$ without revealing anything.
  - The proof is zero knowledge as before
  - There exists an extractor *K* that obtains w such that $(x,w) \in R$ from any *P** with the same probability that *P** convinces *V*

- Equivalently: The protocol securely computes the functionality $f_{zk}((x,w),x) = (-,R(x,w))$

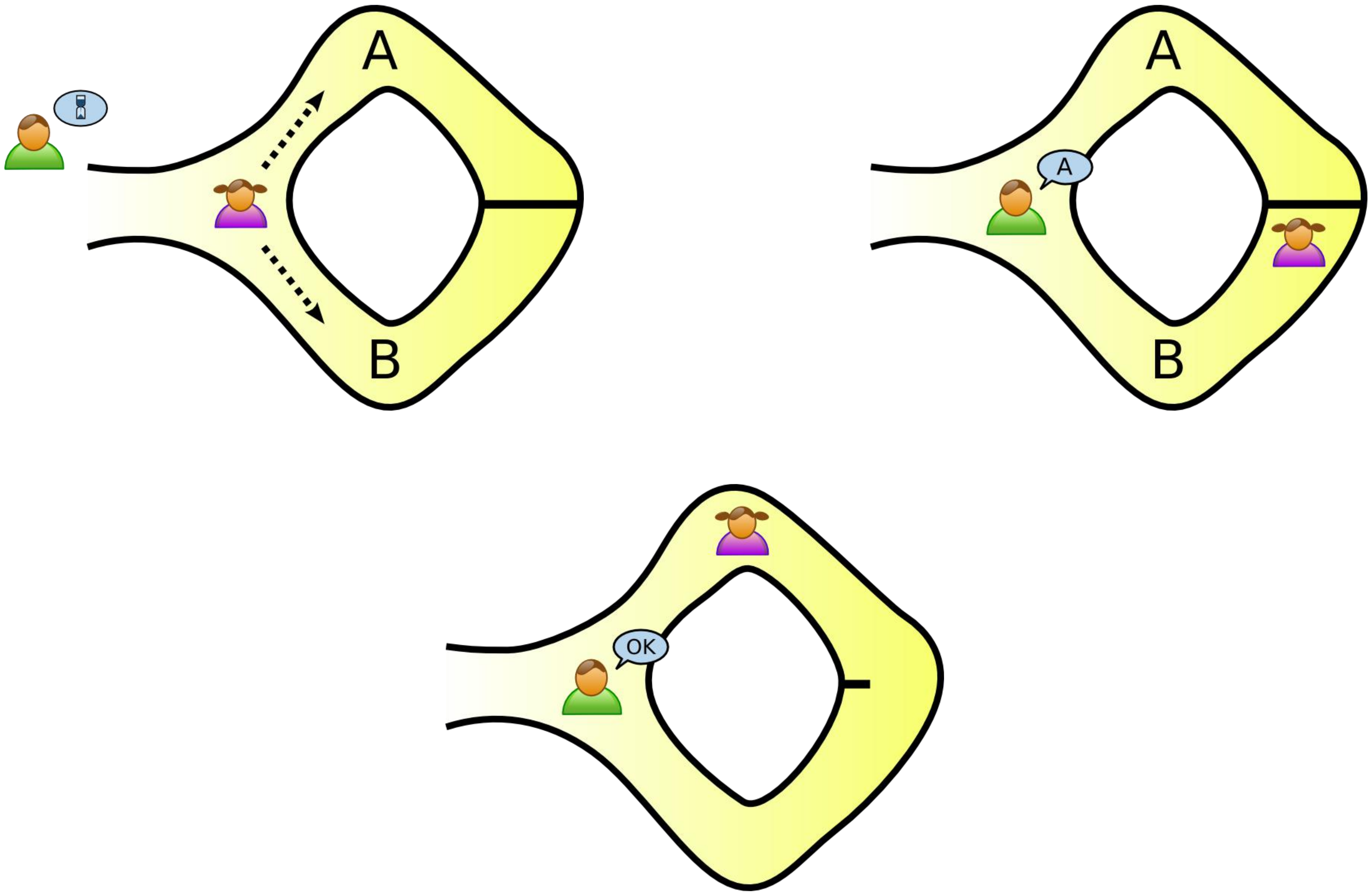
Example: The Ali Baba cave



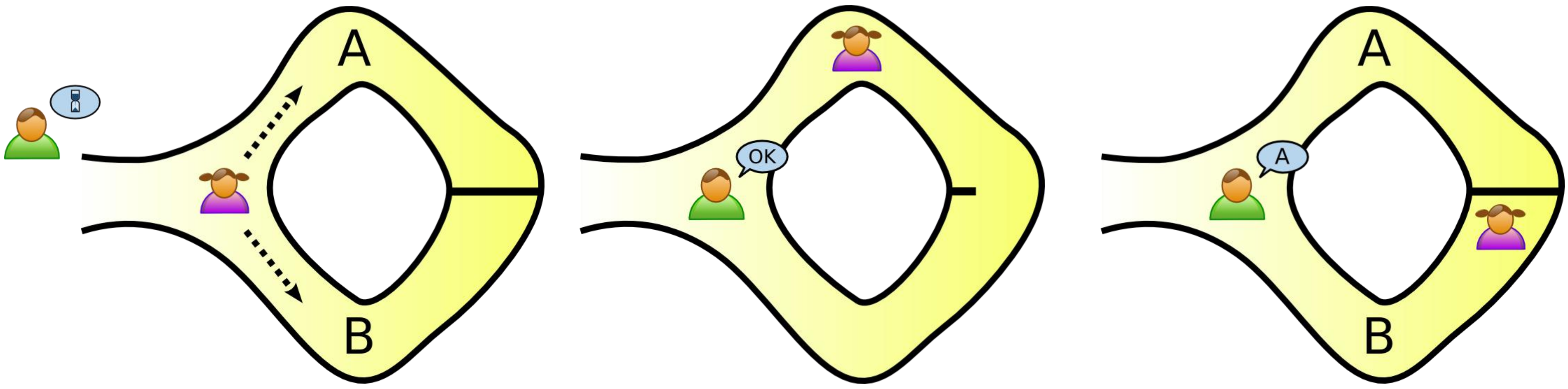
Example: The Ali Baba cave



Example: The Ali Baba cave

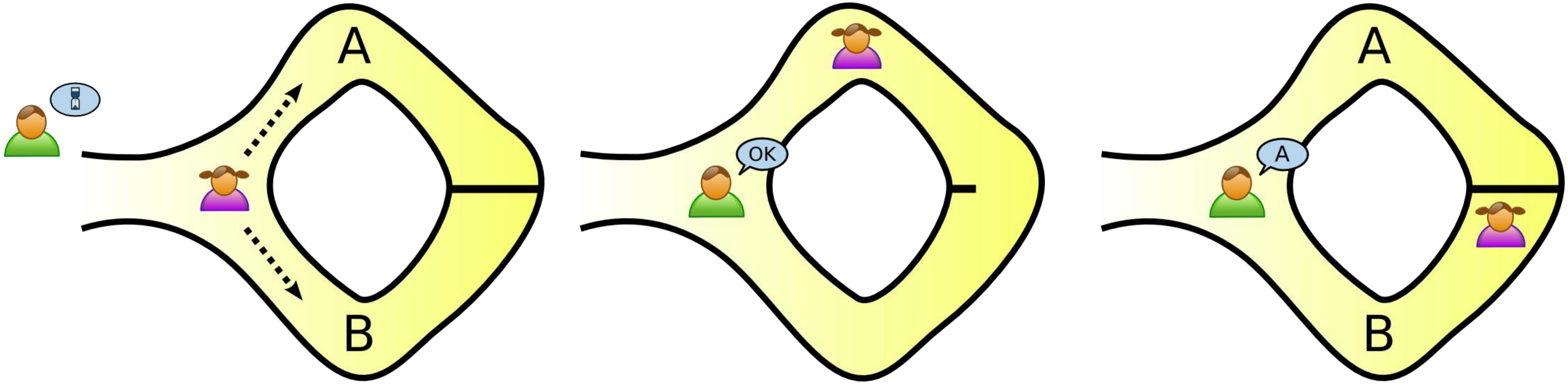


Example: The Ali Baba cave



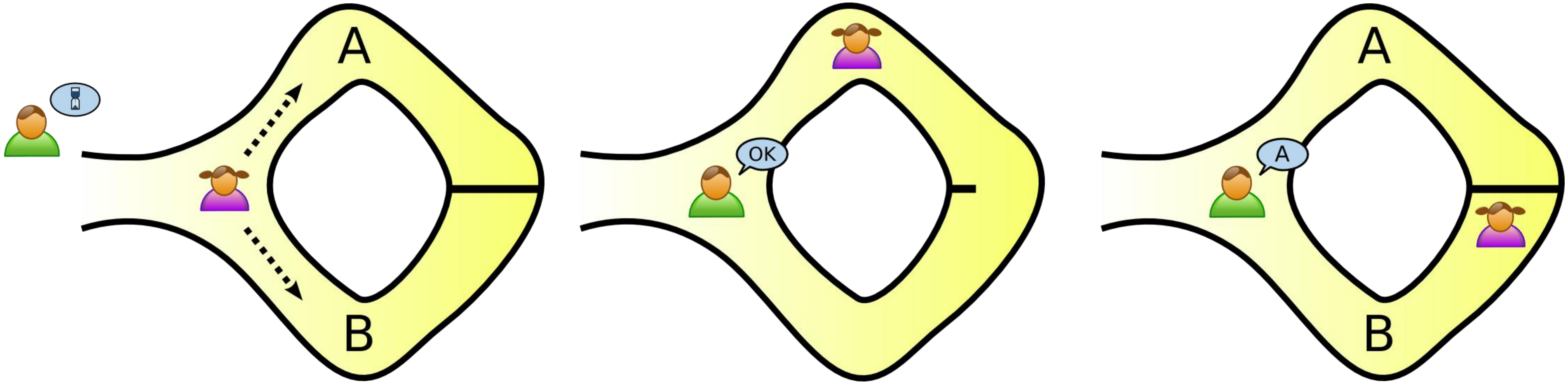
- Third-party observers

Example: The Ali Baba cave



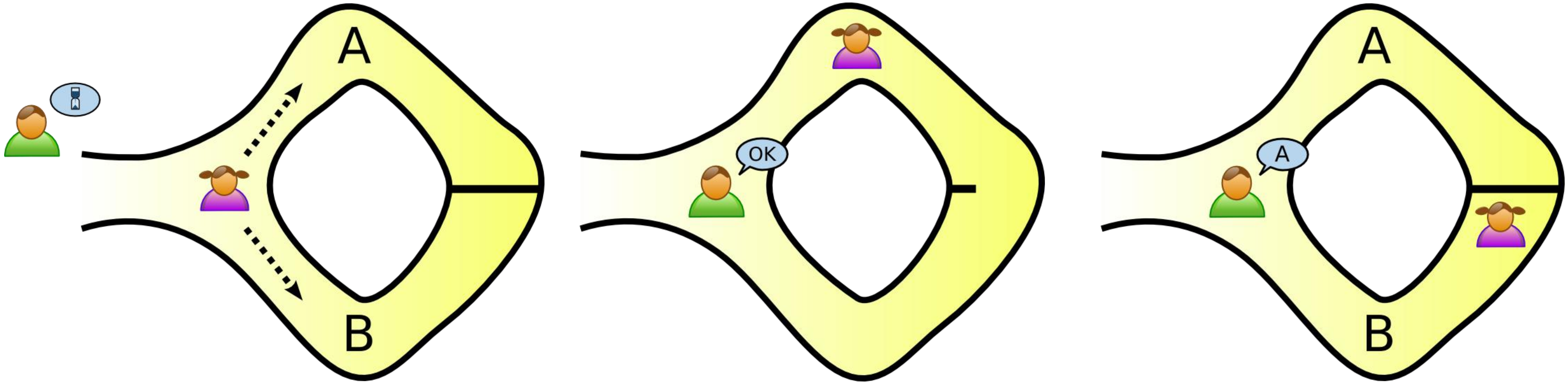
- Third-party observers
  - two people to fake, never be convincing to anyone but the original participants

Example: The Ali Baba cave



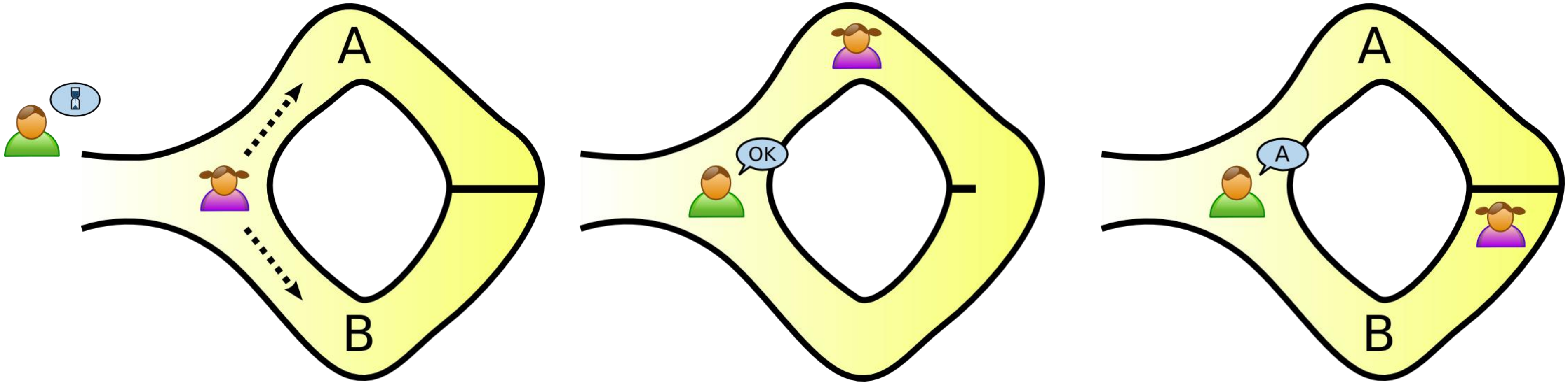
- Third-party observers
  - two people to fake, never be convincing to anyone but the original participants
- Verifier flipping a coin on-camera

Example: The Ali Baba cave



- Third-party observers
  - two people to fake, never be convincing to anyone but the original participants
- Verifier flipping a coin on-camera
  - convince the world, counter to Prover's stated wishes

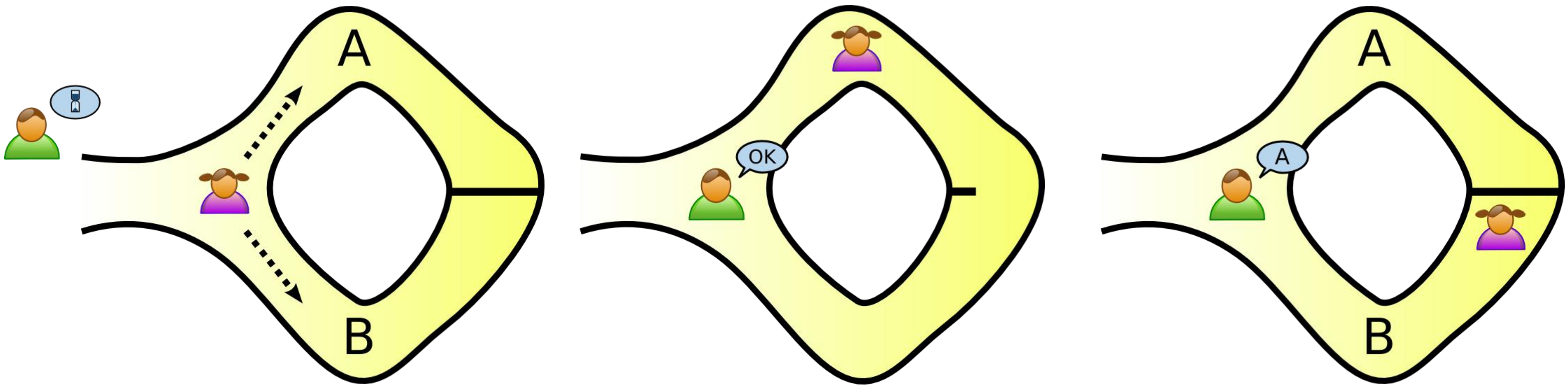
Example: The Ali Baba cave



- Third-party observers
  - two people to fake, never be convincing to anyone but the original participants
- Verifier flipping a coin on-camera
  - convince the world, counter to Prover's stated wishes

- In a single trial

Example: The Ali Baba cave

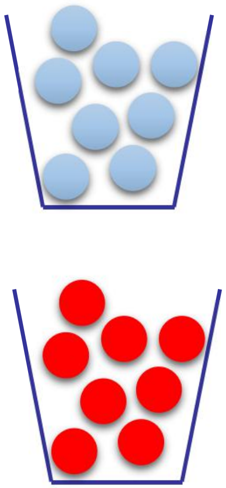


- Third-party observers
  - two people to fake, never be convincing to anyone but the original participants
- Verifier flipping a coin on-camera
  - convince the world, counter to Prover's stated wishes

- In a single trial
  - ok but could be observed by a third party, or recorded

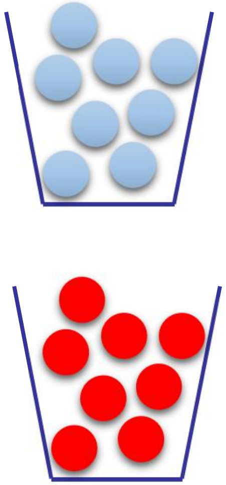
Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



Example: Non-color-blindness

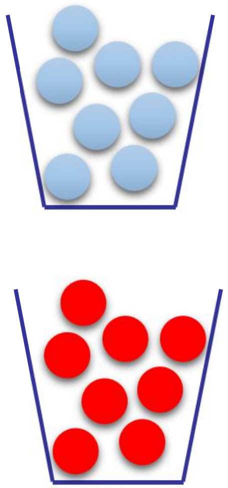
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



1. I draw at random either a red ball or a blue ball and show it to you

Example: Non-color-blindness

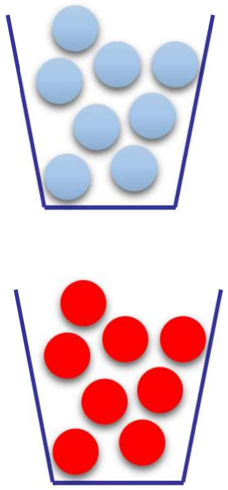
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



1. I draw at random either a red ball or a blue ball and show it to you
2. You say “red” or “blue”

Example: Non-color-blindness

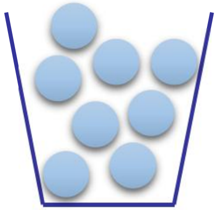
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)

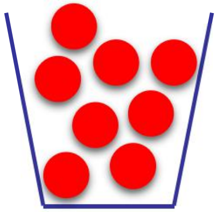


1. I draw at random either a red ball or a blue ball and show it to you
2. You say “red” or “blue”
3. repeat this 10 times

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



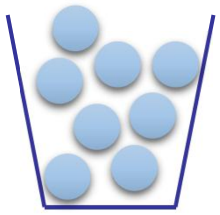


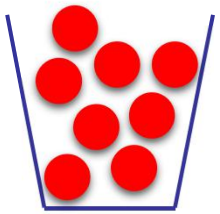
1. I draw at random either a red ball or a blue ball and show it to you
2. You say “red” or “blue”
3. repeat this 10 times

If you got all the answers right, you can tell red from blue.

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



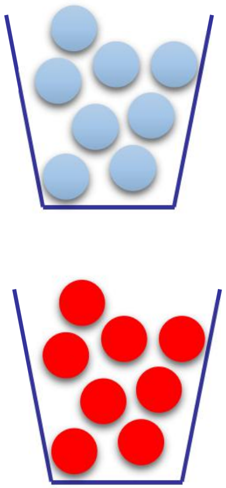


1. I draw at random either a red ball or a blue ball and show it to you
2. You say “red” or “blue”
3. repeat this 10 times

If you got all the answers right, you can tell red from blue.

Example: Non-color-blindness

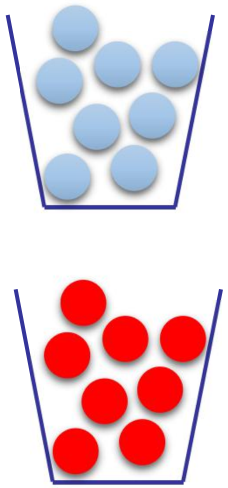
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



- Soundness

Example: Non-color-blindness

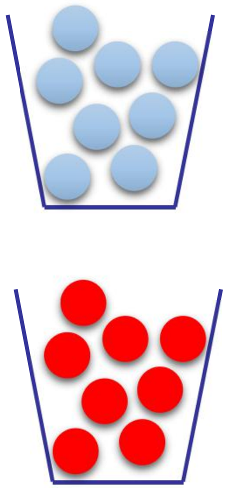
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



- Soundness
  - If the verifier accepts then the property (the prover is not color blind) holds with high probability

Example: Non-color-blindness

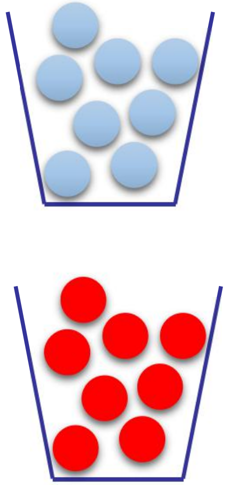
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



- Soundness
  - If the verifier accepts then the property (the prover is not color blind) holds with high probability
- Completeness

Example: Non-color-blindness

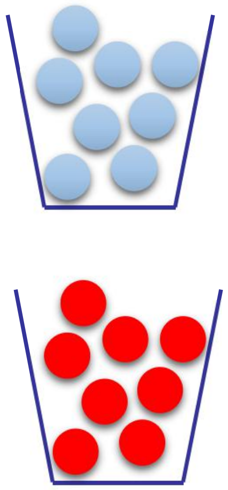
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



- Soundness
  - If the verifier accepts then the property (the prover is not color blind) holds with high probability
- Completeness
  - If property (the prover is not color blind) holds then the verifier always accepts

Example: Non-color-blindness

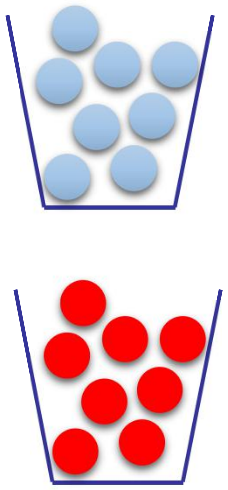
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



- What knowledge did I gain from this interaction?

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)

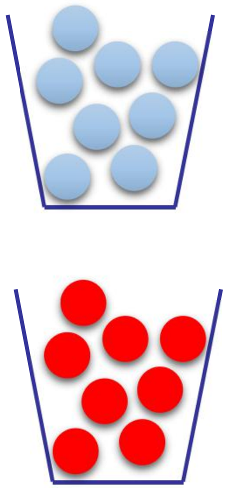


- What knowledge did I gain from this interaction?

I learned that you can tell red from blue!

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



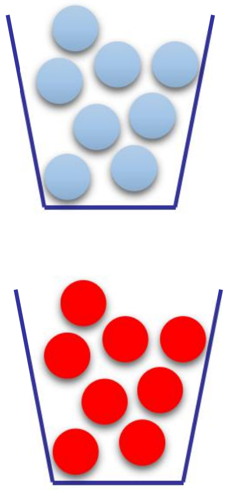
- What knowledge did I gain from this interaction?

  I learned that you can tell red from blue!

  But I also learned the colors of the balls in each bucket.

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



- What knowledge did I gain from this interaction?

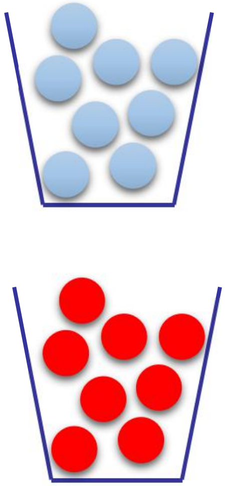
  I learned that you can tell red from blue!

  But I also learned the colors of the balls in each bucket.

  Suppose I was color blind, then I used you to gain some extra knowledge!

Example: Non-color-blindness

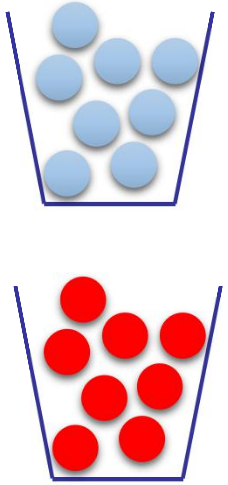
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



1. I pull at random either two balls from the same bucket OR one ball from Bucket 1 and one ball from Bucket 2

Example: Non-color-blindness

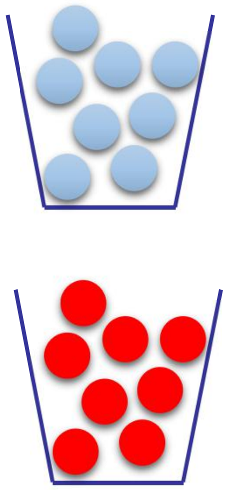
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



1. I pull at random either two balls from the same bucket OR one ball from Bucket 1 and one ball from Bucket 2
2. You say “same” or “different”

Example: Non-color-blindness

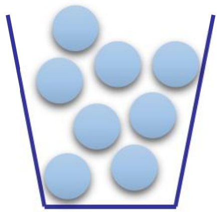
- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)

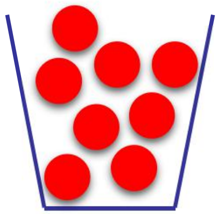


1. I pull at random either two balls from the same bucket OR one ball from Bucket 1 and one ball from Bucket 2
2. You say “same” or “different”
3. We repeat this 10 times

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



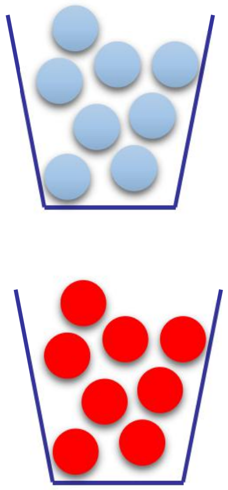


1. I pull at random either two balls from the same bucket OR one ball from Bucket 1 and one ball from Bucket 2
2. You say “same” or “different”
3. We repeat this 10 times

If you got all the answers right, you can tell red from blue.

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)

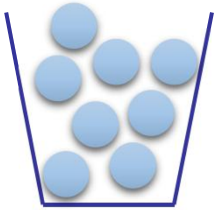


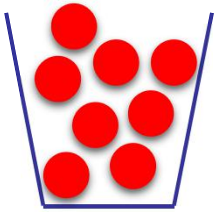
Zero-knowledge:

Suppose I was color blind but you are not.

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)





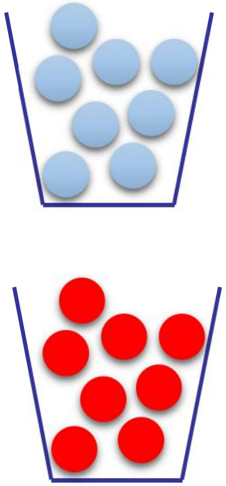
Zero-knowledge:

Suppose I was color blind but you are not.

- In the first protocol, I cannot predict your Answer ahead of time.

Example: Non-color-blindness

- You (the prover) want to convince me (the verifier) that you are not color-blind (two colors are completely identical)



Zero-knowledge:

Suppose I was color blind but you are not.

- In the first protocol, I cannot predict your Answer ahead of time.
- In the second protocol, I know what you will say, so I do not gain knowledge when you say it.

Zero Knowledge

- **Zero-knowledge:** There exists a simulator S such that S(x) is indistinguishable from a real proof execution

Zero Knowledge

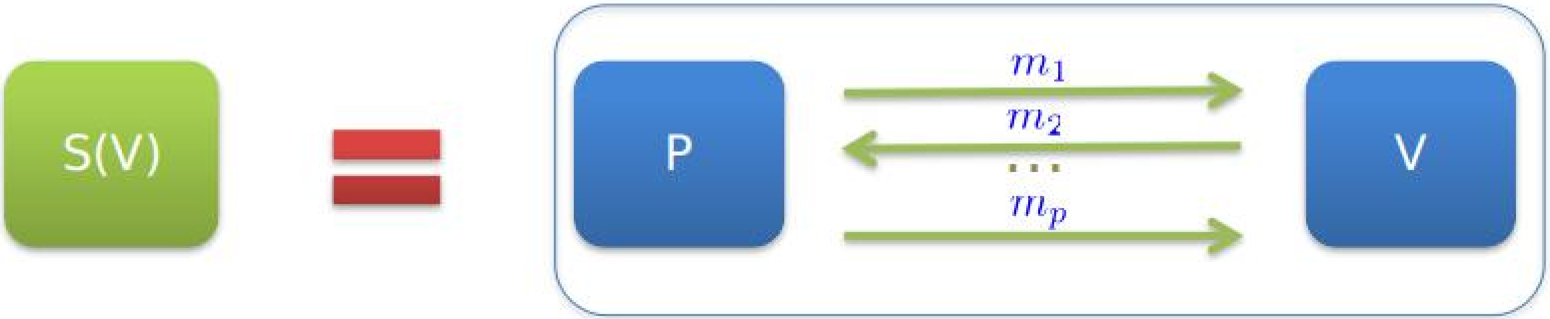
- **Zero-knowledge:** There exists a simulator S such that S(x) is indistinguishable from a real proof execution

- The verifier’s view of the interaction with the prover can be efficiently simulated without interacting with the prover

Zero Knowledge

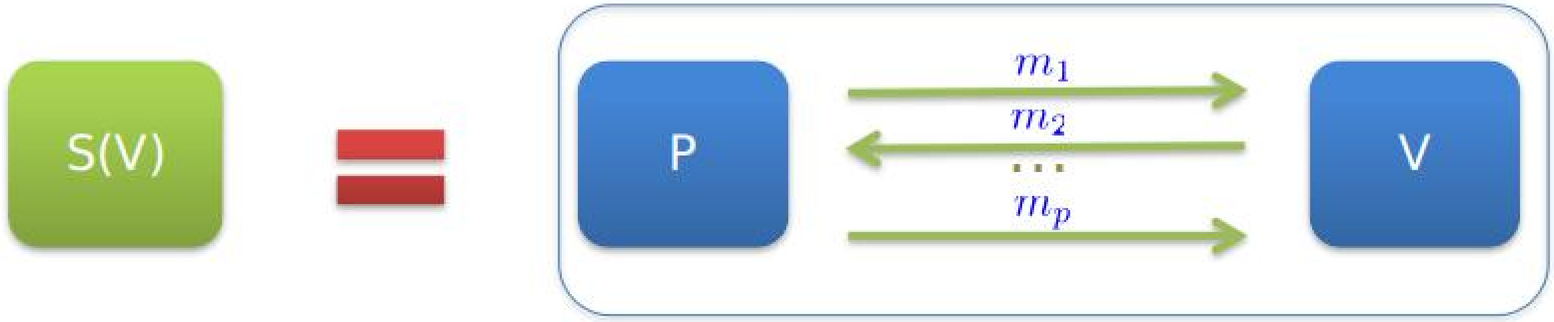
- **Zero-knowledge:** There exists a simulator S such that S(x) is indistinguishable from a real proof execution

- The verifier's view of the interaction with the prover can be efficiently simulated without interacting with the prover



Zero Knowledge

- **Zero-knowledge:** There exists a simulator S such that S(x) is indistinguishable from a real proof execution
- The verifier's view of the interaction with the prover can be efficiently simulated without interacting with the prover



Probability distributions on transcripts are indistinguishable!

Zero Knowledge

- **Zero-knowledge:** There exists a simulator S such that S(x) is indistinguishable from a real proof execution

- The verifier’s view of the interaction with the prover can be efficiently simulated without interacting with the prover

Probability distributions on transcripts are indistinguishable!

Zero Knowledge

- We speak of perfect zero-knowledge if the distributions produced by the simulator and the proof protocol are distributed exactly the same.

Zero Knowledge

- We speak of perfect zero-knowledge if the distributions produced by the simulator and the proof protocol are distributed exactly the same.

- Statistical zero-knowledge means that the distributions are not necessarily exactly the same, but they are statistically close, meaning that their statistical difference is a negligible function.

Zero Knowledge

- We speak of perfect zero-knowledge if the distributions produced by the simulator and the proof protocol are distributed exactly the same.
- Statistical zero-knowledge means that the distributions are not necessarily exactly the same, but they are statistically close, meaning that their statistical difference is a negligible function.
- We speak of computational zero-knowledge if no efficient algorithm can distinguish the two distributions.

Zero Knowledge Proof

- A fundamental theorem:
  - Any language in **NP** can be proven in zero knowledge

Zero Knowledge Proof

- A fundamental theorem:
  - Any language in **NP** can be proven in zero knowledge

- **NP** = the class of all languages that can be verified efficiently
  - There exists a polytime *V* such that

For every $x \in \mathcal{L}$ there exists a w such that $V(x,w) = 1$.

For every $x \notin \mathcal{L}$ and every w, it holds that $V(x,w) = 0$.

Zero Knowledge Proof

- A fundamental theorem:
  - Any language in **NP** can be proven in zero knowledge

- **NP** = the class of all languages that can be verified efficiently
  - There exists a polytime *V* such that

For every $x \in \mathcal{L}$ there exists a w such that $V(x, w) = 1$.

For every $x \notin \mathcal{L}$ and every w, it holds that $V(x, w) = 0$.

Goldreich, Oded; Micali, Silvio; Wigderson, Avi (1991). "Proofs that yield nothing but their validity". Journal of the ACM. 38 (3): 690–728.

Zero Knowledge Proof

- GMW91: NP-complete graph coloring problem with three colors assuming the existence of unbreakable encryption (the existence of one way functions)
  - every problem in NP can be efficiently reduced to this problem
  - all problems in NP have zero-knowledge proofs

Zero Knowledge Proof

- GMW91: NP-complete graph coloring problem with three colors assuming the existence of unbreakable encryption (the existence of one way functions)
  - every problem in NP can be efficiently reduced to this problem
  - all problems in NP have zero-knowledge proofs

- Russell Impagliazzo and Moti Yung as well as Ben-Or et al:
  - assuming one-way functions or unbreakable encryption
  - there are zero-knowledge proofs for all problems in IP = PSPACE
  - in other words, anything that can be proved by an interactive proof system can be proved with zero knowledge.

Zero Knowledge Proof

- GMW91: NP-complete graph coloring problem with three colors assuming the existence of unbreakable encryption (the existence of one way functions)
  - every problem in NP can be efficiently reduced to this problem
  - all problems in NP have zero-knowledge proofs
- Russell Impagliazzo and Moti Yung as well as Ben-Or et al:
  - assuming one-way functions or unbreakable encryption
  - there are zero-knowledge proofs for all problems in IP = PSPACE
  - in other words, anything that can be proved by an interactive proof system can be proved with zero knowledge.

- Russell Impagliazzo, Moti Yung: Direct Minimum-Knowledge Computations. CRYPTO 1987: 40-51
- Ben-Or, Michael; Goldreich, Oded; Goldwasser, Shafi; Hastad, Johan; Kilian, Joe; Micali, Silvio; Rogaway, Phillip (1990). "Everything provable is provable in zero-knowledge". In Goldwasser, S. Advances in Cryptology—CRYPTO '88. Lecture Notes in Computer Science. 403. Springer-Verlag. pp. 37–56.

Zero Knowledge

- An amazing concept

Zero Knowledge

- An amazing concept
- But, can it be efficient?

Zero Knowledge

- An amazing concept

- But, can it be efficient?
  - It seems that zero-knowledge protocols for "interesting languages" are complicated and expensive

Zero Knowledge

- An amazing concept
- But, can it be efficient?
  - It seems that zero-knowledge protocols for "interesting languages" are complicated and expensive
- Zero knowledge is often avoided at significant cost

Sigma Protocol

- A way to obtain efficient zero knowledge

Sigma Protocol

- A way to obtain efficient zero knowledge
  - Many general tools
  - Many interesting languages can be proven with a sigma protocol

Schnorr DLOG

- Let **G** be a group of order **q**, with generator **g**

Schnorr DLOG

- Let **G** be a group of order **q**, with generator **g**
- **P** and **V** have input $\mathbf{h} \in \mathbf{G}$, **P** has **w** such that $\mathbf{g}^{\mathbf{w}} = \mathbf{h}$

Schnorr DLOG

- Let **G** be a group of order **q**, with generator **g**
- **P** and **V** have input $\mathbf{h} \in \mathbf{G}$, **P** has **w** such that $\mathbf{g}^{\mathbf{w}} = \mathbf{h}$
- **P** proves that to **V** that it knows $\mathrm{DLOG}_{\mathbf{g}}(\mathbf{h})$

Schnorr DLOG

- Let $\mathbf{G}$ be a group of order $\mathbf{q}$, with generator $\mathbf{g}$
- $\mathbf{P}$ and $\mathbf{V}$ have input $\mathbf{h} \in \mathbf{G}$, $\mathbf{P}$ has $\mathbf{w}$ such that $\mathbf{g}^{\mathbf{w}} = \mathbf{h}$
- $\mathbf{P}$ proves that to $\mathbf{V}$ that it knows $\mathrm{DLOG}_{\mathbf{g}}(\mathbf{h})$
  - $\mathbf{P}$ chooses a random $\mathbf{r}$ and sends $\mathbf{a} = \mathbf{g}^{\mathbf{r}}$ to $\mathbf{V}$

Schnorr DLOG

- Let **G** be a group of order **q**, with generator **g**
- **P** and **V** have input $\mathbf{h} \in \mathbf{G}$, **P** has **w** such that $\mathbf{g}^{\mathbf{w}} = \mathbf{h}$
- **P** proves that to **V** that it knows $\mathrm{DLOG}_{\mathbf{g}}(\mathbf{h})$
  - **P** chooses a random **r** and sends $\mathbf{a}=\mathbf{g}^{\mathbf{r}}$ to **V**
  - **V** sends **P** a random $\mathbf{e} \in \{0,1\}^{\mathbf{t}}$

Schnorr DLOG

- Let **G** be a group of order **q**, with generator **g**
- **P** and **V** have input $\mathbf{h} \in \mathbf{G}$, **P** has **w** such that $\mathbf{g}^{\mathbf{w}} = \mathbf{h}$
- **P** proves that to **V** that it knows $\mathrm{DLOG}_{\mathbf{g}}(\mathbf{h})$
  - **P** chooses a random **r** and sends $\mathbf{a} = \mathbf{g}^{\mathbf{r}}$ to **V**
  - **V** sends **P** a random $\mathbf{e} \in \{0,1\}^{\mathbf{t}}$
  - **P** sends $\mathbf{z} = \mathbf{r} + \mathbf{ew} \bmod \mathbf{q}$ to **V**

Schnorr DLOG

- Let $\mathbf{G}$ be a group of order $\mathbf{q}$, with generator $\mathbf{g}$
- $\mathbf{P}$ and $\mathbf{V}$ have input $\mathbf{h} \in \mathbf{G}$, $\mathbf{P}$ has $\mathbf{w}$ such that $\mathbf{g}^{\mathbf{w}} = \mathbf{h}$
- $\mathbf{P}$ proves that to $\mathbf{V}$ that it knows $\mathrm{DLOG}_{\mathbf{g}}(\mathbf{h})$
  - $\mathbf{P}$ chooses a random $\mathbf{r}$ and sends $\mathbf{a}=\mathbf{g}^{\mathbf{r}}$ to $\mathbf{V}$
  - $\mathbf{V}$ sends $\mathbf{P}$ a random $\mathbf{e} \in \{0,1\}^{\mathbf{t}}$
  - $\mathbf{P}$ sends $\mathbf{z}=\mathbf{r}+\mathbf{ew}$ mod $\mathbf{q}$ to $\mathbf{V}$
  - $\mathbf{V}$ checks that $\mathbf{g}^{\mathbf{z}} = \mathbf{ah}^{\mathbf{e}}$

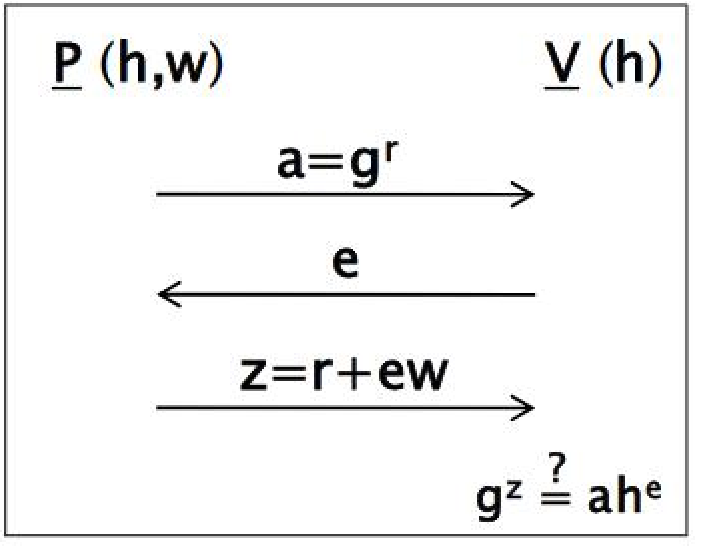
Schnorr DLOG

- Let G be a group of order q, with generator g
- P and V have input $h \in G$, P has w such that $g^w = h$
- P proves that to V that it knows $DLOG_g(h)$
  - P chooses a random r and sends $a=g^r$ to V
  - V sends P a random $e \in \{0,1\}^t$
  - P sends $z=r+ew \bmod q$ to V
  - V checks that $g^z = ah^e$

- **Completeness**
  - $g^z = g^{r+ew} = g^r(g^w)^e = ah^e$

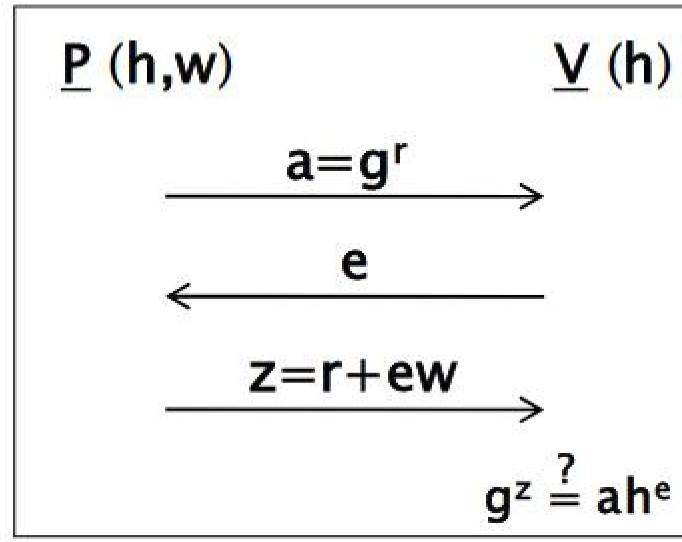
Schnorr DLOG

- **Proof of knowledge**



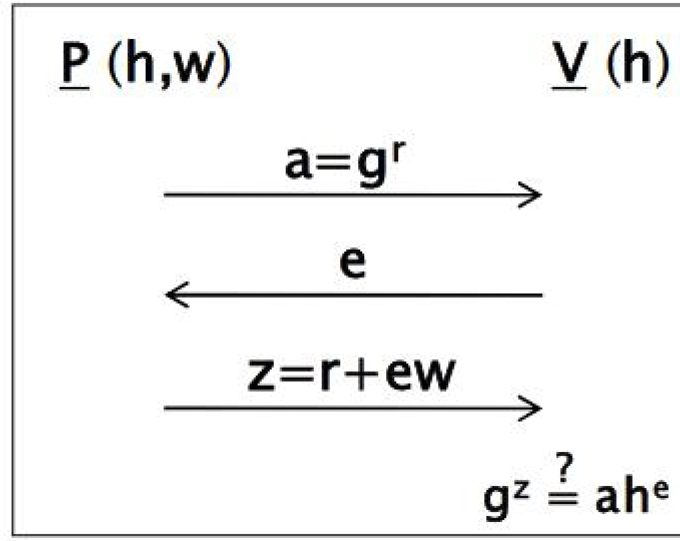
Schnorr DLOG

- **Proof of knowledge**
  - Assume **P** can answer two queries **e** and **e′** for the same **a**



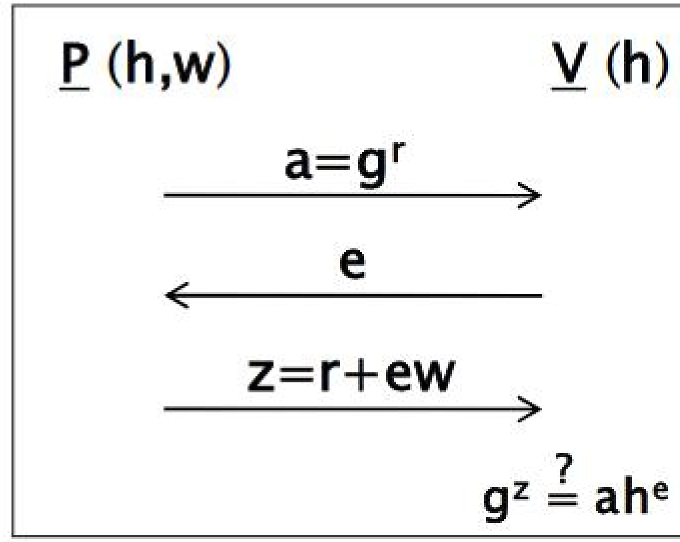
Schnorr DLOG

- **Proof of knowledge**
  - Assume **P** can answer two queries **e** and **e′** for the same **a**
  - Then, have $g^z = ah^e$, $g^{z'} = ah^{e'}$



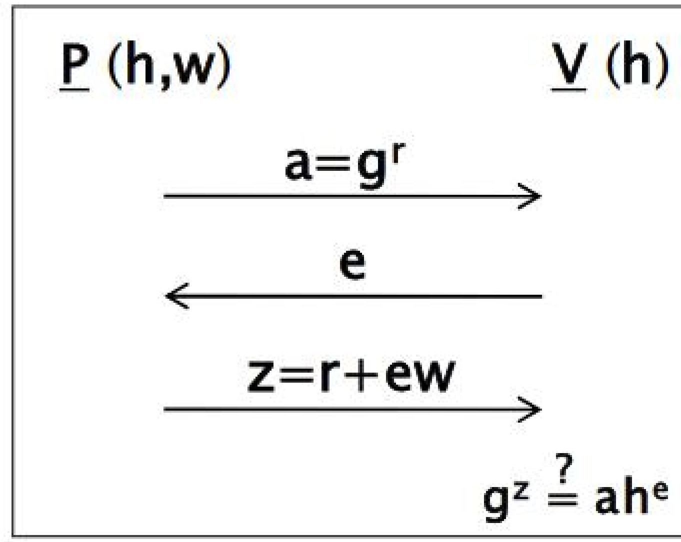
Schnorr DLOG

- **Proof of knowledge**
  - Assume **P** can answer two queries **e** and **e′** for the same **a**
  - Then, have $\mathbf{g^z = ah^e}$, $\mathbf{g^{z'}=ah^{e'}}$
  - Thus, $\mathbf{g^z h^{-e} = g^{z'} h^{-e'}}$ and $\mathbf{g^{z-z'}=h^{e-e'}}$



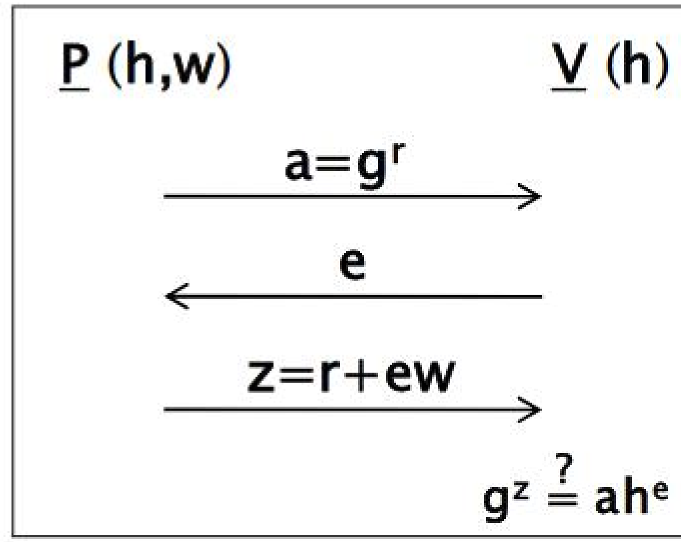
Schnorr DLOG

- **Proof of knowledge**
  - Assume **P** can answer two queries **e** and **e′** for the same **a**
  - Then, have $g^{z} = ah^{e}$, $g^{z'}=ah^{e'}$
  - Thus, $g^{z}h^{-e} = g^{z'}h^{-e'}$ and $g^{z-z'}=h^{e-e'}$
  - Therefore $h = g^{(z-z')/(e-e')}$



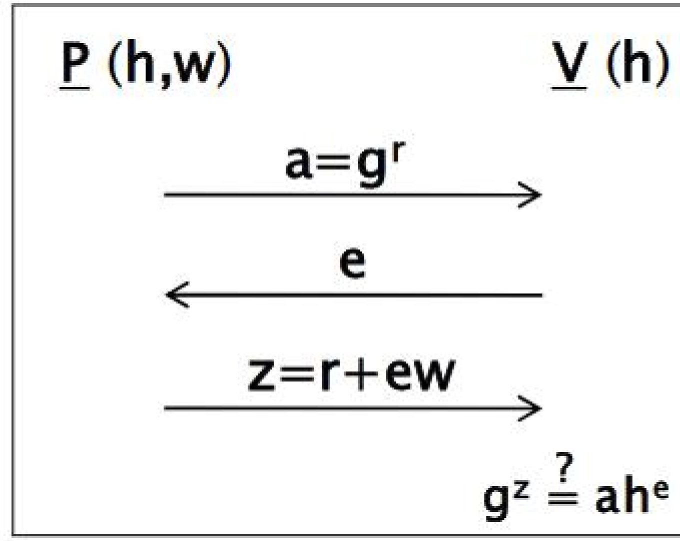
Schnorr DLOG

- **Proof of knowledge**
  - Assume **P** can answer two queries **e** and **e′** for the same **a**
  - Then, have $g^{z} = ah^{e}$, $g^{z'} = ah^{e'}$
  - Thus, $g^{z}h^{-e} = g^{z'}h^{-e'}$ and $g^{z-z'} = h^{e-e'}$
  - Therefore $h = g^{(z-z')/(e-e')}$
  - That is: $DLOG_{g}(h) = (z-z')/(e-e')$



Schnorr DLOG

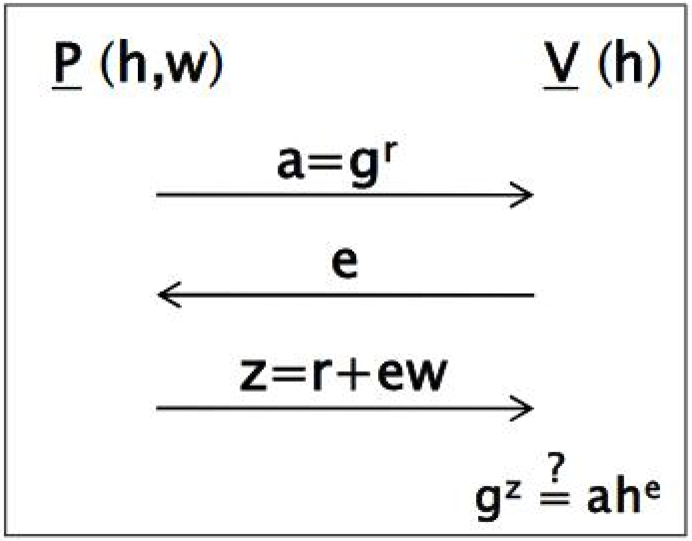
- **Proof of knowledge**
  - Assume **P** can answer two queries **e** and **e′** for the same **a**
  - Then, have $g^z = ah^e$, $g^{z'}=ah^{e'}$
  - Thus, $g^z h^{-e} = g^{z'} h^{-e'}$ and $g^{z-z'}=h^{e-e'}$
  - Therefore $h = g^{(z-z')/(e-e')}$
  - That is: DLOG$g(h) = (z-z')/(e-e')$



- **Conclusion:**
  - If P can answer with probability greater than $1/2^t$, then it must know the dlog

Schnorr DLOG

- **What about zero knowledge?**



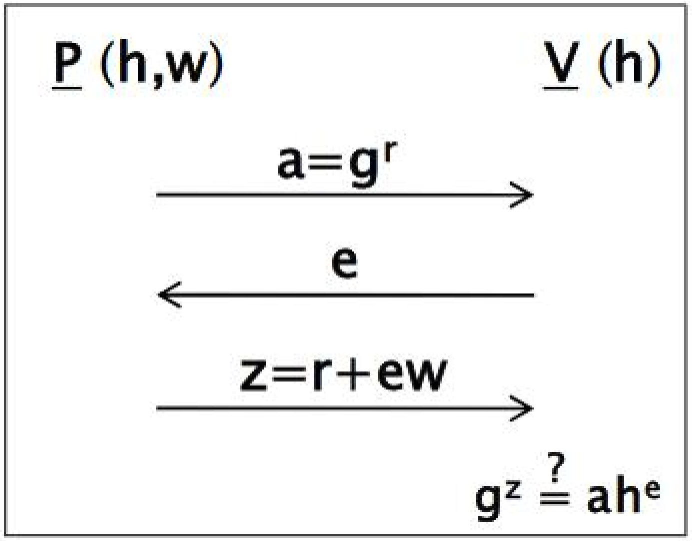
Schnorr DLOG

- **What about zero knowledge? Seems not…**

$$
\begin{array}{lcr}
\underline{P}\ (h,w) & & \underline{V}\ (h) \\
& \xrightarrow{\quad a=g^r \quad} & \\
& \xleftarrow{\quad e \quad} & \\
& \xrightarrow{\quad z=r+ew \quad} & \\
& & g^z \stackrel{?}{=} ah^e
\end{array}
$$

Schnorr DLOG

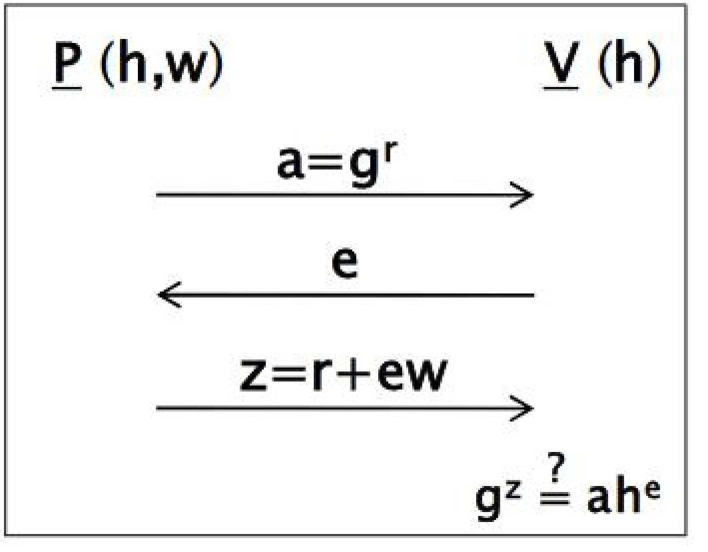
- **What about zero knowledge? Seems not…**
- **Honest-verifier zero knowledge**
  - Choose a random $\mathbf{z}$ and $\mathbf{e}$, and compute $\mathbf{a} = \mathbf{g^z h^{-e}}$
  - Clearly, ($\mathbf{a,e,z}$) have same distribution, and $\mathbf{g^z = ah^e}$



Schnorr DLOG

- **What about zero knowledge? Seems not…**
- **Honest-verifier zero knowledge**
  - Choose a random **z** and **e**, and compute $\mathbf{a} = \mathbf{g^z h^{-e}}$
  - Clearly, (**a,e,z**) have same distribution, and $\mathbf{g^z = ah^e}$

- **This is not very strong, but we will see that it yields efficient general ZK**



Sigma Protocol

Definitions

- **Sigma protocol template**
  - **Common input: P** and **V** both have **x**
  - **Private input: P** has **w** such that **(x,w)**$\in$**R**
  - **Protocol:**
    - **P** sends a message **a**
    - **V** sends a <u>random</u> **t**-bit string **e**
    - **P** sends a reply **z**
    - **V** accepts based solely on (**x,a,e,z**)

Sigma Protocol

- **Completeness: as usual**
- **<u>Special</u> soundness:**
  - There exists an algorithm **A** that given any **x** and pair of transcripts (**a**,**e**,**z**),(**a**,**e′**,**z′**) with **e**≠**e′** outputs **w** s.t. (**x**,**w**)∈**R**
- **<u>Special</u> honest-verifier ZK**
  - There exists an **M** that given **x** and **e** outputs (**a**,**e**,**z**) which is distributed exactly like a real execution where **V** sends **e**

Sigma Protocol DH Tuple

Sigma Protocol DH Tuple

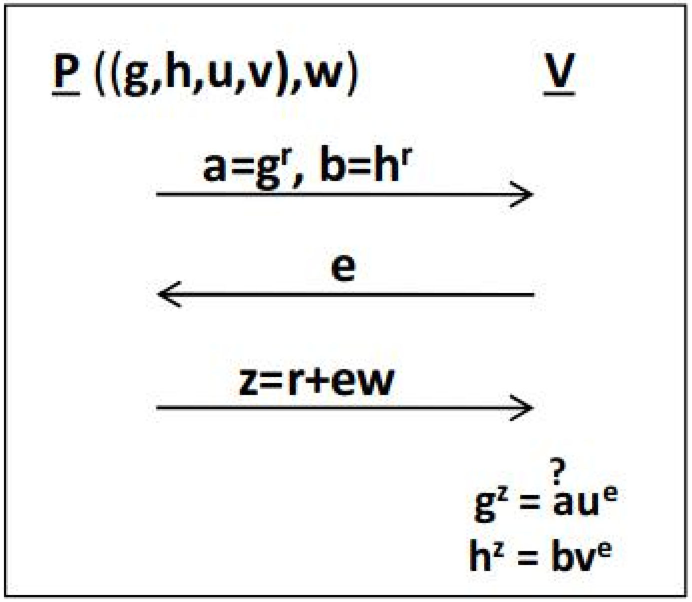
- **Relation R of Diffie-Hellman tuples**
  - (**g,h,u,v**) $\in$**R** iff exists **w** s.t. **u=g^w^** and **v = h^w^**
  - Useful in many protocols

Sigma Protocol DH Tuple

- **Relation R of Diffie-Hellman tuples**
  - **(g,h,u,v)** $\in$**R** iff exists **w** s.t. **u=g^{w}** and **v = h^{w}**
  - Useful in many protocols
- **Protocol**
  - **P** chooses a random **r** and sends **a=g^{r}**, **b=h^{r}**
  - **V** sends a random **e**
  - **P** sends **z=r+ew** mod **q**
  - **V** checks that **g^{z}=au^{e}**, **h^{z}=bv^{e}**

Sigma Protocol DH Tuple

- **Completeness:** as in DLOG



Sigma Protocol DH Tuple

- **Completeness:** as in DLOG
- **Special soundness:**
  - Given **(a,b,e,z)**,**(a,b,e′,z′)**, we have $g^z = au^e$, $g^{z'} = au^{e'}$, $h^z = bv^e$, $h^{z'} = bv^{e'}$ and so like in DLOG on both
    - $w = (z - z')(e - e')$

<u>P</u> ((g,h,u,v),w) <u>V</u>

$$\xrightarrow{a = g^r,\ b = h^r}$$

$$\xleftarrow{e}$$

$$\xrightarrow{z = r + ew}$$

$$g^z \stackrel{?}{=} au^e$$

$$h^z = bv^e$$

Sigma Protocol DH Tuple

- **Completeness:** as in DLOG
- **Special soundness:**
  - Given (**a,b,e,z**),(**a,b,e′,z′**), we have $\mathbf{g^z=au^e, g^{z'}=au^{e'}, h^z=bv^e, h^{z'}=bv^{e'}}$ and so like in DLOG on both
    - $\mathbf{w = (z\text{-}z')(e\text{-}e')}$
- **Special HVZK**
  - Given (**g,h,u,v**) and **e**, choose random **z** and compute
    - $\mathbf{a = g^z u^{-e}}$
    - $\mathbf{b = h^z v^{-e}}$

$$\underline{P}\ ((g,h,u,v),w) \qquad\qquad \underline{V}$$

$$\xrightarrow{\mathbf{a=g^r,\ b=h^r}}$$

$$\xleftarrow{\mathbf{e}}$$

$$\xrightarrow{\mathbf{z=r+ew}}$$

$$\mathbf{g^z \overset{?}{=} au^e}$$
$$\mathbf{h^z = bv^e}$$

Basic Properties

- **Any sigma protocol is an interactive proof with soundness error 2^{-t}**

Basic Properties

- **Any sigma protocol is an interactive proof with soundness error 2^{-t}**
- **Properties of sigma protocols are invariant under parallel composition**

Basic Properties

- **Any sigma protocol is an interactive proof with soundness error 2^{-t}**
- **Properties of sigma protocols are invariant under parallel composition**
- **Any sigma protocol is a proof of knowledge with error 2^{-t}**
  – The difference between the probability that $\mathbf{P}^*$ convinces **V** and the probability that **K** obtains a witness is at most $\mathbf{2^{-t}}$

Tools for Sigma Protocols

- **Prove compound statements**
  - AND, OR, subset
  - Can be done efficiently
- **ZK from sigma protocols**
  - Can first make a compound sigma protocol and then compile it
- **ZKPOK from sigma protocols**

AND of Sigma Protocol

- To prove the AND of multiple statements
  - Run all in parallel
  - Can use the same verifier challenge **e** in all

AND of Sigma Protocol

- To prove the AND of multiple statements
  - Run all in parallel
  - Can use the same verifier challenge **e** in all

- Sometimes it’s possible to do better than this
  - Statements can be batched

OR of Sigma Protocol

- This is more complicated
  - Given two statements and two appropriate Sigma protocols, wish to prove that at least one is true, without revealing which

OR of Sigma Protocol

- This is more complicated
  - Given two statements and two appropriate Sigma protocols, wish to prove that at least one is true, without revealing which

- The solution
  - Using the simulator, if **e** is known ahead of time it is possible to cheat
  - We construct a protocol where the prover can cheat in one out of the two proofs

OR of Sigma Protocol

- **The template for x_0 or x_1:**

OR of Sigma Protocol

- **The template for x_0 or x_1:**
  - **P** sends two first messages $(\mathbf{a}_0, \mathbf{a}_1)$

OR of Sigma Protocol

- **The template for x_0 or x_1:**
  - **P** sends two first messages $(\mathbf{a_0},\mathbf{a_1})$
  - **V** sends a single challenge **e**

OR of Sigma Protocol

- **The template for x_0 or x_1:**
  - **P** sends two first messages $(\mathbf{a_0},\mathbf{a_1})$
  - **V** sends a single challenge $\mathbf{e}$
  - **P** replies with
    - Two challenges $\mathbf{e_0},\mathbf{e_1}$ s.t. $\mathbf{e_0} \oplus \mathbf{e_1} = \mathbf{e}$
    - Two final messages $\mathbf{z_0},\mathbf{z_1}$

OR of Sigma Protocol

- **The template for x_0 or x_1:**
  - **P** sends two first messages $(\mathbf{a}_0,\mathbf{a}_1)$
  - **V** sends a single challenge **e**
  - **P** replies with
    - Two challenges $\mathbf{e}_0,\mathbf{e}_1$ s.t. $\mathbf{e}_0 \oplus \mathbf{e}_1 = \mathbf{e}$
    - Two final messages $\mathbf{z}_0,\mathbf{z}_1$
  - **V** accepts if $\mathbf{e}_0 \oplus \mathbf{e}_1 = \mathbf{e}$ and $(\mathbf{a}_0,\mathbf{e}_0,\mathbf{z}_0),(\mathbf{a}_1,\mathbf{e}_1,\mathbf{z}_1)$ are both accepting

OR of Sigma Protocol

- **P sends two first messages (a_0,a_1)**
  - P has a witness for x_0 (and not for x_1)
  - P chooses a random e_1 and runs SIM to get (a_1,e_1,z_1)
  - P sends (a_0,a_1)
- **V sends a single challenge e**
- **P replies with e_0,e_1 s.t. $e_0 \oplus e_1 = e$ and with z_0,z_1**
  - P already has z_1 and can compute z_0 using the witness

OR of Sigma Protocol

- **P sends two first messages (a_0,a_1)**
  - P has a witness for x_0 (and not for x_1)
  - P chooses a random e_1 and runs SIM to get (a_1,e_1,z_1)
  - P sends (a_0,a_1)
- **V sends a single challenge e**
- **P replies with e_0,e_1 s.t. $e_0 \oplus e_1 = e$ and with z_0,z_1**
  - P already has z_1 and can compute z_0 using the witness

<u>Soundness</u>

- P doesn't know a witness for x_1, so can only answer for a single e_1
- This means that **e** defines a single challenge e_0, like in a regular proof

OR of Sigma Protocol

- **Special soundness**
  - Relative to first message ($\mathbf{a}_0$,$\mathbf{a}_1$), and two different $\mathbf{e}$,$\mathbf{e}'$, at least one of $\mathbf{e}_0 \neq \mathbf{e}'_0$ or $\mathbf{e}_1 \neq \mathbf{e}'_1$ (because $\mathbf{e}_0 \oplus \mathbf{e}_1 = \mathbf{e}$ and $\mathbf{e}'_0 \oplus \mathbf{e}'_1 = \mathbf{e}'$)
  - Thus, we will obtain two different continuations for at least one of the statements
- **Honest verifier ZK**
  - Can choose both $\mathbf{e}_0$,$\mathbf{e}_1$, so no problem

ZK from Sigma: Preliminaries

- **Commitment schemes:**
  - **Binding**: after the commitment phase, the committer cannot change the value
  - **Hiding**: the receiver does not know anything about the commitment

ZK from Sigma: Preliminaries

- **Commitment schemes:**
  - **Binding**: after the commitment phase, the committer cannot change the value
  - **Hiding**: the receiver does not know anything about the commitment
- **Variants**
  - Perfect and computational binding
  - Perfect and computational hiding
  - Cannot have both perfect binding and hiding

ZK from Sigma Protocols

- **The basic idea**
  - Have **V** first commit to its challenge **e** using a **perfectly-hiding commitment**

ZK from Sigma Protocols

- **The basic idea**
  - Have **V** first commit to its challenge **e** using a **perfectly-hiding commitment**
- **The protocol**
  - **P** sends the 1st message α of the commit protocol
  - **V** sends a commitment c=$\mathbf{Com}_{\alpha}$(**e;r**)
  - **P** sends a message **a**
  - **V** sends (**e,r**)
  - **P** checks that c=$\mathbf{Com}_{\alpha}$(**e;r**) and if yes sends a reply **z**
  - **V** accepts based on (**x,a,e,z**)

ZK from Sigma Protocols

- **Soundness:**
  - The perfectly hiding commitment reveals nothing about **e** and so soundness is preserved

ZK from Sigma Protocols

- **Soundness:**
  - The perfectly hiding commitment reveals nothing about **e** and so soundness is preserved
- **Zero knowledge**
  - In order to simulate:
    - Send **a**′ generated by the simulator, for a random **e**′
    - Receiver **V**'s decommitment to **e**
    - Run the simulator again with **e**, rewind **V** and send **a**
      - Repeat until **V** decommits to **e** again
    - Conclude by sending **z**
  - Analysis…

ZKPOK from Sigma Protocols

- Is the previous protocol a proof of knowledge?

ZKPOK from Sigma Protocols

- **Is the previous protocol a proof of knowledge?**
  - It seems not to be
  - The extractor for the Sigma protocol needs to obtain two transcripts with the same **a** and different **e**
  - The prover may choose its first message **a** differently for every commitment string, so if the extractor changes **e**, the prover changes **a**

ZKPOK from Sigma Protocols

$\underline{\text{P}}$ (h,w) $\qquad\qquad\qquad\qquad$ $\underline{\text{V}}$ (h)

$$\xrightarrow{\quad a=g^r \quad}$$

$$\xleftarrow{\quad e \quad}$$

$$\xrightarrow{\quad z=r+ew \quad}$$

$$g^z \stackrel{?}{=} ah^e$$

ZKPOK from Sigma Protocols

- **Solution: use a trapdoor (equivocal) commitment scheme**
  - Given a trapdoor, it is possible to open the commitment to any value

ZKPOK from Sigma Protocols

- **Solution: use a trapdoor (equivocal) commitment scheme**
  - Given a trapdoor, it is possible to open the commitment to any value
- **Pedersen has this property, and the previous protocol can be modified only slightly to get a proof of knowledge**

Pedersen Commitments

- **Highly efficient perfectly-hiding commitments**
  - **Parameters:** generator $\boldsymbol{g}$, order $\boldsymbol{q}$
  - **Commit protocol** (commit to $x \in \mathbb{Z}_q$):
    - Receiver chooses random $k \leftarrow \mathbb{Z}_q$ and sends $h = g^k$
    - Sender sends $c = g^r \cdot h^x$, for a random $r \leftarrow \mathbb{Z}_q$
  - **Perfect hiding:**
    - For every $x, y \in \mathbb{Z}_q$ there exist $r, s \in \mathbb{Z}_q$ such that $r + kx = s + ky \bmod q$
  - **Computational binding:**
    - If can find $(x, r), (y, s)$ such that $g^r \cdot h^x = g^s \cdot h^y$ then can compute $k = DLOG_g(h) = {}^{r-s}/_{y-x} \bmod q$

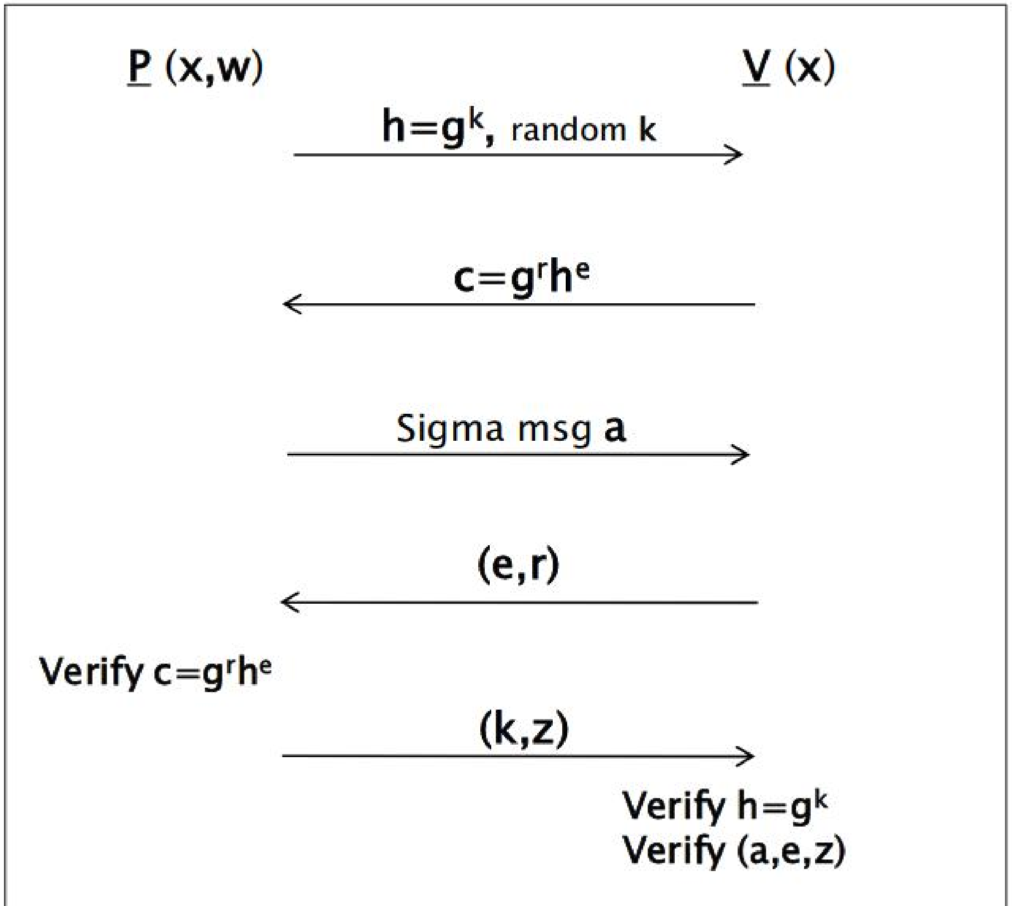
Pedersen has this property – given the discrete log k of h, can decommit to any value

- Commit to **x**: $c = g^r h^x$
- To decommit to **y**, find **s** such that $r+kx = s+ky$
- Compute $s = r+k(x-y) \bmod q$

ZKPOK from Sigma Protocols

- **The basic idea**
  - Have **V** first to its challenge **e** using a perfectly-hiding trapdoor (equivocal) commitment
- **The protocol**
  - **P** sends the 1st message α of the commit protocol
  - **V** sends a commitment c=$\mathbf{Com}_{\alpha}$(**e;r**)
  - **P** sends a message **a**
  - **V** sends (**e,r**)
  - **P** checks that c=$\mathbf{Com}_{\alpha}$(**e;r**) and if yes sends the **trapdoor** and **z**
  - **V** accepts if the **trapdoor** is correct and (**x,a,e,z**) is accepting

ZKPOK from Sigma Protocols



ZKPOK from Sigma Protocols

- **Why does this help?**
  - **Zero-knowledge** remains the same
  - **Extraction:** after verifying the proof once, the extractor obtains **k** and can rewind back to the decommitment of **c** and send any (**e′**,**r′**)

ZK and Sigma Protocols

- **We typically want zero knowledge, so why bother with sigma protocols?**
  - We have many useful general transformations
    - E.g., parallel composition, compound statements
    - The ZK and ZKPOK transformations can be applied on top of the above, so obtain transformed ZK
  - It is **much harder** to prove ZK than Sigma
    - ZK – distributions and simulation
    - Sigma: only HVZK and special oundness

Using Sigma Protocols and ZK

- **Prove that the El Gamal encryption (u,v) under public-key (g,h) is to the value m**
  - By encryption definition $\mathbf{u=g^r}$, $\mathbf{v=h^r \cdot m}$
  - ThUS (**g,h,u,v/m**) is a DH tuple
  - So, given (**g,h,u,v,m**), just prove that (**g,h,u,v/m**) is a DH tuple

Using Sigma Protocols and ZK

- **Prove that the El Gamal encryption (u,v) under public-key (g,h) is to the value m**
  - By encryption definition $\mathbf{u=g^r}$, $\mathbf{v=h^r \cdot m}$
  - ThUS (**g,h,u,v/m**) is a DH tuple
  - So, given (**g,h,u,v,m**), just prove that (**g,h,u,v/m**) is a DH tuple

- **Database of ElGamal(K_i),E_{Ki}(T_i)**
  - Can release $\mathbf{T_i}$ without revealing anything about $\mathbf{T_j}$ for $\mathbf{j \neq I}$

Non-Interactive ZK (ROM)

- **The Fiat-Shamir paradigm**
  - To prove a statement **x**
  - Generate **a**, compute **e=H**(**a**,**x**), compute **z**
  - Send (**a**,**e**,**z**)
- **Properties:**
  - **Soundness:** follows from random oracle property
  - **Zero knowledge:** same
  - Can achieve simulation-soundness (non malleability) by including unique **sid** in **H**

Acknowledge

- Some materials are extracted from the slides created by Prof. Bingsheng Zhang,Yehuda Lindell, and Qi Wang.