Cryptocomplexity II

Kryptokomplexität II

Sommersemester 2024

Chapter 5: Arthur-Merlin Games and Zero-Knowledge

Dozent: Prof. Dr. J. Rothe

What is Zero-Knowledge?

- "There are **known knowns**.
  These are things we know that we know.
- There are **known unknowns**.
  That is to say, there are things we know we don't know.
- But, there are also **unknown unknowns**.
  These are things we don't know we don't know."

And there is **zero-knowledge**.

These are things we know that somebody else knows, and we provably cannot know what they are.

Outline of this Chapter

- Interactive Proof: intuition
- Graph Isomorphism and Graph Automorphism: definition and properties
- Arthur-Merlin Games
- Interactive Proof Systems
- Zero-Knowledge Protocols

Interactive Proof: Historical Example

- What is a proof?
- Consider the theorem:
  The quadratic equation $x^2 + px + q = 0$ *has the two solutions*

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}\right)^2 - q}.$$

- Its proof consists in simply substituting each of x_1 and x_2 for x and checking if this evaluates to zero.
- However, up to the Middle Ages, solving *cubic* equations was a true challenge.

Interactive Proof: Historical Example

- Nicolò Tartaglia (1500–1557) found a solution formula for equations of the form $x^3 + px = q$, where p and q are positive:

$$x = \sqrt[3]{\sqrt{\left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2} + \frac{q}{2}} + \sqrt[3]{\sqrt{\left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2} - \frac{q}{2}}.$$

- However, he kept it secret.

- Scipione del Ferro (1465–1526) had discovered this formula even earlier, and kept it secret until just before his death when he gave it to his student Antonio Maria Fior.

Interactive Proof: Historical Example

- In a famous contest in 1535, and Fior and Tartaglia presented each other with 30 challenges of the form:
  - $x^3 + px = q$ for Tartaglia and
  - $x^3 + px^2 = q$ for Fior,

  which Tartaglia won.

- Later on, Tartaglia was persuaded by Gerolamo Cardano (1501–1576) to reveal his secret formula to him.
  He promised under oath to never publish it.

- However, Cardano did publish it in his book *Ars Magna* in 1545 when he learned that del Ferro discovered this formula even earlier.

Permutation Group

Definition

- A *permutation* is a bijective mapping of a set onto itself.
- The *set of all permutations of* $[n] = \{1, 2, \ldots, n\}$ is denoted by $\mathfrak{S}_n$.
- For algorithmic purposes, we represent permutations $\pi \in \mathfrak{S}_n$ as lists of n ordered pairs $(i, \pi(i))$ from $[n] \times [n]$.
- For π and τ in $\mathfrak{S}_n$, define their *composition* $\pi\tau$ to be the permutation in $\mathfrak{S}_n$ that results from first applying π and then applying τ to the elements from $[n]$, i.e., $(\pi\tau)(i) = \tau(\pi(i))$ for each $i \in [n]$.
- $\mathfrak{S}_n$ is said to be a *permutation group* with respect to the composition of permutations. Its neutral element is the *identical permutation*, defined as $\mathrm{id}(i) = i$ for each $i \in [n]$.

Graph Isomorphism and Graph Automorphism

Definition

Let G and H are graphs with the same number of vertices, and assume

$$V(G) = \{1, 2, \ldots, n\} = V(H).$$

- An *isomorphism between G and H* is an edge-preserving bijection from $V(G)$ onto $V(H)$.

  That is, G and H are *isomorphic* ($G \cong H$, for short) if there exists a permutation $\pi \in \mathfrak{S}_n$ such that for any two vertices $i, j \in V(G)$,

$$\{i, j\} \in E(G) \iff \{\pi(i), \pi(j)\} \in E(H). \quad (1)$$

Graph Isomorphism and Graph Automorphism

Definition

- An *automorphism of G* is an edge-preserving bijection from $V(G)$ onto itself. Every graph contains the trivial automorphism id.

- Denote the *set of all isomorphisms between G und H* by $\textsc{Iso}(G,H)$, and denote the *set of all automorphisms of G* by $\textsc{Aut}(G)$.

- Define the *graph isomorphism problem* (GI, for short) and the *graph automorphism problem* (GA, for short) by

$$\begin{aligned} \mathtt{GI} &= \{\langle G,H\rangle \,\big|\, G \text{ and } H \text{ are isomorphic graphs}\}; \\ \mathtt{GA} &= \{G \,\big|\, G \text{ contains a nontrivial automorphism}\}. \end{aligned}$$

Graph Isomorphism and Graph Automorphism

Example

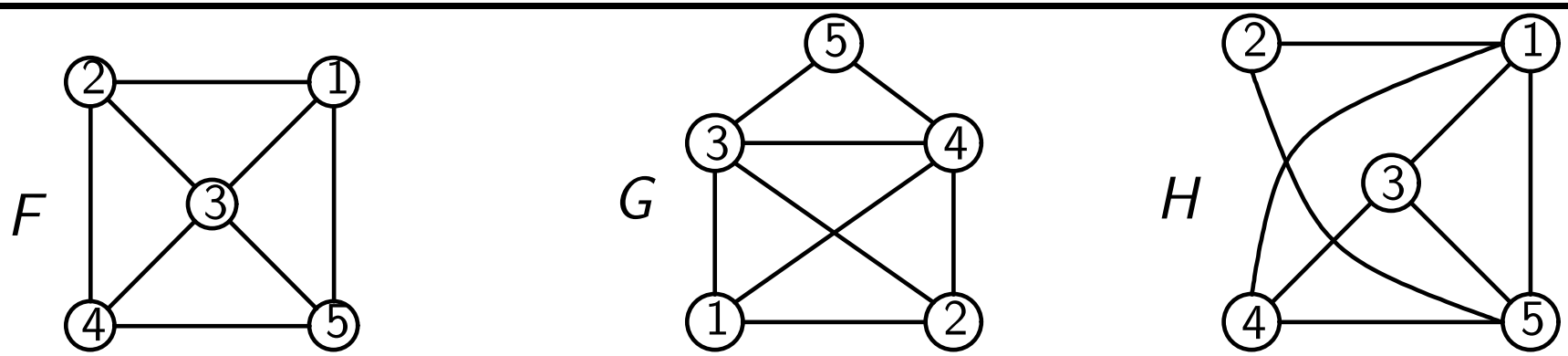


Figure: Three graphs: G is isomorphic to H, but not to F

- G and H above are isomorphic graphs.

  An isomorphism π between G and H is given by $\pi = \begin{pmatrix} 1\,2\,3\,4\,5 \\ 3\,4\,1\,5\,2 \end{pmatrix}$

  or, in cyclic notation, by $\pi = (13)(245)$.

Graph Isomorphism and Graph Automorphism

Example (continued)

- There are three more isomorphisms between G and H, i.e., $\|\textsc{Iso}(G,H)\| = 4$.
- However, neither G nor H is isomorphic to F. This is immediately clear if one looks at the *sequence of vertex degrees*.
- A nontrivial automorphism $\varphi : V(G) \to V(G)$ of G is given by $\varphi = \begin{pmatrix} 1\,2\,3\,4\,5 \\ 2\,1\,4\,3\,5 \end{pmatrix}$ or, in cyclic notation, by $\varphi = (12)(34)(5)$.
- There are two more nontrivial automorphisms of G, i.e., $\|\textsc{Aut}(G)\| = 4$.
- $\textsc{Aut}(F)$, $\textsc{Aut}(G)$, and $\textsc{Aut}(H)$ are subgroups of $\mathfrak{S}_5$.

Graph Isomorphism and Graph Automorphism

Lemma

For any two given graphs G and H, we have:

$$\|\textsc{Iso}(G,H)\| = \begin{cases} \|\textsc{Aut}(G)\| = \|\textsc{Aut}(H)\| & \text{if } G \cong H \\ 0 & \text{if } G \ncong H; \end{cases} \tag{2}$$

$$\|\textsc{Aut}(G \cup H)\| = \begin{cases} 2 \cdot \|\textsc{Aut}(G)\| \cdot \|\textsc{Aut}(H)\| & \text{if } G \cong H \\ \|\textsc{Aut}(G)\| \cdot \|\textsc{Aut}(H)\| & \text{if } G \ncong H. \end{cases} \tag{3}$$

Graph Isomorphism and Graph Automorphism

Proof: $\textsc{Iso}(G,H)$ and $\textsc{Aut}(G)$ have equal size if and only if G and H are isomorphic.

This is true, since if G and H are isomorphic, then

$$\textsc{Aut}(G) = \textsc{Iso}(G,G) \quad \text{implies} \quad \|\textsc{Iso}(G,H)\| = \|\textsc{Aut}(G)\|.$$

Otherwise, if $G \ncong H$, then $\textsc{Iso}(G,H)$ is empty, whereas $\textsc{Aut}(G)$ always contains the trivial automorphism id.

This implies assertion (2) of the lemma.

Graph Isomorphism and Graph Automorphism

For proving the assertion (3), we assume that G and H are connected; otherwise, we consider the complementary graphs $\overline{G}$ and $\overline{H}$ in place of G and H (exercise).

An automorphism of $G \cup H$ exchanging the vertices of G and H is composed of an isomorphism in $\text{ISO}(G, H)$ and an isomorphism in $\text{ISO}(H, G)$. Thus,

$$\|\text{AUT}(G \cup H)\| = \|\text{AUT}(G)\| \cdot \|\text{AUT}(H)\| + \|\text{ISO}(G, H)\|^2,$$

which implies assertion (3) via (2). ❑

Graph Isomorphism and Graph Automorphism

Definition

Let $\mathfrak{G}$ and $\mathfrak{H}$ be permutation groups, where $\mathfrak{H}$ is a subgroup of $\mathfrak{G}$.

For $\tau \in \mathfrak{G}$, the *right co-set of $\mathfrak{H}$ in $\mathfrak{G}$* is defined by

$$\mathfrak{H}\tau = \{\pi\tau \,|\, \pi \in \mathfrak{H}\}.$$

- Any two right co-sets of $\mathfrak{H}$ in $\mathfrak{G}$ are either identical or disjoint: The permutation group $\mathfrak{G}$ can be partitioned into right co-sets of $\mathfrak{H}$ in $\mathfrak{G}$:

$$\mathfrak{G} = \mathfrak{H}\tau_1 \cup \mathfrak{H}\tau_2 \cup \cdots \cup \mathfrak{H}\tau_k. \tag{4}$$

- Every right co-set of $\mathfrak{H}$ in $\mathfrak{G}$ has cardinality $\|\mathfrak{H}\|$. The set $\{\tau_1, \ldots, \tau_k\}$ from (4) is called the *complete right transversal of $\mathfrak{H}$ in $\mathfrak{G}$*.

Graph Isomorphism and Graph Automorphism

- If G and H are isomorphic graphs and τ is an isomorphism in $\textsc{Iso}(G,H)$, then $\textsc{Iso}(G,H) = \textsc{Aut}(G)\tau$.

  That is, $\textsc{Iso}(G,H)$ is a right co-set of $\textsc{Aut}(G)$ in $\mathfrak{S}_n$.

- Since any two right co-sets are either disjoint or equal, $\mathfrak{S}_n$ can be partitioned into right co-sets of $\textsc{Aut}(G)$ according to (4):

$$\mathfrak{S}_n = \textsc{Aut}(G)\tau_1 \cup \textsc{Aut}(G)\tau_2 \cup \cdots \cup \textsc{Aut}(G)\tau_k, \tag{5}$$

  where $\|\textsc{Aut}(G)\tau_i\| = \|\textsc{Aut}(G)\|$ for each i, $1 \leq i \leq k$.

- Thus, this set $\{\tau_1, \tau_2, \ldots, \tau_k\}$ of permutations in $\mathfrak{S}_n$ is a complete right transversal of $\textsc{Aut}(G)$ in $\mathfrak{S}_n$.

Graph Isomorphism and Graph Automorphism

- Denoting by $\pi(G)$ the graph H that is obtained by applying the permutation $\pi \in \mathfrak{S}_n$ to the vertices of G, and noting that $H \cong G$, it follows that

$$\{\tau_i(G) \,\big|\, 1 \leq i \leq k\} = \{H \,\big|\, H \cong G\}.$$

- Since there are exactly $n! = n(n-1)\cdots 2 \cdot 1$ permutations in $\mathfrak{S}_n$,

$$\|\{H \,\big|\, H \cong G\}\| = k = \frac{\|\mathfrak{S}_n\|}{\|\mathrm{AUT}(G)\|} = \frac{n!}{\|\mathrm{AUT}(G)\|}$$

follows from (5). This proves the following corollary.

Corollary

To any graph G with n vertices, $\frac{n!}{\|\mathrm{AUT}(G)\|}$ graphs are isomorphic.

Graph Isomorphism and Graph Automorphism

Define the set

$$\begin{aligned} A(G_1, G_2) &= \{\langle H, \varphi\rangle \,|\, H \cong G_1 \wedge \varphi \in \textsc{Aut}(H)\} \cup \\ &\quad \{\langle H, \varphi\rangle \,|\, H \cong G_2 \wedge \varphi \in \textsc{Aut}(H)\}. \end{aligned}$$

Lemma

For any two given graphs G_1 and G_2 with n vertices each, we have

$$\|A(G_1, G_2)\| = \begin{cases} n! & \text{if } G_1 \cong G_2 \\ 2n! & \text{if } G_1 \ncong G_2. \end{cases}$$

Graph Isomorphism and Graph Automorphism

Proof: If F and G are isomorphic, then $\|\text{AUT}(F)\| = \|\text{AUT}(G)\|$ implies

$$\|\{\langle F, \varphi\rangle \,\big|\, F \cong G \wedge \varphi \in \text{AUT}(F)\}\| = \frac{n!}{\|\text{AUT}(F)\|} \cdot \|\text{AUT}(F)\| = n!$$

by the previous corollary.

Analogously, $\|\{\langle F, \varphi\rangle \,\big|\, F \cong H \wedge \varphi \in \text{AUT}(F)\}\| = n!$.

- If G and H are isomorphic, then the sets
  $\{\langle F, \varphi\rangle \,\big|\, F \cong G \wedge \varphi \in \text{AUT}(F)\}$ and $\{\langle F, \varphi\rangle \,\big|\, F \cong H \wedge \varphi \in \text{AUT}(F)\}$
  are equal, which implies $\|A(G, H)\| = n!$.
- If G and H are nonisomorphic, then the sets
  $\{\langle F, \varphi\rangle \,\big|\, F \cong G \wedge \varphi \in \text{AUT}(F)\}$ and $\{\langle F, \varphi\rangle \,\big|\, F \cong H \wedge \varphi \in \text{AUT}(F)\}$
  are disjoint. Hence, $\|A(G, H)\| = 2n!$. ❑

Arthur-Merlin Games

- Merlin and Arthur play the following game.
  The goal of the game is for them to jointly solve some problem.

- Suppose they are trying to solve the *graph nonisomorphism problem* GNI: Given a pair of graphs, G and H, they thus want to decide whether or not G and H are *nonisomorphic*.

- They draw one after the other taking turns, and they gamble for every problem instance.

Arthur-Merlin Games

- Merlin's intention always is to convince Arthur that the given graphs indeed are nonisomorphic, even if in fact they are isomorphic.
  So, one move of Merlin is to present a "proof" that the given graphs are nonisomorphic.
- However, Arthur is suspicious and does not trust the sneaky wizard. Of course, he himself cannot come up with such powerful proofs of his own. After all, Merlin has supernatural, nondeterministic powers and spells at his disposal, and Arthur does not.
  Still, Arthur doubts that the proofs are valid, flips some coins and, depending on these random choices, he challenges the wizard's proofs. Such is one of Arthur's moves in this game.
- Again, it's Merlin's turn to move, and so on.

Arthur-Merlin Games

- Eventually, after a finite number of moves, they will have determined whether or not to accept the input.
  It is also possible that Arthur makes the first move in a game.

- Suppose that
  - Merlin is represented by an NP machine M, and
  - Arthur is represented by a randomized polynomial-time bounded Turing machine A.

- Let x be the problem instance at stake, and let L be the problem they want to solve.

Arthur-Merlin Games and the Arthur-Merlin Hierarchy

- One move of Merlin is a proof for "$x \in L$," and he can find such a proof by simulating $M(x, y)$, where y encodes the history of moves made as yet.
  That is, the string y describes all nondeterministic choices of Merlin and all random choices of Arthur previously made in this game.

- In order to satisfy the impatient king, Merlin must convince him with overwhelming probability.

- One move of Arthur is given by the computation of $A(x, y)$ that depends on Arthur's random choices, where again y encodes the previous history of the game.

Quantifier Representation of Complexity Classes

We express this intuition by the following quantifier representation of complexity classes.

Definition

1. Let B be a predicate, and let p be a given polynomial.
   For each fixed string x, define
   - $(\exists^p y)[B(x,y)] \iff$ there is some y, $|y| \leq p(|x|)$, such that $B(x,y)$.
   - $(\forall^p y)[B(x,y)] \iff$ for all y, $|y| \leq p(|x|)$, $B(x,y)$.
   - $(\exists^{+p} y)[B(x,y)] \iff$ at least three-quarters of all strings y with $|y| \leq p(|x|)$ satisfy $B(x,y)$.

   When clear from the context, we omit the superscript "p" and write $\exists y$, $\forall y$, and $\exists^+ y$ instead of $\exists^p y$, $\forall^p y$, and $\exists^{+p} y$.

Quantifier Representation of Complexity Classes

Definition

2. Let $\mathfrak{Q}_1$ and $\mathfrak{Q}_2$ be two strings of n quantifiers each. The pair $(\mathfrak{Q}_1, \mathfrak{Q}_2)$ is *sensible* if and only if for each $(n+1)$-ary predicate B, for each x, and for each $\vec{y} = (y_1, y_2, \ldots, y_n)$,

$$(\mathfrak{Q}_1 \vec{y})\,[B(x, \vec{y})] \wedge (\mathfrak{Q}_2 \vec{y})\,[\neg B(x, \vec{y})]$$

is a contradiction. Here, y_i is the variable quantified by the i^{th} quantifier in $\mathfrak{Q}_1$ and $\mathfrak{Q}_2$, respectively.

Quantifier Representation of Complexity Classes

Definition

3. Let $(\mathfrak{Q}_1, \mathfrak{Q}_2)$ be a sensible pair of strings consisting of n (polynomially length-bounded) quantifiers each. Define the complexity class $(\mathfrak{Q}_1 \,|\, \mathfrak{Q}_2)$ as follows: L belongs to $(\mathfrak{Q}_1 \,|\, \mathfrak{Q}_2)$ if and only if there exists an $(n+1)$-ary predicate $B \in \mathrm{P}$ such that for each $x \in \Sigma^*$:

$$\begin{aligned} x \in L &\implies (\mathfrak{Q}_1 \vec{y})\,[B(x, \vec{y})]; \\ x \notin L &\implies (\mathfrak{Q}_2 \vec{y})\,[\neg B(x, \vec{y})], \end{aligned}$$

where $\vec{y} = (y_1, y_2, \ldots, y_n)$ and y_i is the variable quantified by the i^{th} quantifier in $\mathfrak{Q}_1$ and $\mathfrak{Q}_2$, respectively, and $|y_i| \leq p(|x|)$ for some suitable polynomial p.

Arthur-Merlin Games and the Arthur-Merlin Hierarchy

Definition (Arthur-Merlin Hierarchy (Babai and Moran, 1988))

The *levels of the Arthur-Merlin hierarchy* are the following classes:

$$\mathrm{A} = (\exists^+|\exists^+), \quad \mathrm{AM} = (\exists^+\exists|\exists^+\forall), \quad \mathrm{AMA} = (\exists^+\exists\exists^+|\exists^+\forall\exists^+),$$
$$\mathrm{M} = (\exists|\forall), \quad \mathrm{MA} = (\exists\exists^+|\forall\exists^+), \quad \mathrm{MAM} = (\exists\exists^+\exists|\forall\exists^+\forall), \ldots$$

Define the *Arthur-Merlin hierarchy*, AMH, as the union of all these classes.

Arthur-Merlin Games and the Arthur-Merlin Hierarchy

Example (Arthur-Merlin Hierarchy)

Consider the class MA, which consists of precisely those problems L for which there exist an NPTM M and a randomized polynomial-time Turing machine A such that for each input x:

- If $x \in L$, then there exists a path y of $M(x)$ such that $A(x, y)$ accepts with probability at least $3/4$. That is, Arthur cannot refute Merlin's correct proof y for "$x \in L$," and Merlin wins.
- If $x \notin L$, then for each path y of $M(x)$, $A(x, y)$ rejects with probability at least $3/4$. That is, Arthur cannot be fooled by Merlin's false proofs for "$x \in L$" and thus wins.

Analogously, the classes AM, MAM, AMA, ... can be described.

Relation to Interactive Proof Systems

Definition (Goldwasser, Micali, and Rackoff, 1989)

1. An *interactive proof system* (a.k.a. *IP protocol*) is a pair (V, P), where
   - V, the verifier, is a randomized polynomial-time Turing machine and
   - P, the prover, is of unbounded power (sometimes: "an NP machine")

   communicating over a joint tape. P does not "see" V's random bits.

2. (V, P) accepts a language L if for each $x \in \Sigma^*$,
   - $x \in L \implies (\exists P)[\Pr((V, P) \text{ accepts } x) \geq 3/4]$;
   - $x \notin L \implies (\forall \hat{P})[\Pr((V, \hat{P}) \text{ accepts } x) \leq 1/2]$,

   where the "$x \in L$" case quantifies over all possible prover strategies of P, and the "$x \notin L$" case quantifies over all possible provers $\hat{P}$.

3. IP denotes the class of all languages acceptable by an IP protocol.

Relation to Interactive Proof Systems

- Arthur-Merlin games were introduced by Babai and Moran (1988).
- Independently, Goldwasser, Micali und Rackoff (1989) developed the theory of *interactive proof systems* that yields an essentially equivalent concept. A difference is that
  - Arthur's random bits are public, whereas
  - the verifier's random bits are private.

  However, Goldwasser and Sipser (1989) showed that in fact it does not matter whether one uses private or public coins.
- The probabilities in the definition are chosen at will: For $\varepsilon > 0$,
  - $1/2 + \varepsilon$ (for acceptance) and
  - $1/2 - \varepsilon$ (for rejection) would work as well.

  For acceptance, one could even require probability 1.
- If the verifier were deterministic, the IP definition would just yield NP.

Authentication Protocol for Arthur and Merlin

Arthur and Merlin again play one of their games.

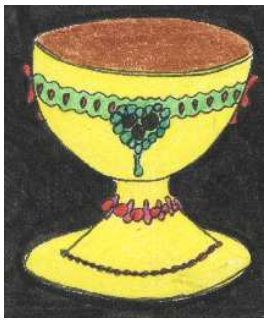
This time, Arthur wishes to verify Merlin's identity, as he is uncertain of whether he is talking with Merlin or with some other wizard who merely pretends to be Merlin.

To verify Merlin's identity, Arthur challenges him for a proof of his secret, a magic spell that puts a dangerous, fire-breathing dragon to sleep.

Merlin alone knows this spell.

The dragon lives in a secret, subterranean labyrinth (shown on the next slide) that may be entered only by Arthur's permission.

Authentication Protocol for Arthur and Merlin



Dragon

The Holy Grail

The One Ring

left entrance

Saruman

right entrance

main entrance

Arthur

Authentication Protocol for Arthur and Merlin

The dragon sits there right in the middle between the Holy Grail and the One Ring That Rules Them All.

So, if the labyrinth is entered through the left entrance and the dragon is awake, one can get only the Holy Grail and not the Ruling Ring.

If the labyrinth is entered through the right entrance and the dragon is awake, one can get only the Ruling Ring and not the Holy Grail.

The only way from the Holy Grail to the Ruling Ring or vice versa passes by the dragon and can be used only if the dragon sleeps.

(As you may know, dragons never sleep, except when they are compelled to by a magic spell.)

Authentication Protocol for Arthur and Merlin

The Arthur-Merlin game is as follows.

First, Merlin enters through the main entrance, closes this door, and chooses either the left or the right entrance to the labyrinth.

Arthur now follows him through the main entrance and does not know whether Merlin has used the left or the right entrance.

He challenges Merlin by requesting to see either the Holy Grail or the Ruling Ring.

If Merlin has used the left entrance and Arthur wishes to see the Holy Grail, Merlin just takes it and leaves the labyrinth.

Similarly, Merlin has no problem to authenticate himself if he has used the right entrance and Arthur requests to see the Ruling Ring.

Authentication Protocol for Arthur and Merlin

On the other hand,

- if Merlin has used the left entrance and Arthur requests to see the Ruling Ring, or
- if Merlin has entered through the right entrance and Arthur wishes to see the Holy grail,

Merlin (and Merlin alone) is able to authenticate himself by using his magic spell to put the dragon to sleep.

Meanwhile, Saruman the White, the malicious wizard of Orthanc, has also reached Camelot.

He has had a hard time in Orthanc lately, but he managed to escape from there.

Authentication Protocol for Arthur and Merlin

Now, after having suffered defeat by Gandalf, he is out for revenge and needs the power of the Ruling Ring more than ever.

He has heard rumors that Arthur keeps it in his secret labyrinth.

Using his magic, Saruman therefore appears disguised as Merlin in Camelot and requests entry to the hidden labyrinth.

He is much less powerful a wizard than Merlin and, sure enough, he does not know his secret spells.

In fact, Saruman's magic power can bring about no more than the computing power of a polynomial-time randomized Turing machine.

Still, he wishes to steal the Ruling Ring and therefore he pretends to know Merlin's secret.

Authentication Protocol for Arthur and Merlin

When Saruman chooses the left or the right entrance to the labyrinth, he does not know Arthur's challenge in advance.

Thus, all he can do is toss a coin.

If the outcome (heads for left and tails for right, say) is in his favor, he uses the entrance corresponding to Arthur's subsequent request and succeeds.

In this case, Arthur is taken in by him.

However, if Saruman's random choice is unlucky, he cannot fool Arthur.

Authentication Protocol for Arthur and Merlin

For example, if Saruman enters the labyrinth through the right entrance but Arthur requests to see the Holy Grail, then Saruman loses.

He cannot pass the dragon, since he does not know Merlin's secret spell for putting the dragon to sleep.

By repeatedly challenging Saruman for a proof of Merlin's secret, Arthur will detect the attempted fraud with high probability.

The first time Saruman fails to present the Holy Grail or the Ruling Ring, Arthur knows for sure that he is not Merlin.

Goldreich, Micali, and Wigderson's IP Protocol for GI

Step	Merlin	Saruman	Arthur
1	chooses a large graph G_0 with n vertices and a secret $\pi \in \mathfrak{S}_n$ at random, computes $G_1 = \pi(G_0)$; (G_0, G_1) is public		
2		$(G_0, G_1) \Rightarrow$	
3	chooses a permutation $\mu \in \mathfrak{S}_n$ and a bit $m \in \{0,1\}$ at random, and computes $H = \mu(G_m)$		
4		$H \Rightarrow$	
5			chooses a bit $a \in \{0,1\}$ at random and requests an α in $\mathrm{ISO}(G_a, H)$

Goldreich, Micali, and Wigderson's IP Protocol for GI

Step	Merlin	Saruman	Arthur
6		$\Leftarrow a$	
7	computes $\alpha \in \mathrm{Iso}(G_a, H)$ by: if $a = m$ then $\alpha = \mu$; if $0 = a \neq m = 1$ then $\alpha = \pi\mu$; if $1 = a \neq m = 0$ then $\alpha = \pi^{-1}\mu$		
8		$\alpha \Rightarrow$	
9			verifies that $\alpha(G_a) = H$ and accepts accordingly

Table: IP protocol for graph isomorphism

Goldreich, Micali, and Wigderson's IP Protocol for GI

Theorem

The Goldreich–Micali–Wigderson protocol is an IP protocol for GI*.*

It can be used as a challenge-and-response authentication protocol.

Proof: As in the previous story, Merlin wants to authenticate himself by proving knowledge of his secret.

Merlin has the power of an NP machine, and his secret is the isomorphism between two large isomorphic graphs.

Since GI is not known to be polynomial-time solvable, not even via randomized algorithms, it may be assumed that neither Arthur nor the fraudulent wizard Saruman are able to discover Merlin's secret.

Goldreich, Micali, and Wigderson's IP Protocol for GI

Merlin can easily create his secret, and he does not even need his full NP power:

- He first chooses a large graph G_0 with n vertices and a permutation $\pi \in \mathfrak{S}_n$ at random.
- Then, he computes the graph $G_1 = \pi(G_0)$.

That is, G_0 and G_1 are isomorphic graphs and π is an isomorphism between them.

Merlin makes the pair (G_0, G_1) public, and he keeps the isomorphism $\pi \in \mathrm{ISO}(G_0, G_1)$ secret.

Suppose that Gandalf, a trusted third party, certifies that (G_0, G_1) indeed was created by Merlin.

Goldreich, Micali, and Wigderson's IP Protocol for GI

Sure enough, Merlin cannot just send π to Arthur because then he would have given his secret away.

Rather, to prove that G_0 and G_1 indeed are isomorphic, Merlin randomly chooses a permutation $\mu \in \mathfrak{S}_n$ and a bit $m \in \{0,1\}$ under the uniform distribution, and he computes the graph $H = \mu(G_m)$.

That is, m determines which of G_0 or G_1 is to be permuted by μ to yield H, a graph isomorphic to G_m via μ.

Merlin sends H to Arthur, who chooses a random bit $a \in \{0,1\}$ under the uniform distribution.

Arthur then sends a to Merlin as his challenge, requesting Merlin to respond with an isomorphism α between G_a and H.

Goldreich, Micali, and Wigderson's IP Protocol for GI

Arthur accepts Merlin's response α if and only if $\alpha(G_a) = H$.

This protocol works, since Merlin knows:

- his secret isomorphism $\pi \in \mathrm{Iso}(G_0, G_1)$,
- his random bit $m \in \{0,1\}$, and
- his random permutation $\mu \in \mathrm{Iso}(G_m, H)$.

Thus, he can easily determine an isomorphism $\alpha \in \mathrm{Iso}(G_a, H)$, which he uses for authentication.

Since G_0 and G_1 are isomorphic, Arthur accepts with probability one.

Remark: The case of nonisomorphic graphs does not occur in this protocol. But it can be modified so that Arthur and Merlin decide the problem GI: (G_0, G_1) is their input then and not chosen by Merlin. If $G_0 \not\cong G_1$, Arthur rejects Merlin's false proof with probability $1/2$.

Goldreich, Micali, and Wigderson's IP Protocol for GI

Now, suppose that Saruman, disguised as Merlin, executes the protocol with Arthur.

Saruman knows the public pair (G_0, G_1) of isomorphic graphs but not Merlin's isomorphism π, which is kept secret during the protocol.

Still, he wishes to pretend to be Merlin.

He randomly chooses a permutation $\sigma \in \mathfrak{S}_n$ and his bit $s \in \{0,1\}$, and computes the graph $H_S = \sigma(G_s)$.

Then, Saruman sends H_S to Arthur and receives Arthur's challenge a.

If he is lucky and $s = a$, then Saruman wins.

However, if $s \neq a$, the computation of $\alpha = \pi\sigma$ or $\alpha = \pi^{-1}\sigma$ would require knowledge of π.

Goldreich, Micali, and Wigderson's IP Protocol for GI

Since computing an isomorphism is too hard even for randomized polynomial-time algorithms, Saruman cannot determine π if the graphs G_0 and G_1 are chosen large enough.

Without knowing π, he can only guess.

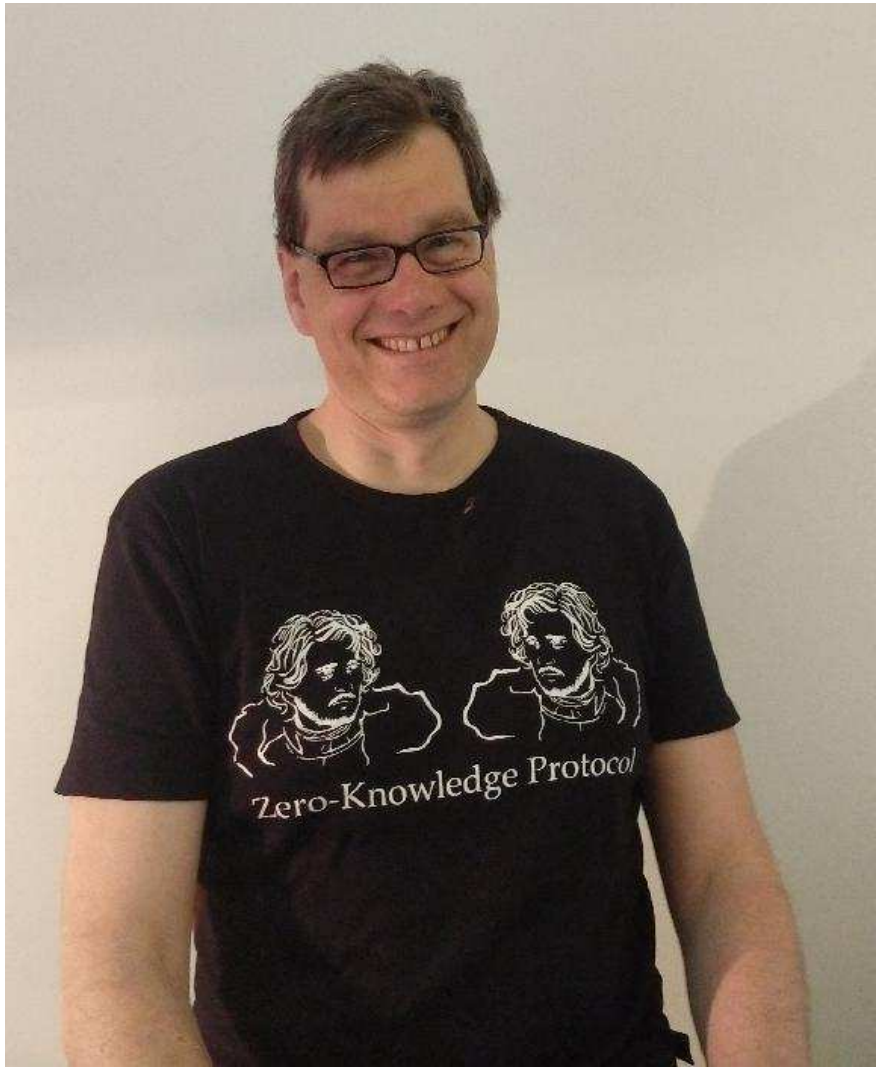
His chances of hitting a bit s with $s = a$ are at most $1/2$.

Of course, Saruman can always guess and thus his probability of success is exactly $1/2$.

If Arthur challenges him sufficiently often, say in k independent rounds of this protocol, the cheating probability can be made as small as 2^{-k}.

Already for $k = 20$, this probability is negligible: Saruman's probability of success is then less than one in one million. ❑

And Why Is It a Zero-Knowledge Protocol?



Goldreich, Micali, and Wigderson's ZK Protocol for GI

Definition

Let $L \in \mathrm{IP}$ be accepted by some IP protocol (V, P). We say that (V, P) is a *zero-knowledge protocol for L* if there exists a polynomial-time randomized Turing machine S, called the simulator, such that

- (V, S) simulates the original protocol (V, P), and
- for each $x \in L$, the tuples $(m_1, m_2, \ldots, m_k)$ and $(s_1, s_2, \ldots, s_k)$ that describe the information conveyed in (V, P) and in (V, S) are identically distributed over the coin tosses in (V, P) and in (V, S).

Theorem

The Goldreich–Micali–Wigderson protocol for GI *is zero-knowledge.*

Goldreich, Micali, and Wigderson's ZK Protocol for GI

Proof: Consider the simulated protocol with Saruman taking Merlin's place:

Step	Saruman		Arthur
1 & 2	**Merlin**'s pair (G_0, G_1) of isomorphic graphs is public information		
3	chooses a permutation $\sigma \in \mathfrak{S}_n$ and a bit $s \in \{0,1\}$ at random, and computes $H = \sigma(G_s)$		
4		$H \Rightarrow$	
5			chooses a bit $a \in \{0,1\}$ at random and requests an isomorphism in $\mathrm{ISO}(G_a, H)$

Table: Simulation of the zero-knowledge protocol for graph isomorphism

Goldreich, Micali, and Wigderson's ZK Protocol for GI

Step	Saruman		Arthur
6		$\Leftarrow a$	
7	if $a = s$, Saruman sends $\alpha = \sigma$; if $a \neq s$, he deletes this round		
8		$\alpha \Rightarrow$	
9			$a = s$ implies $\alpha(G_a) = H$, thus Arthur accepts Saruman's false identity

Table: Simulation of the ZK protocol for graph isomorphism (continued)

Goldreich, Micali, and Wigderson's ZK Protocol for GI

In the simulated protocol, Saruman does not know Merlin's secret isomorphism π, but he pretends to know it.

Let us assume that Arthur and Saruman execute a number of rounds of the protocol, always using the same pair (G_0, G_1) of isomorphic graphs, which is Merlin's public information.

That is, steps 1 and 2 are skipped.

Thus, the information conveyed in one round of the protocol has the form of a triple, (H, a, α).

Whenever Saruman happens to choose a random bit s with $s = a$, he simply sends $\alpha = \sigma$ to Arthur and wins: Arthur must accept him as Merlin.

Goldreich, Micali, and Wigderson's ZK Protocol for GI

On the other hand, if $s \neq a$ then Saruman cannot fool Arthur and fails.

However, that is no problem for them.

They simply delete this round from the protocol and restart.

In this way, they can produce a sequence of triples of the form (H, a, α) that are indistinguishable from the corresponding sequence of triples in the original protocol.

It follows that the Goldreich–Micali–Wigderson protocol has the zero-knowledge property. ❑