Introduction to Zero Knowledge Proofs

CS 598 DH

Today's objectives

Introduce Zero Knowledge Proofs

See a feasibility result for ZK of arbitrary statements in NP

Discuss how to upgrade semi-honest protocols to malicious

Setting

Semi-honest Security

Malicious Security

General-Purpose Tools

GMW Protocol

- Multi-party
- Multi-round

Garbled Circuit

- Constant Round
- Two Party

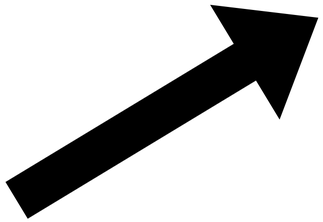
Primitives

Oblivious Transfer

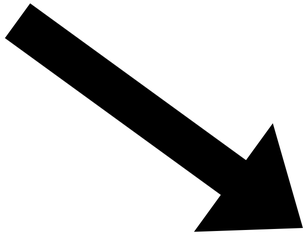
Pseudorandom functions/encryption

Commitments

Zero Knowledge Proofs



One way to achieve malicious security



Fascinating and useful in their own right

What is a proof?

What is a proof?

A convincing argument that a statement is true

What is a proof?

A convincing argument that a statement is true

An *interaction* between:

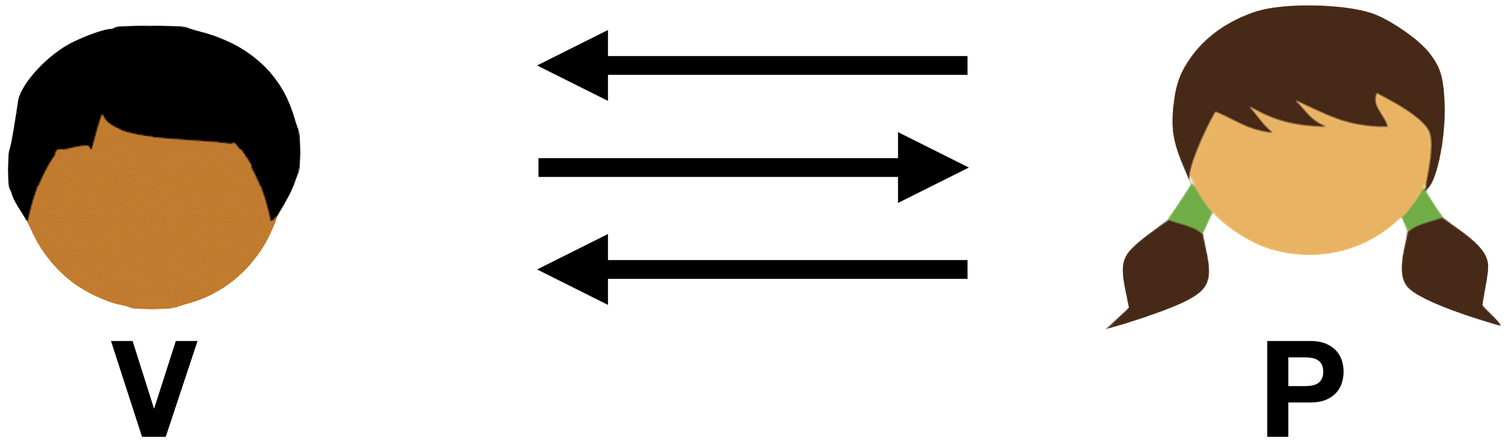
- The individual who is arguing
- The individual being convinced

What is a proof?

A convincing argument that a statement is true

An *interaction* between:

- The individual who is arguing **The Prover**
- The individual being convinced **The Verifier**

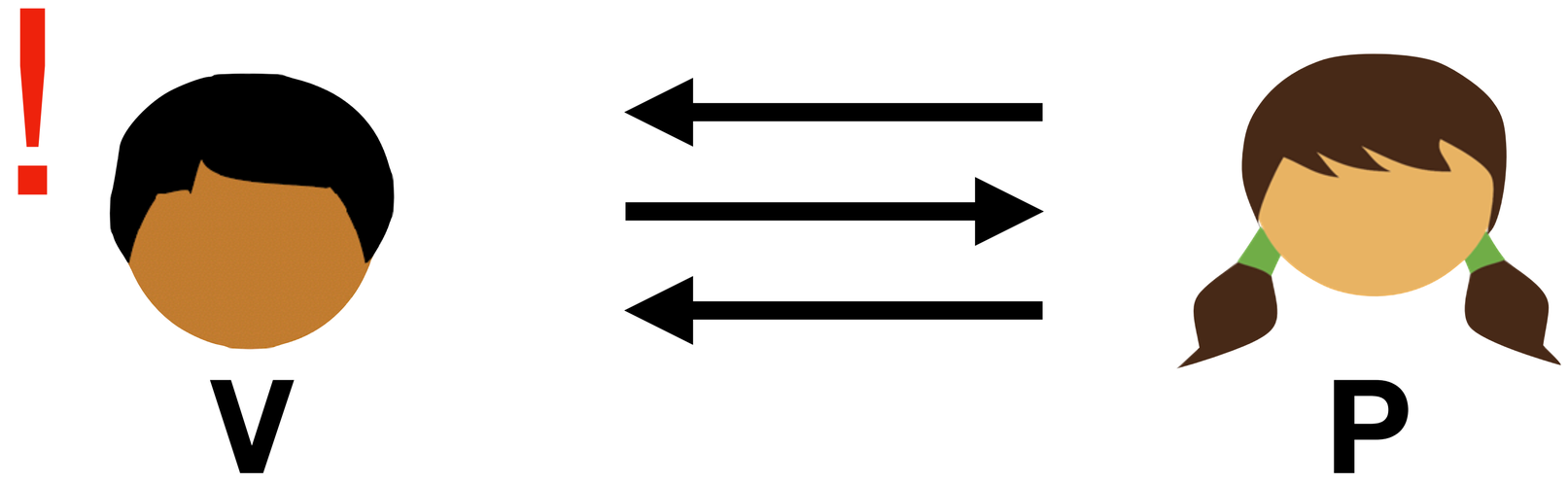


What is a proof?

A convincing argument that a statement is true

An *interaction* between:

- The individual who is arguing **The Prover**
- The individual being convinced **The Verifier**



What is a *zero-knowledge* proof?

A convincing argument that a statement is true

What is a *zero-knowledge* proof?

A convincing argument that a statement is true

Where the verifier learns nothing except that the statement is true

What is a *zero-knowledge* proof?

A convincing argument that a statement is true

Where the verifier learns nothing except that the statement is true

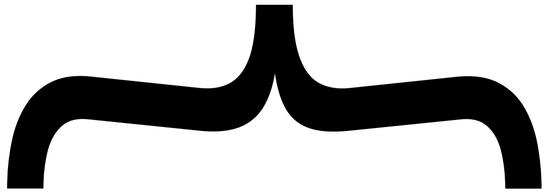
Any side information the Prover uses to show that the statement is true remains hidden

What is a *zero-knowledge* proof?

A convincing argument that a statement is true

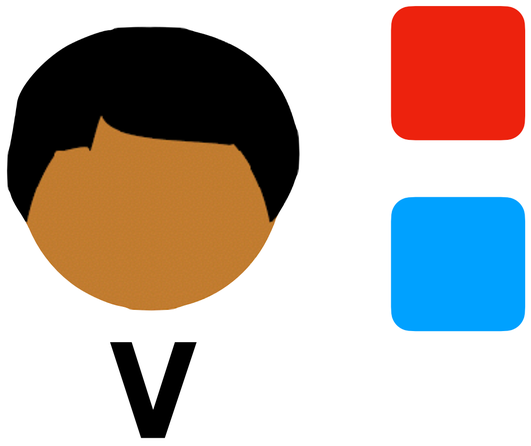
Where the verifier learns nothing except that the statement is true

The witness

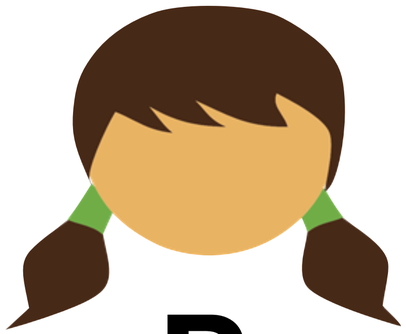


Any side information the Prover uses to show that the statement is true remains hidden

What is a *zero-knowledge* proof?

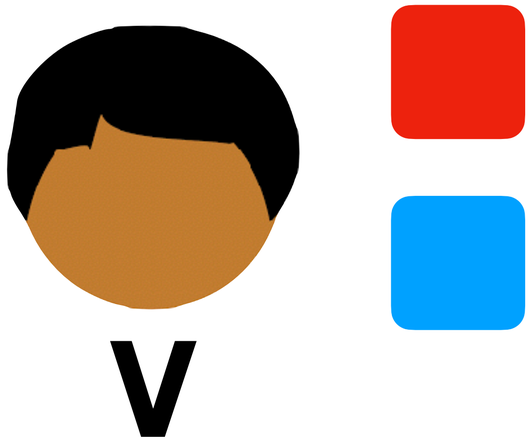


V



P

What is a *zero-knowledge* proof?



V

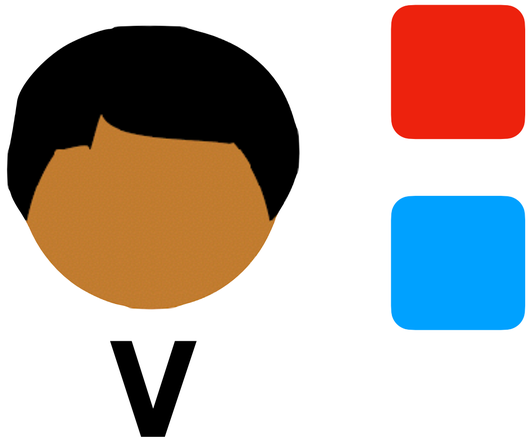
P

Suppose V is colorblind

Can P prove to V she can distinguish the colors of the blocks

without telling Bob which block is which?

What is a *zero-knowledge* proof?



V

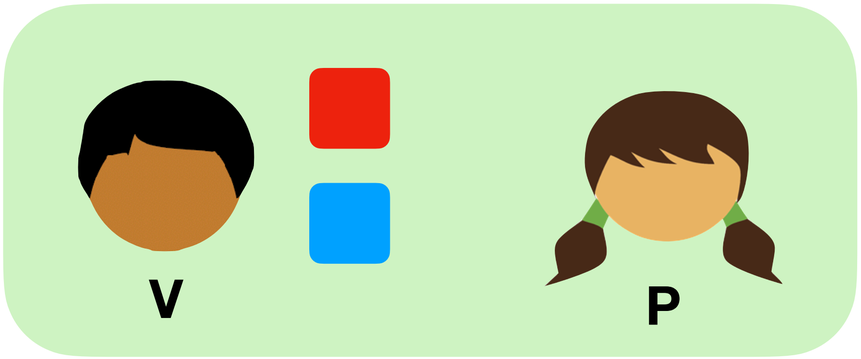
P

Suppose V is colorblind

Can P prove to V she can distinguish the blocks

without telling V which block is which?

What is a *zero-knowledge* proof?

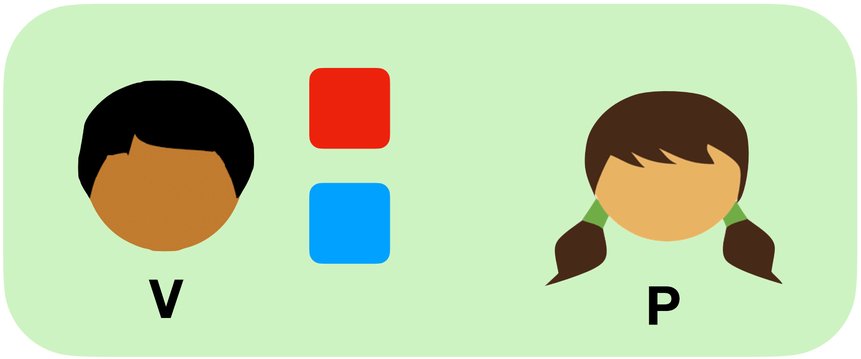


Statement: The formal proposition being proved “I can distinguish these blocks”

Witness: “side information” used to prove the statement

Alice can see colors of blocks

What is a *zero-knowledge* proof?

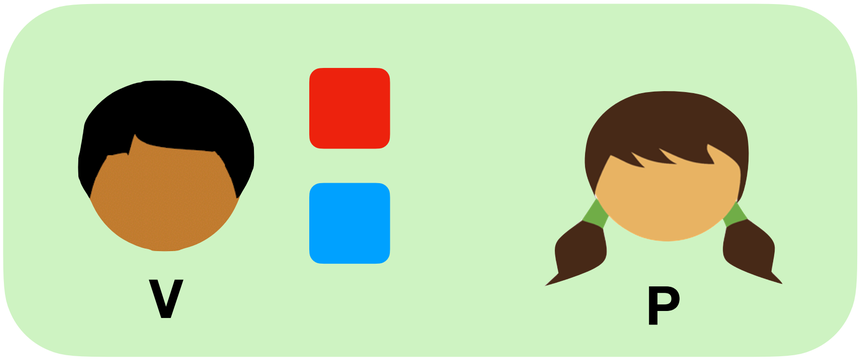


Statement: The formal proposition being proved “I can distinguish these blocks”

Witness: “side information” used to prove the statement
Alice can see colors of blocks

Proof System: An interaction allowing P to prove certain kinds of statements

What is a *zero-knowledge* proof?



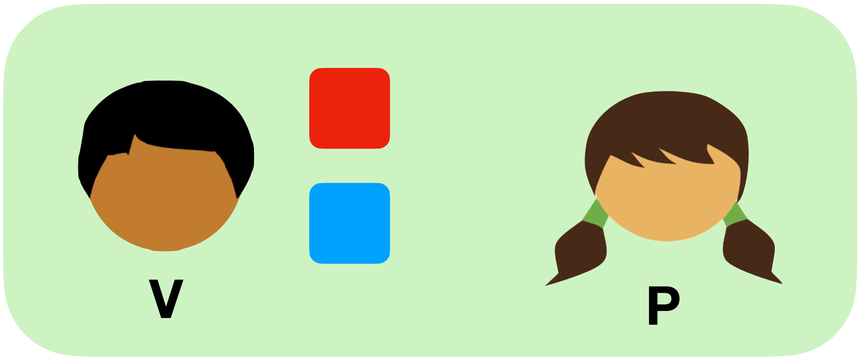
Statement: The formal proposition being proved “I can distinguish these blocks”

Witness: “side information” used to prove the statement
Alice can see colors of blocks

Proof System: An interaction allowing P to prove certain kinds of statements

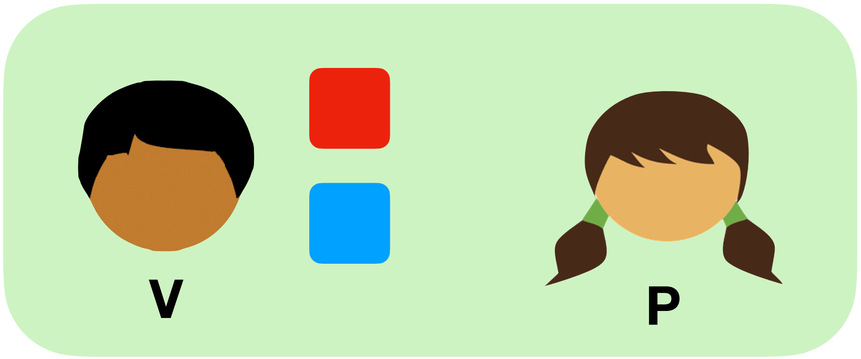
Formally, P proves that the statement x is in a *language* $\mathscr{L}$

What is a *zero-knowledge* proof?



Completeness: If $x \in \mathscr{L}$ and if P and V are honest, then V accepts the proof (except with negligible probability)

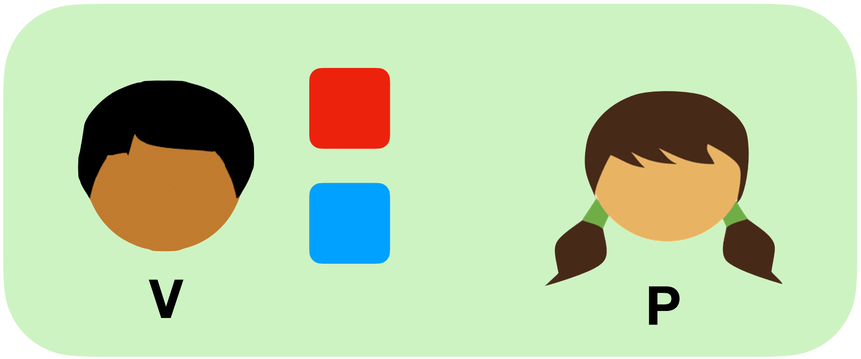
What is a *zero-knowledge* proof?



Completeness: If $x \in \mathscr{L}$ and if P and V are honest, then V accepts the proof (except with negligible probability)

"P can prove true things"

What is a *zero-knowledge* proof?

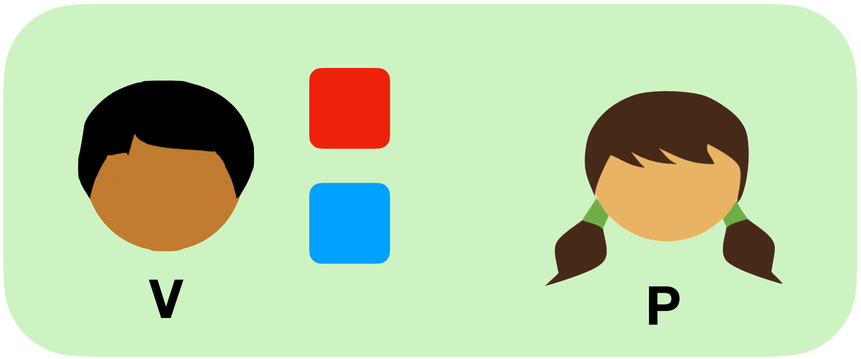


Completeness: If $x \in \mathscr{L}$ and if P and V are honest, then V accepts the proof (except with negligible probability)

"P can prove true things"

Soundness: If $x \notin \mathscr{L}$, even malicious P cannot cause honest V to accept the proof

What is a *zero-knowledge* proof?



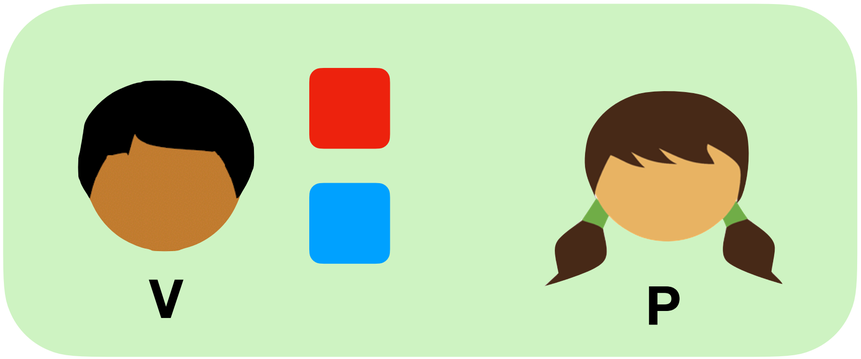
Completeness: If $x \in \mathscr{L}$ and if P and V are honest, then V accepts the proof (except with negligible probability)

"P can prove true things"

Soundness: If $x \notin \mathscr{L}$, even malicious P cannot cause honest V to accept the proof

"P cannot prove false things"

What is a *zero-knowledge* proof?



Completeness: If $x \in \mathscr{L}$ and if P and V are honest, then V accepts the proof (except with negligible probability)

"P can prove true things"

Soundness: If $x \notin \mathscr{L}$, even malicious P cannot cause honest V to accept the proof

"P cannot prove false things"

Zero Knowledge: "V learns nothing except that the thing is true"

What is a *zero-knowledge* proof?

Zero Knowledge: V can be simulated

What is a *zero-knowledge* proof?

Zero Knowledge: V can be simulated

Malicious Security

A protocol Π securely realizes a functionality f in the presence of a malicious adversary if for ***every*** *real-world adversary $\mathcal{A}$ corrupting party i,* ***there exists*** *an ideal-world adversary $\mathcal{S}_i$ (a simulator) such that for all inputs x, y the following holds:*

$$\text{Real}^{\Pi}_{\mathcal{A}}(x, y) \approx \text{Ideal}^{f}_{\mathcal{S}_i}(x, y)$$

How To Simulate It – A Tutorial on the Simulation Proof Technique*

Yehuda Lindell

Dept. of Computer Science
Bar-Ilan University, ISRAEL
lindell@biu.ac.il

April 25, 2021

Abstract

One of the most fundamental notions of cryptography is that of *simulation*. It stands behind the concepts of semantic security, zero knowledge, and security for multiparty computation. However, writing a simulator and proving security via the use of simulation is a non-trivial task, and one that many newcomers to the field often find difficult. In this tutorial, we provide a guide to how to write simulators and prove security via the simulation paradigm. Although we have tried to make this tutorial as stand-alone as possible, we assume some familiarity with the notions of secure encryption, zero-knowledge, and secure computation.

Keywords: secure computation, the simulation technique, tutorial

*This tutorial appeared in the book *Tutorials on the Foundations of Cryptography*, published in honor of Oded Goldreich's 60th birthday.

SIAM J. COMPUT.
Vol. 18, No. 1, pp. 186–208, February 1989

012

THE KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF SYSTEMS*

SHAFI GOLDWASSER†, SILVIO MICALI‡, AND CHARLES RACKOFF†

Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the "knowledge" contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.

Key words. cryptography, zero knowledge, interactive proofs, quadratic residues

AMS(MOS) subject classifications. 68Q15, 94A60

1. Introduction. It is often regarded that saying a language L is in NP (that is, acceptable in nondeterministic polynomial time) is equivalent to saying that there is a polynomial time "proof system" for L. The proof system we have in mind is one where on input x, a "prover" creates a string α, and the "verifier" then computes on x and α in time polynomial in the length of the binary representation of x to check that x is indeed in L. It is reasonable to ask if there is a more general, and perhaps more natural, notion of a polynomial time proof system. This paper proposes one such notion.

We will still allow the verifier only polynomial time and the prover arbitrary computing power, but will now allow both parties to flip unbiased coins. The result is a probabilistic version of NP, where a small probability of error is allowed. However, to obtain what appears to be the full generality of this idea, we must also allow the prover and verifier to *interact* (i.e., to talk back and forth) and to *keep secret* their coin tosses. We call these proof systems "interactive proof systems." This notion is formally defined in § 2, where we also define what it means for a language to have an interactive proof system.

It is far from clear how to use this power of interaction. Languages with nondeterministic polynomial time algorithms or with probabilistic polynomial time algorithms have proof systems with little or no interaction. We would therefore like examples of languages that appear to have neither nondeterministic nor probabilistic polynomial time algorithms, and yet have interactive proof systems. Although we do not present any such examples here, there are now examples in the literature. Using ideas from an initial version of this paper [GMR] Goldreich, Micali, and Wigderson [GMW] have shown that the "graph nonisomorphism" language has an interactive

* Received by the editors August 26, 1985; accepted for publication (in revised form) April 18, 1988. A preliminary version of this paper appeared in the Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science, 1986, pp. 174–187.

Editor's Note. This paper was originally scheduled to appear in the February 1988 Special Issue on Cryptography (SIAM J. Comput., 17 (1988)).

† Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. The work of these authors was supported by National Science Foundation grants DCR-84-13577 and DCR-85-09905.

‡ Computer Science Department, University of Toronto, Toronto, ONT M5S 1A4 Canada. The work of this author was supported by the Natural Sciences and Engineering Research Council of Canada under grant A3611.

How to Prove All NP Statements in Zero-Knowledge 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

ABSTRACT

Under the assumption that encryption functions exist, we show that all languages in NP possess zero-knowledge proofs.

That is, it is possible to demonstrate that a CNF formula is satisfiable without revealing any other property of the formula. In particular, without yielding neither a satisfying assignment nor weaker properties such as whether there is a satisfying assignment in which x_1=*TRUE*, or whether there is a satisfying assignment in which x_1=x_3 etc.

The above result allows us to prove two fundamental theorems in the field of (two-party and multi-party) cryptographic protocols. These theorems yield automatic and efficient transformations that, given a protocol that is correct with respect to an extremely weak adversary, output a protocol correct in the most adversarial scenario. Thus, these theorems imply powerful methodologies for developing two-party and multi-party cryptographic protocols.

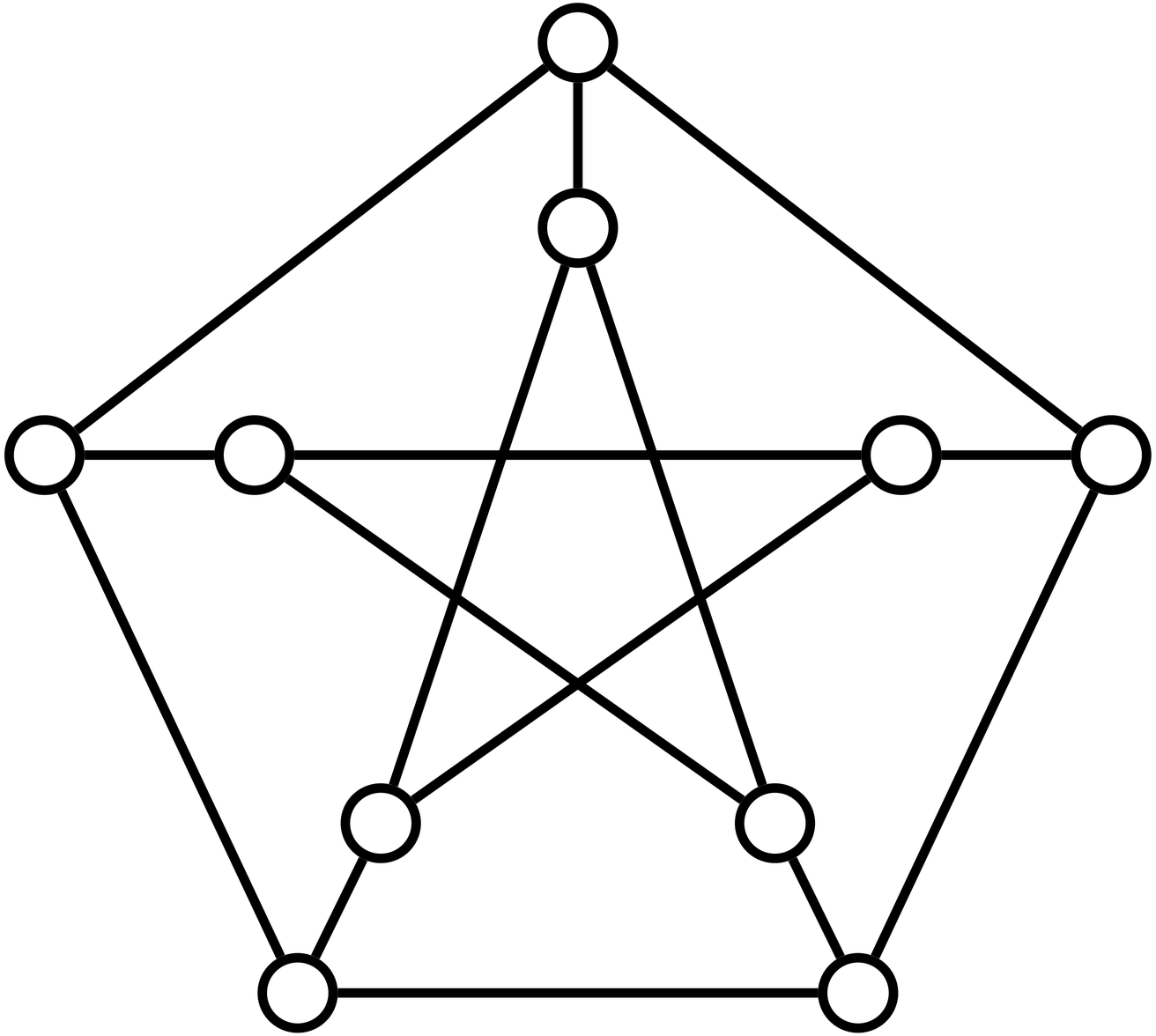
1. INTRODUCTION

A fundamental measure proposed by Goldwasser, Micali and Rackoff [GMR] is that of the amount of knowledge released during an interactive proof. Informally, an interactive proof is a two-party protocol through which one party (*the prover*) can convince his counterparts (*the verifier*) in the validity of some statement concerning a common input. (The prover should be able to do so if and only if the statement is indeed valid.) Loosely speaking, an interactive proof system is called zero-knowledge if whatever the verifier

Work done while the first author was in the Laboratory for Computer Science, MIT, partially supported by an IBM Postdoctoral Fellowship, and NSF Grant DCR-8509905. The second author was supported by NSF Grant DCR-8413577 and an IBM Faculty Development Award. Work done while the third author was in Mathematical Sciences Research Institute, Berkeley.

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 171-185, 1987.

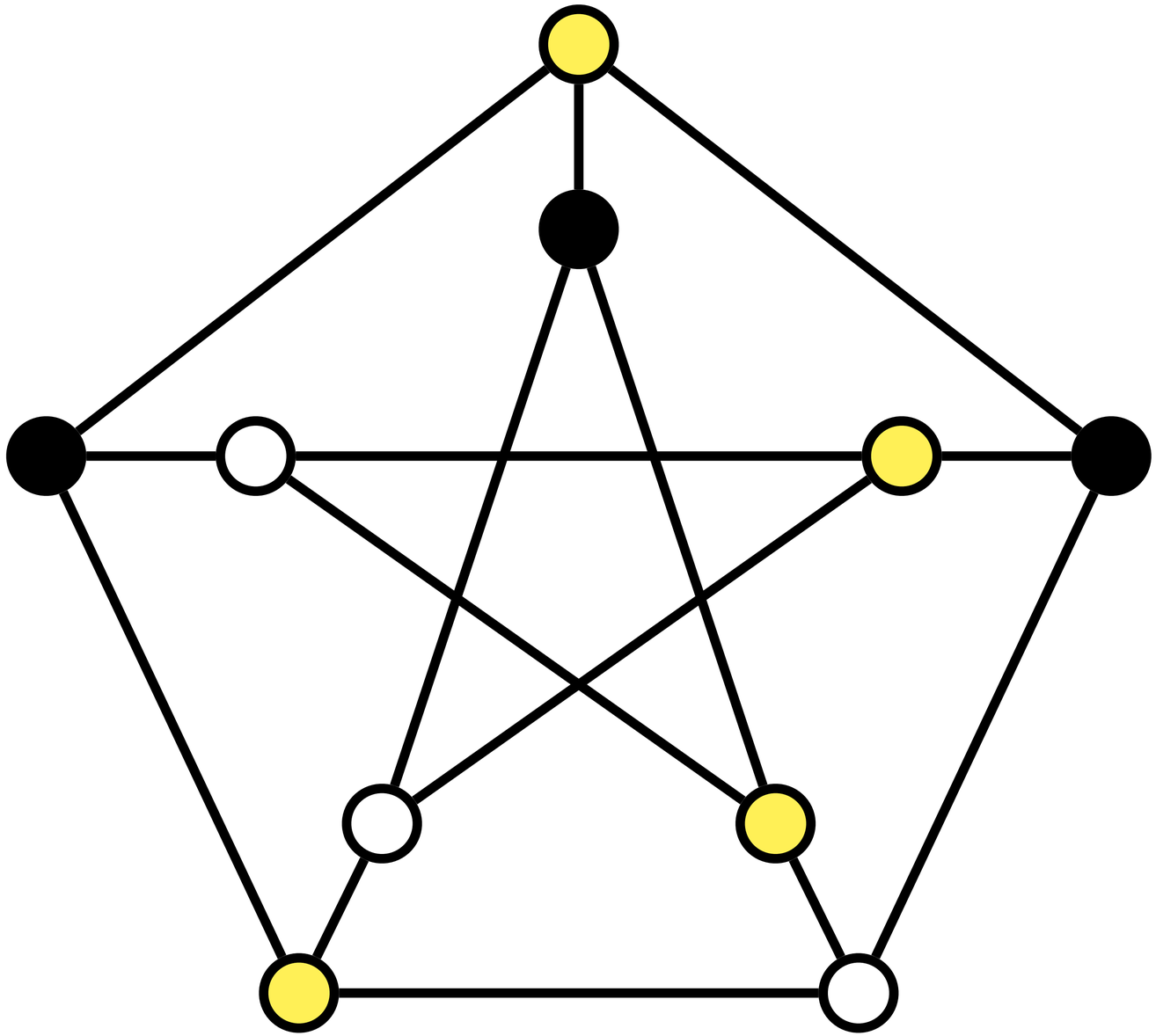
Graph 3-Coloring



Color each vertex so that each edge has two colors

Believed to be a hard problem; NP complete

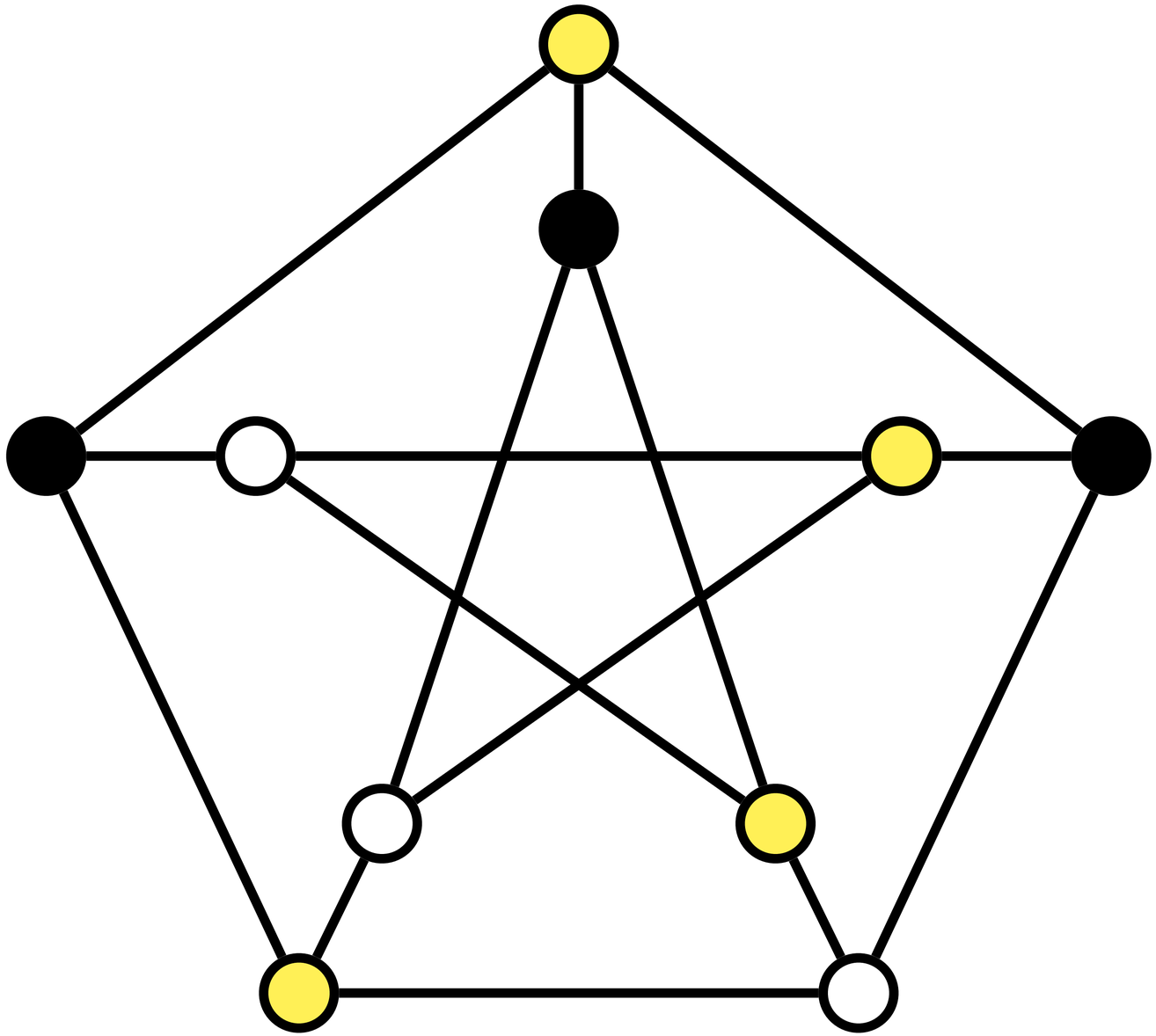
Graph 3-Coloring



Color each vertex so that each edge has two colors

Believed to be a hard problem; NP complete

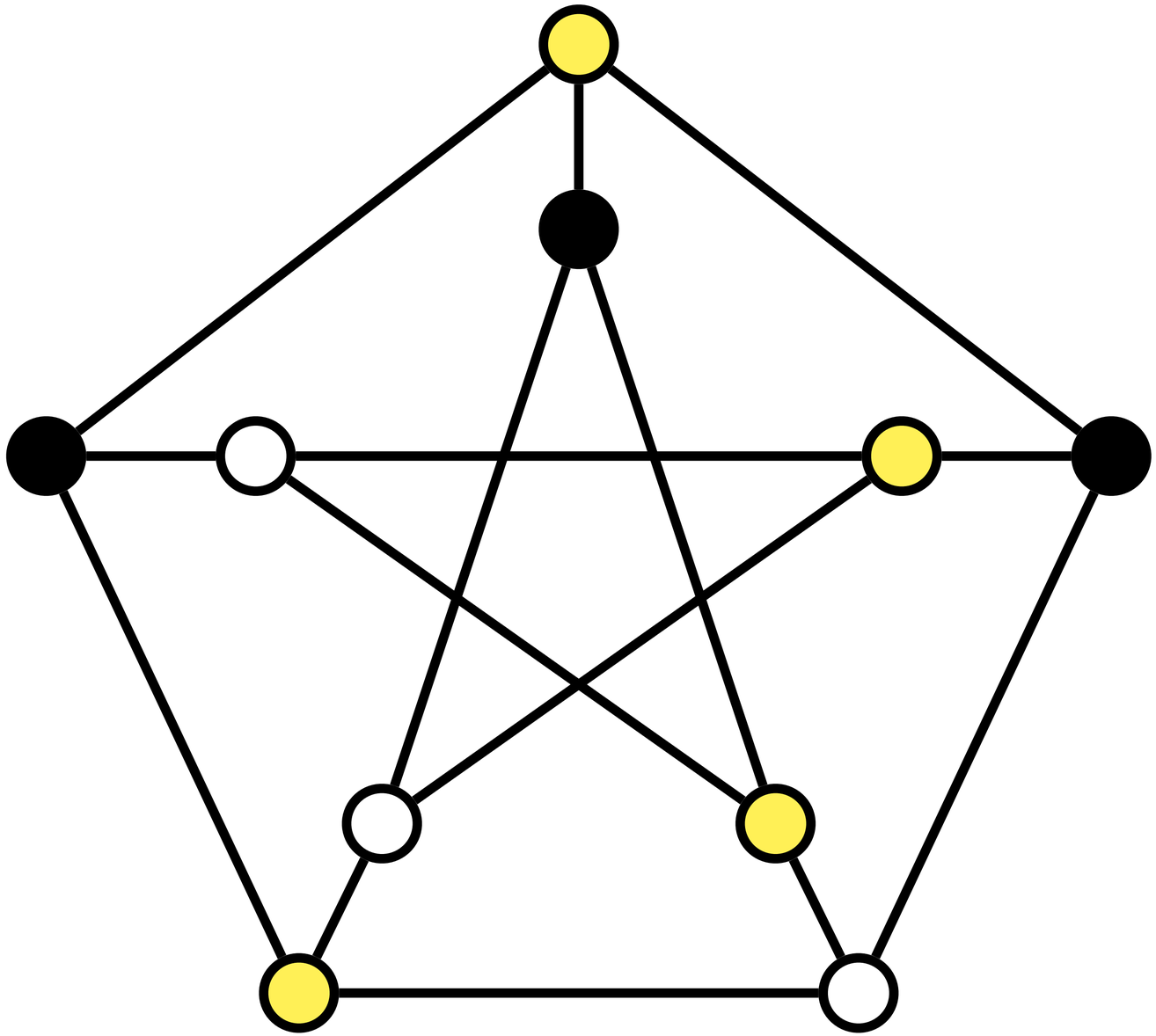
Graph 3-Coloring



ZK Proof system for 3-colorability

Statement: a graph
“this graph is 3-colorable”

Graph 3-Coloring

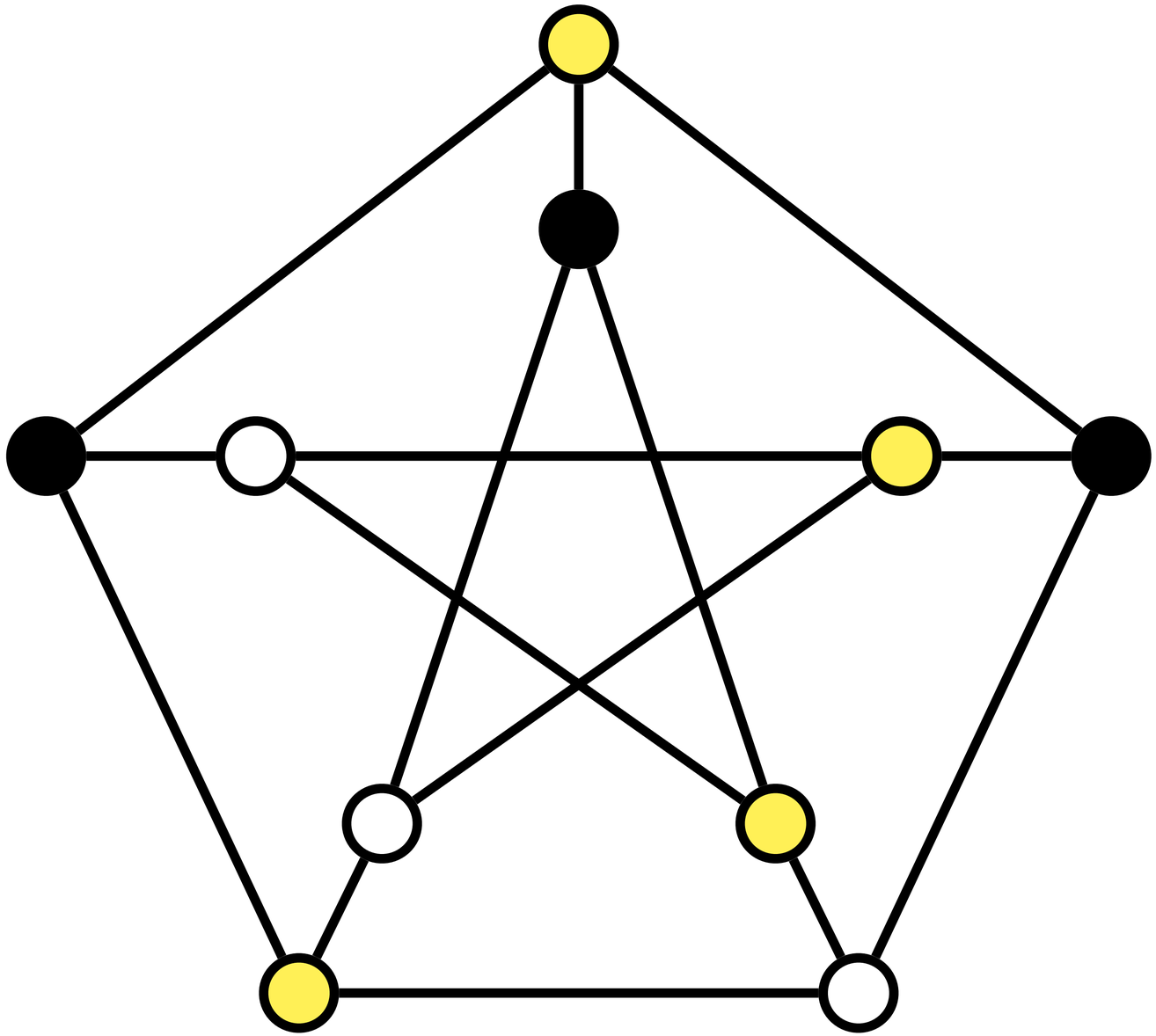


ZK Proof system for 3-colorability

Statement: a graph
"this graph is 3-colorable"

Witness:

Graph 3-Coloring

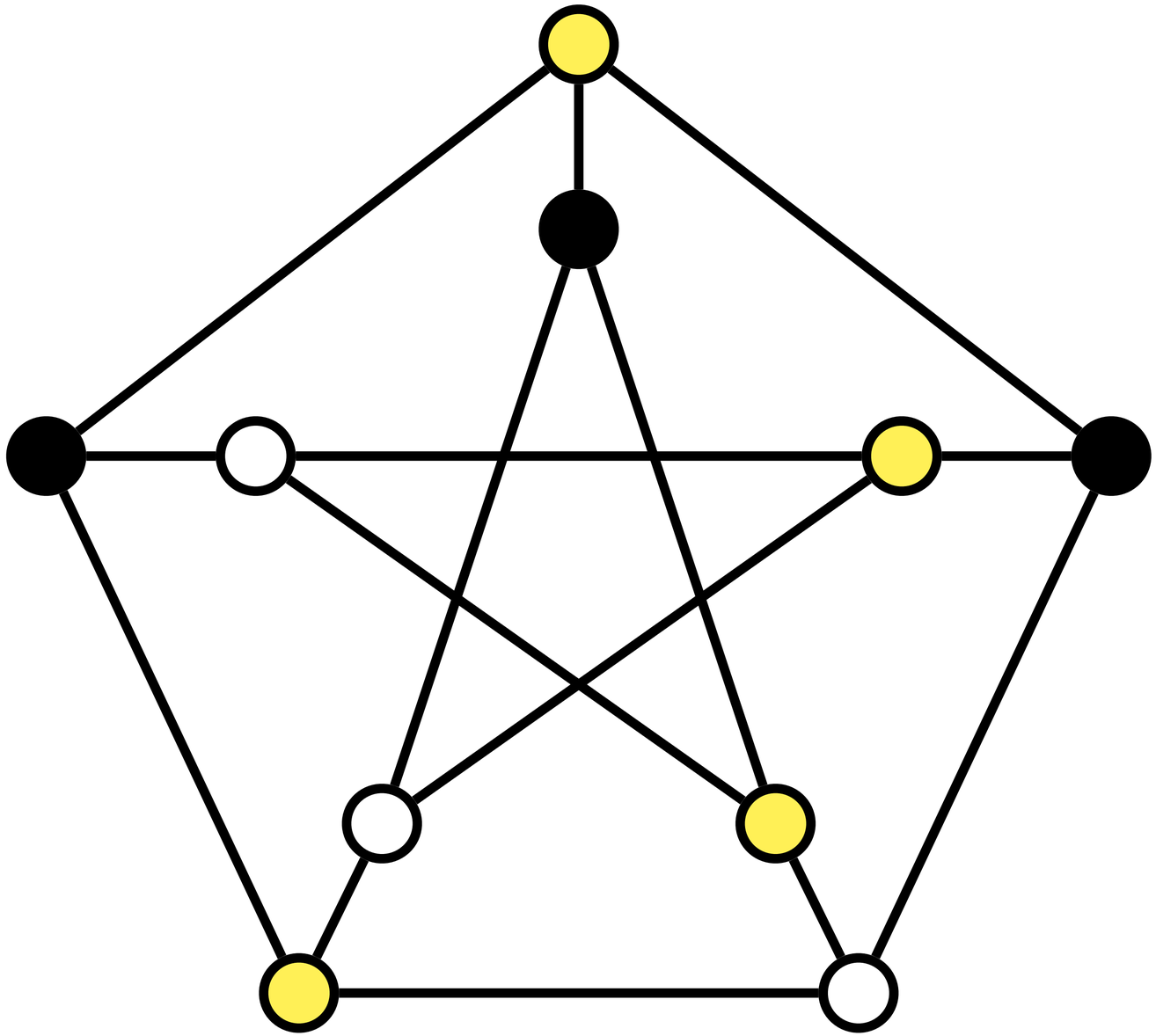


ZK Proof system for 3-colorability

Statement: a graph
"this graph is 3-colorable"

Witness: a coloring

Graph 3-Coloring



ZK Proof system for 3-colorability

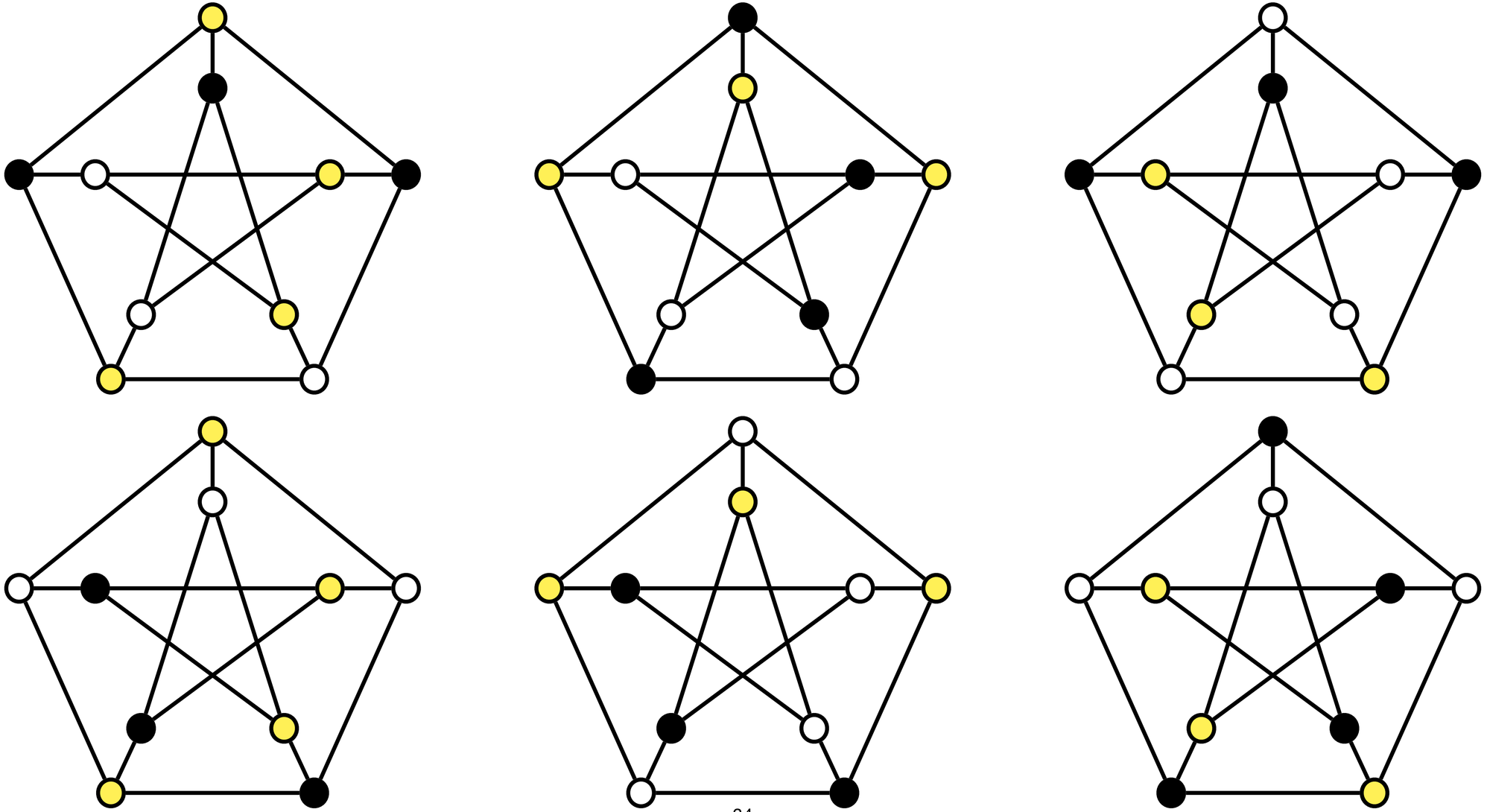
Statement: a graph
"this graph is 3-colorable"

Witness: a coloring

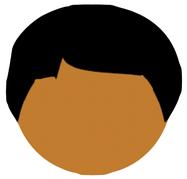
Basic cryptographic tool: Commitments

Graph 3-Coloring

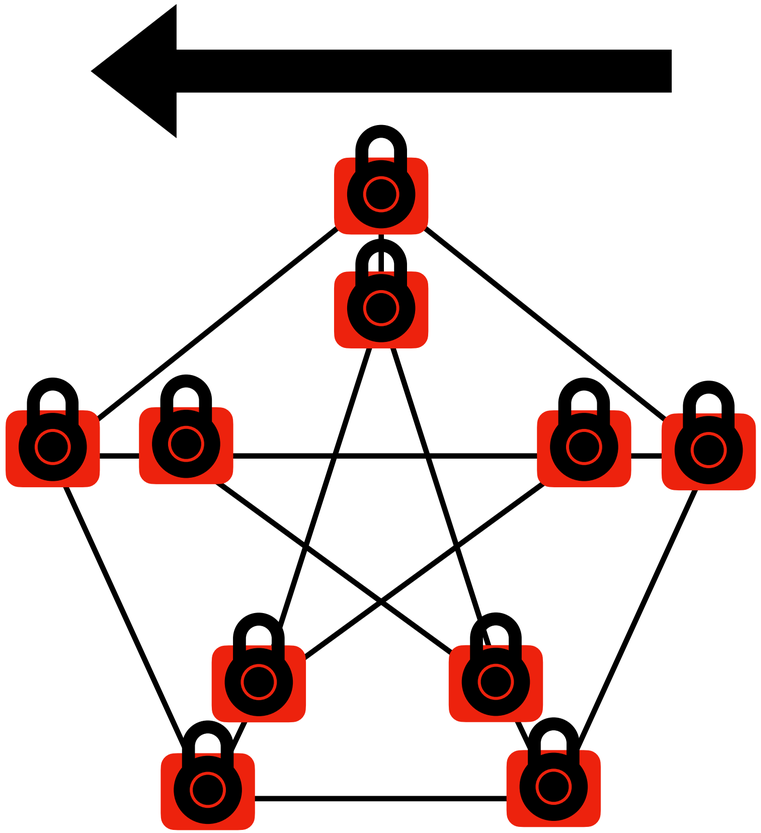
Observation: If you have 1 coloring, you can construct 6 colorings



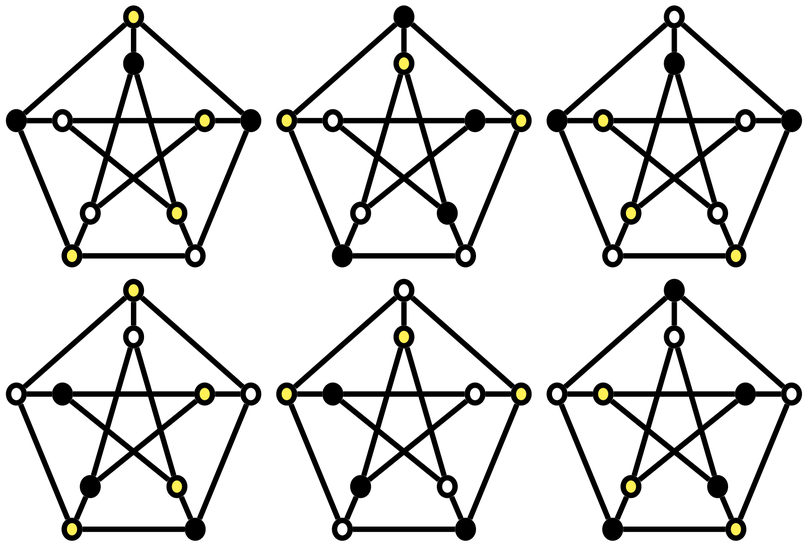
Graph 3-Coloring



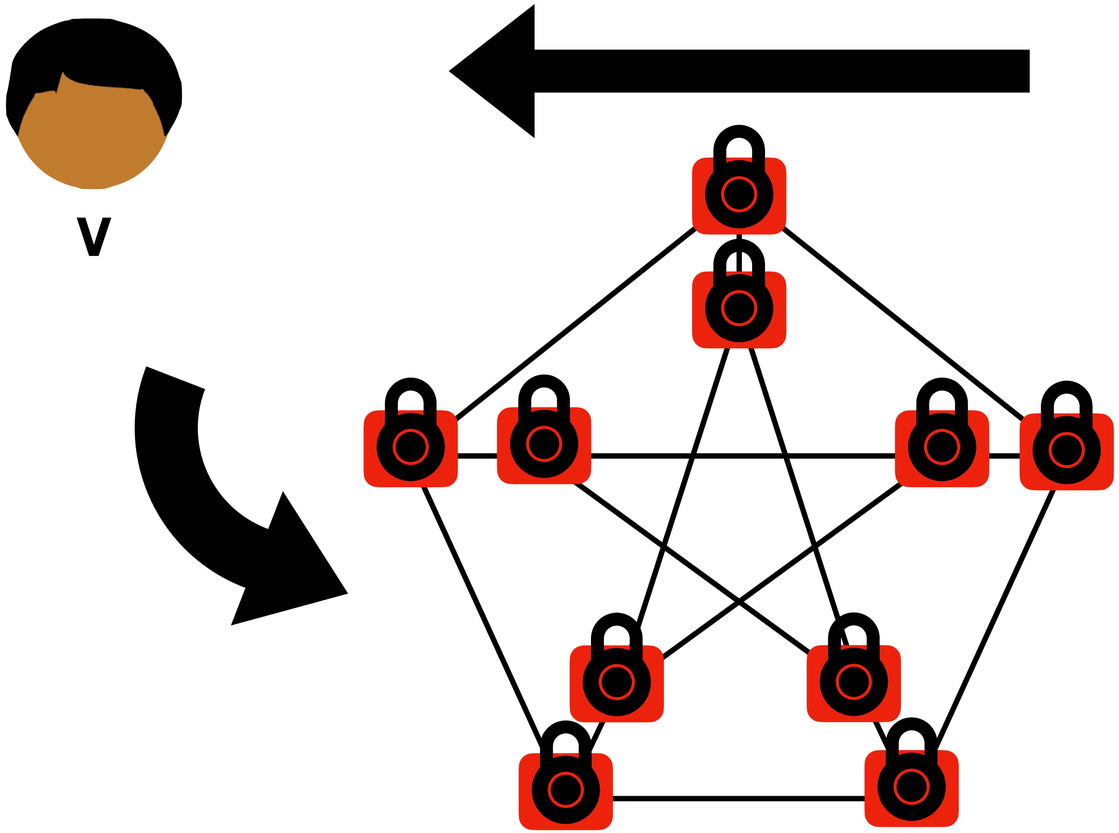
V

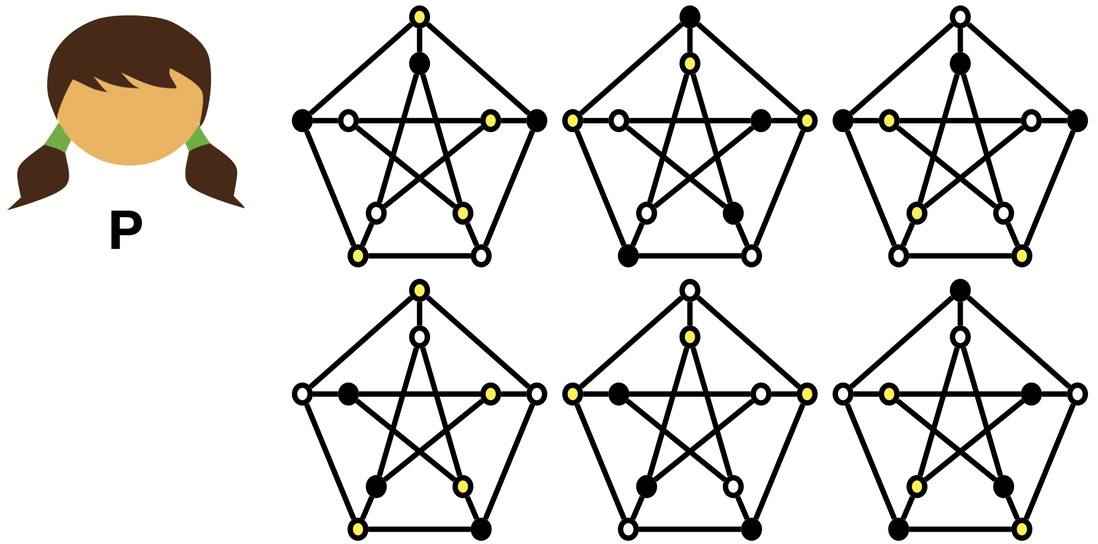


P

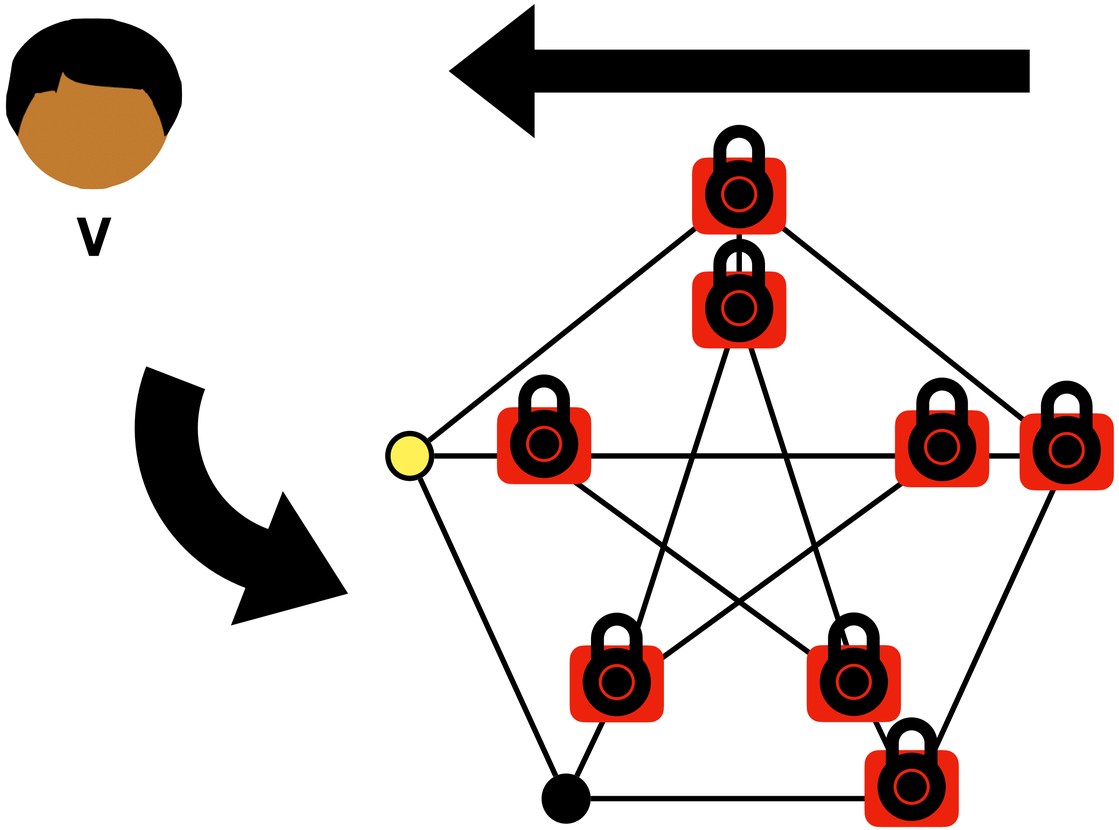


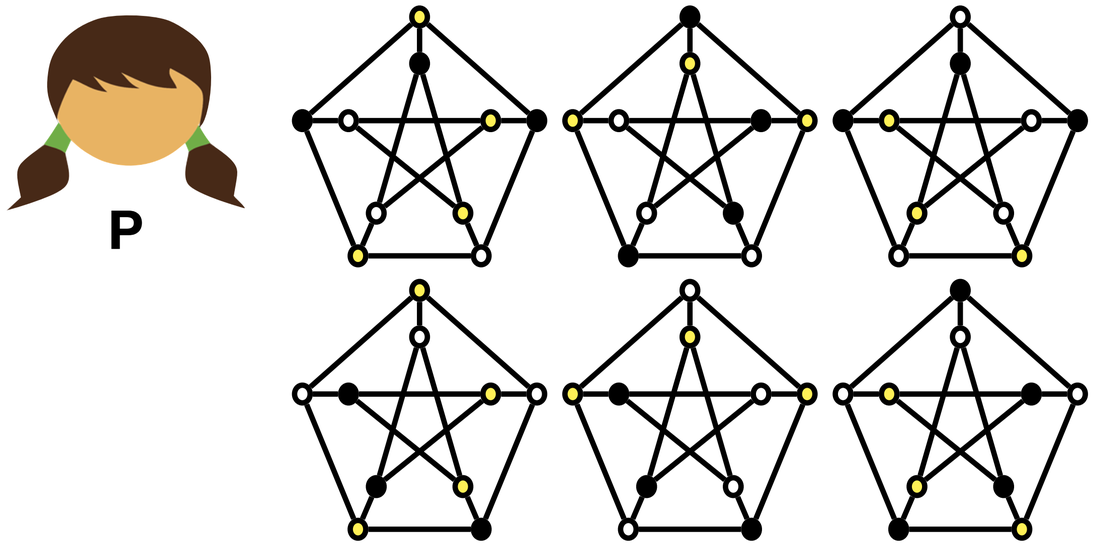
Graph 3-Coloring



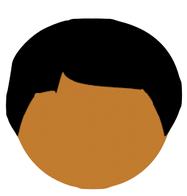


Graph 3-Coloring

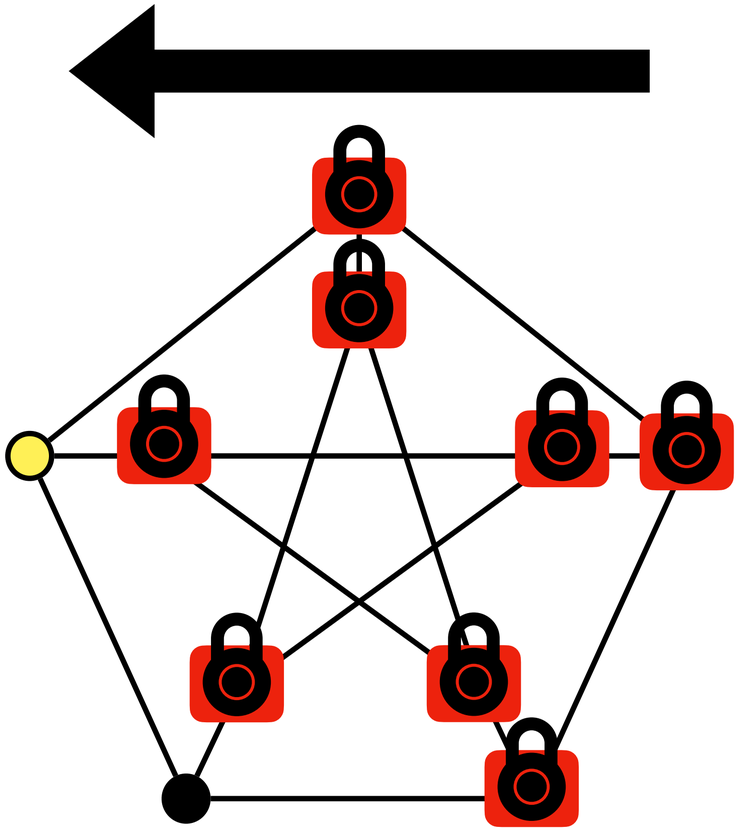




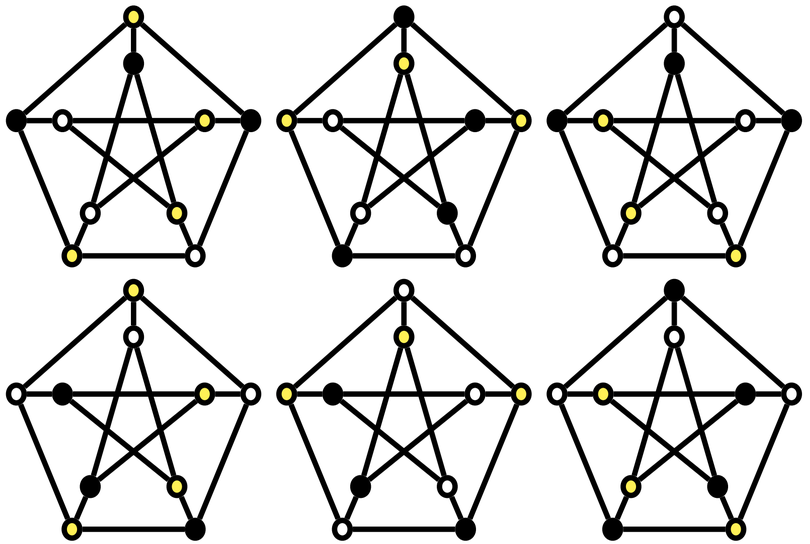
Graph 3-Coloring



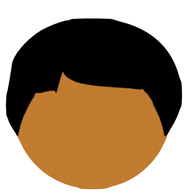
V



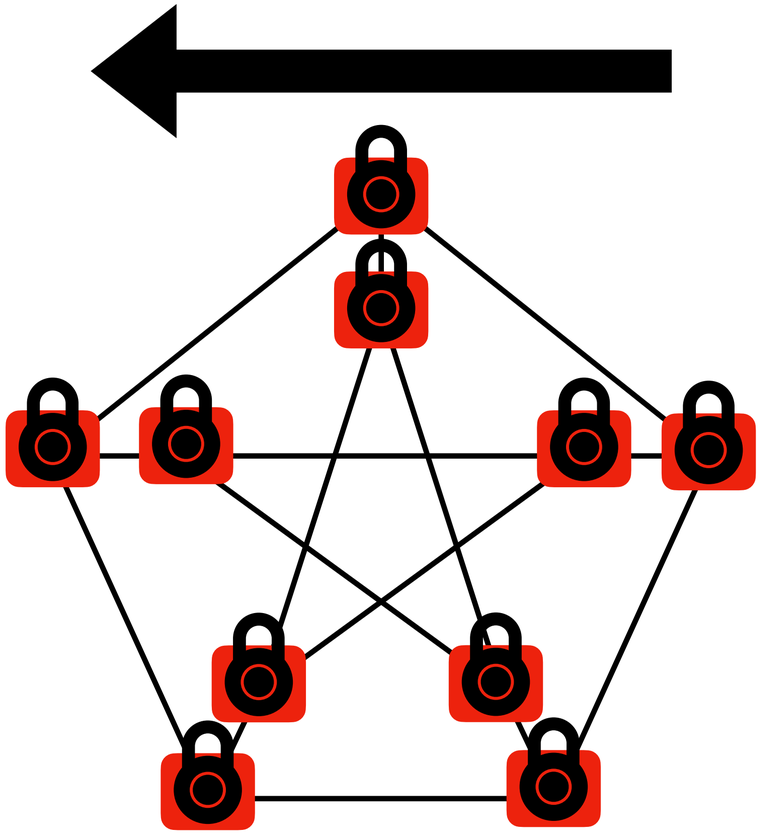
P



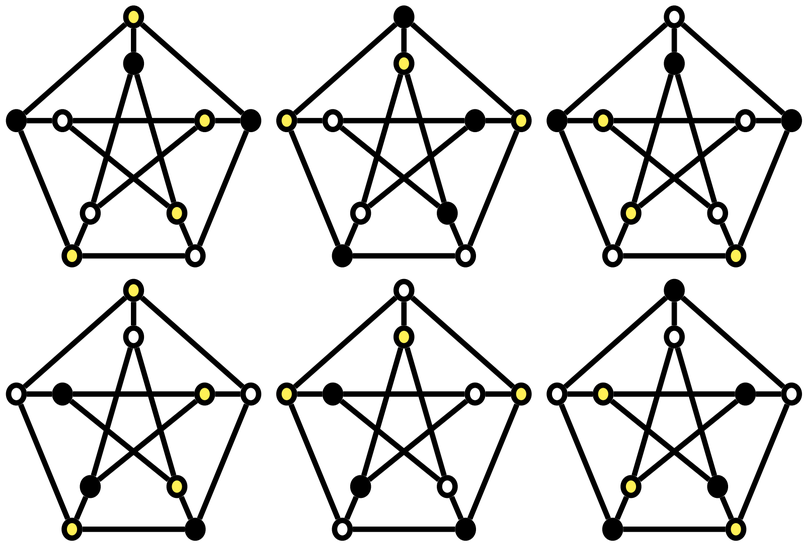
Graph 3-Coloring



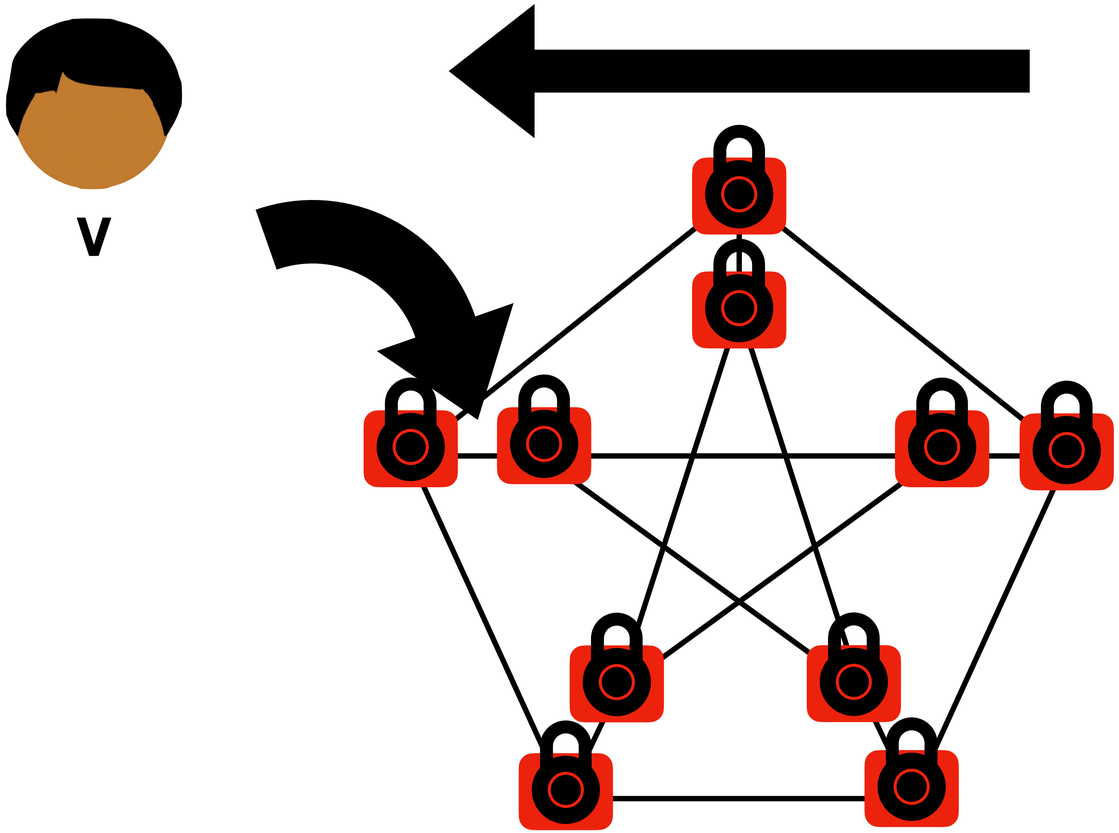
V

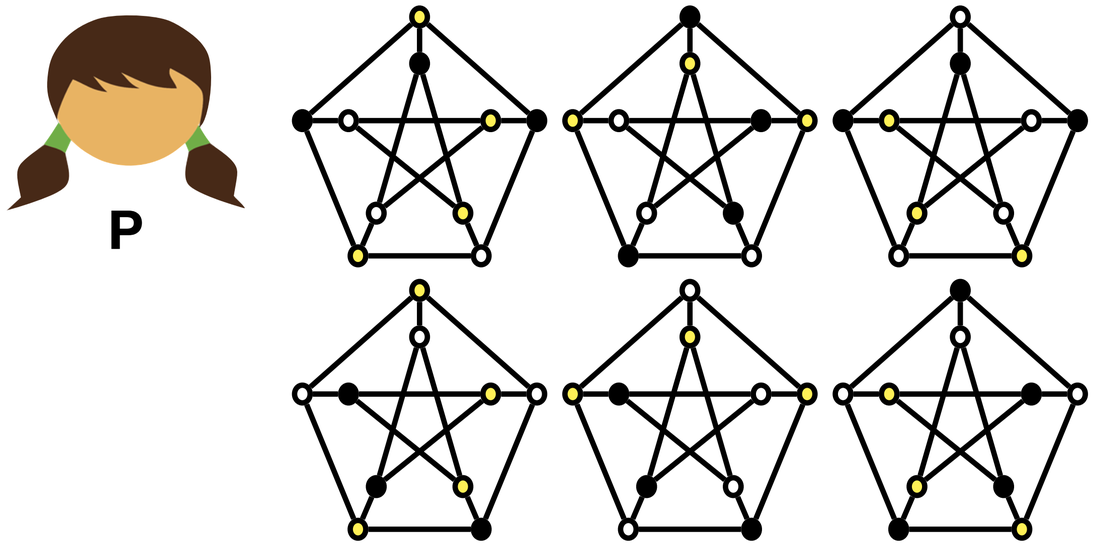


P

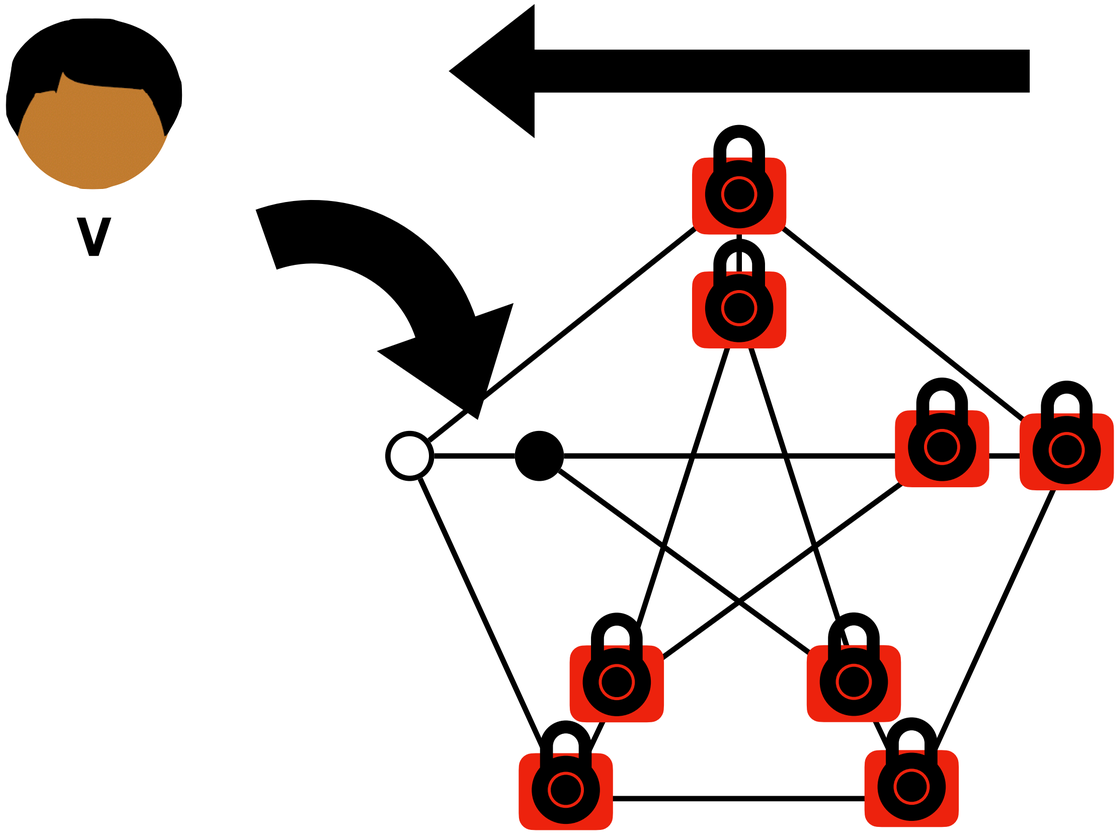


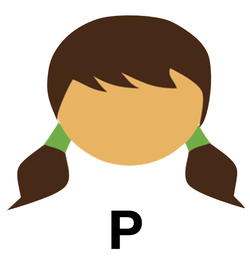
Graph 3-Coloring

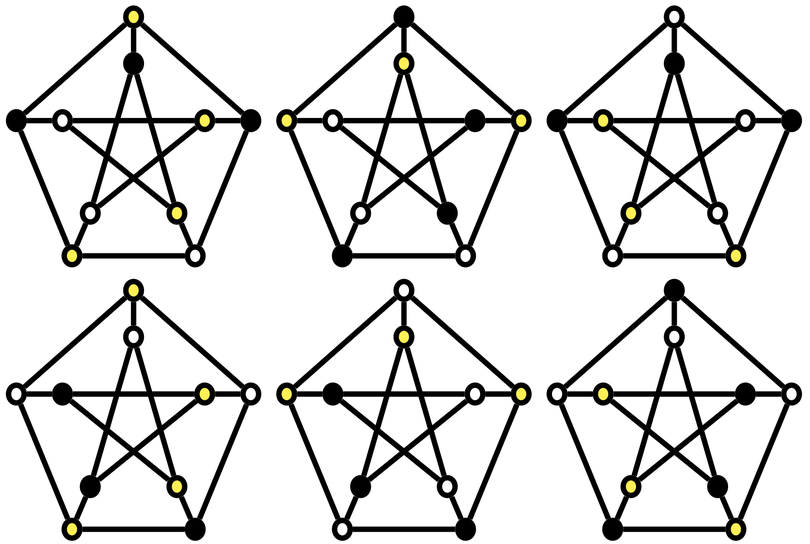




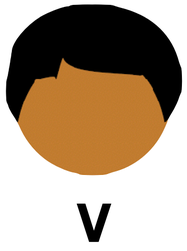
Graph 3-Coloring



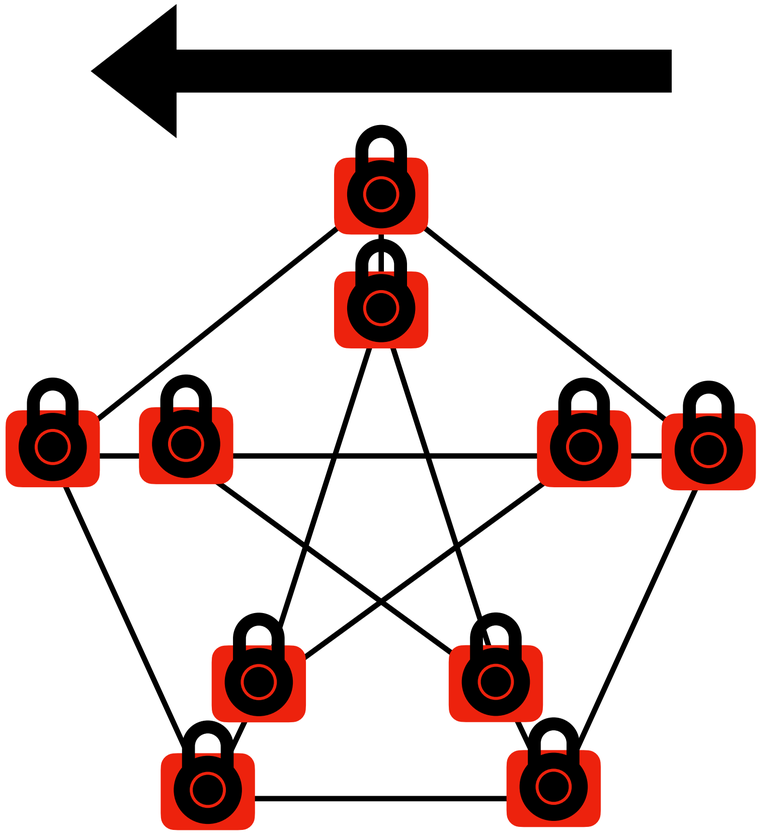




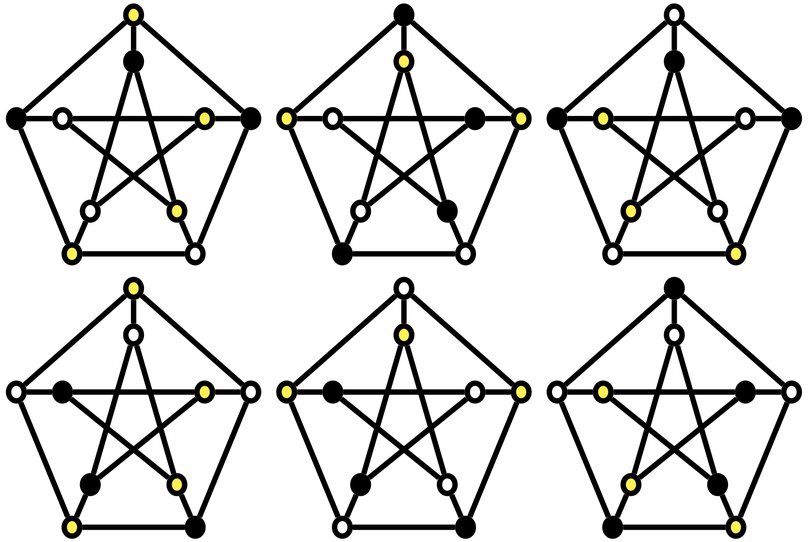
Graph 3-Coloring



V

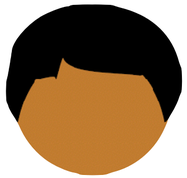


P

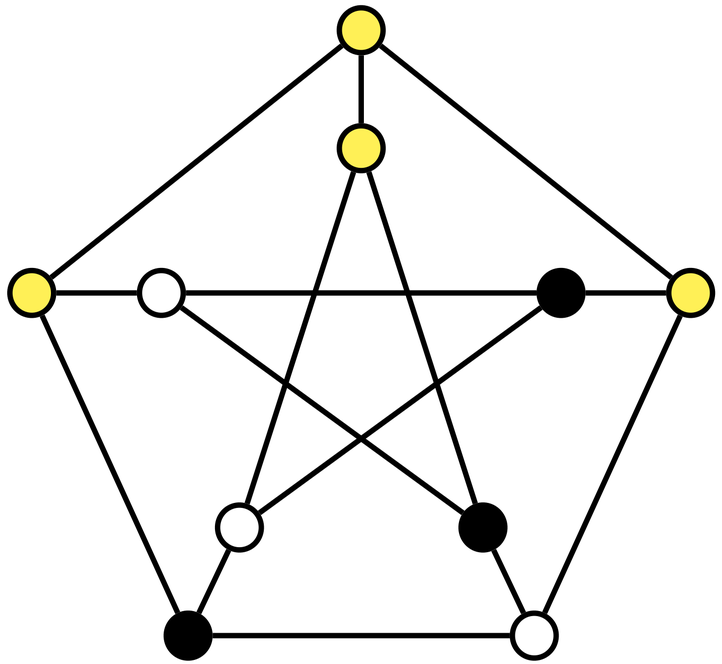


Completeness: Honest P can always open a valid edge

Graph 3-Coloring



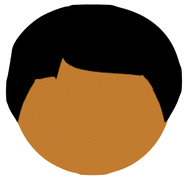
V



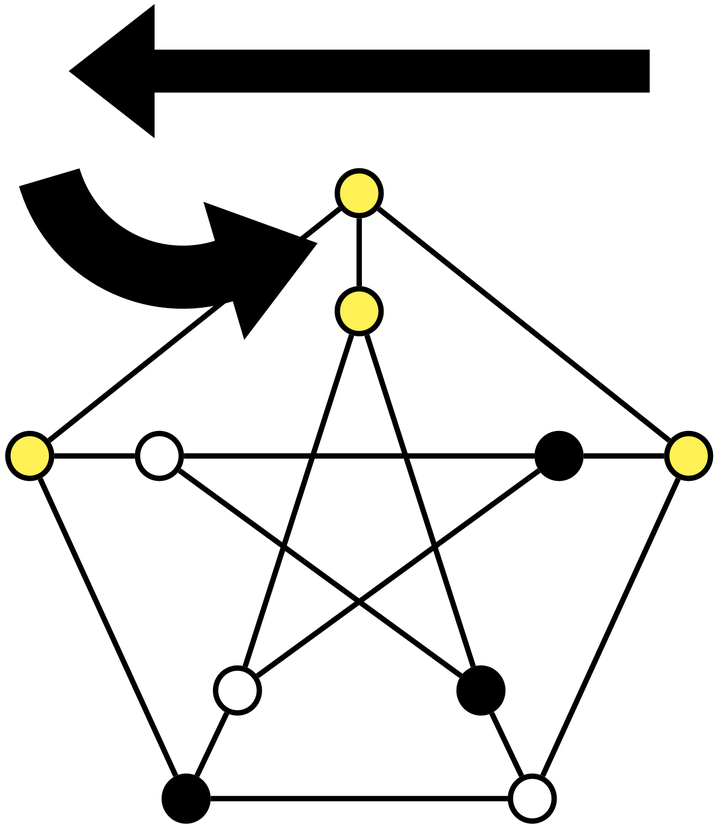
P

Soundness: P who does not know a coloring must cheat on some edge

Graph 3-Coloring



V

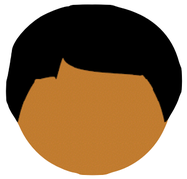


P

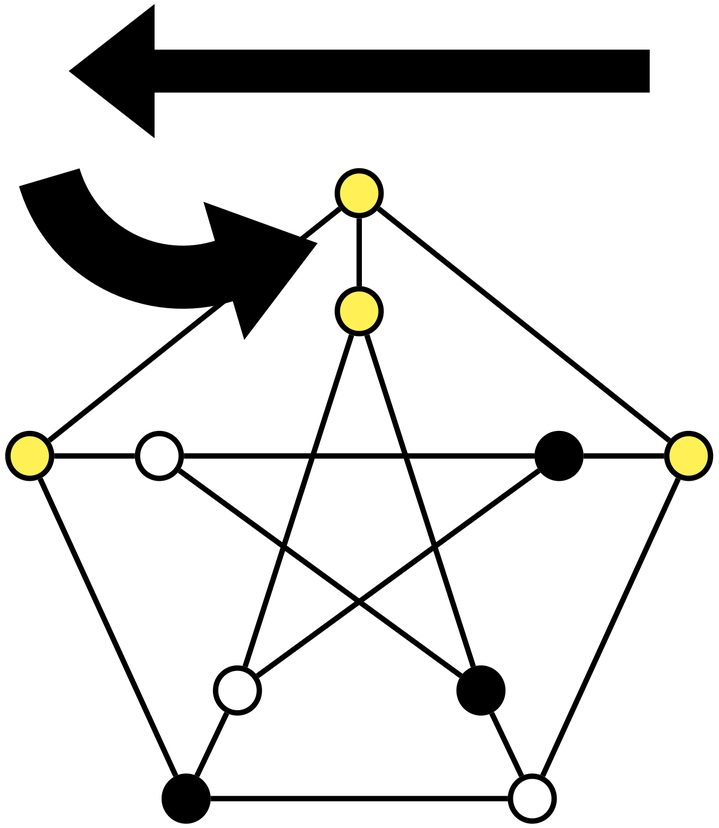
Soundness: P who does not know a coloring must cheat on some edge

V chooses that edge with probability $\frac{1}{E}$

Graph 3-Coloring



V



P

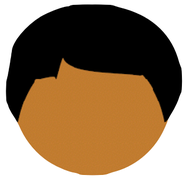
Soundness: P who does not know a coloring must cheat on some edge

V chooses that edge with probability $\frac{1}{E}$

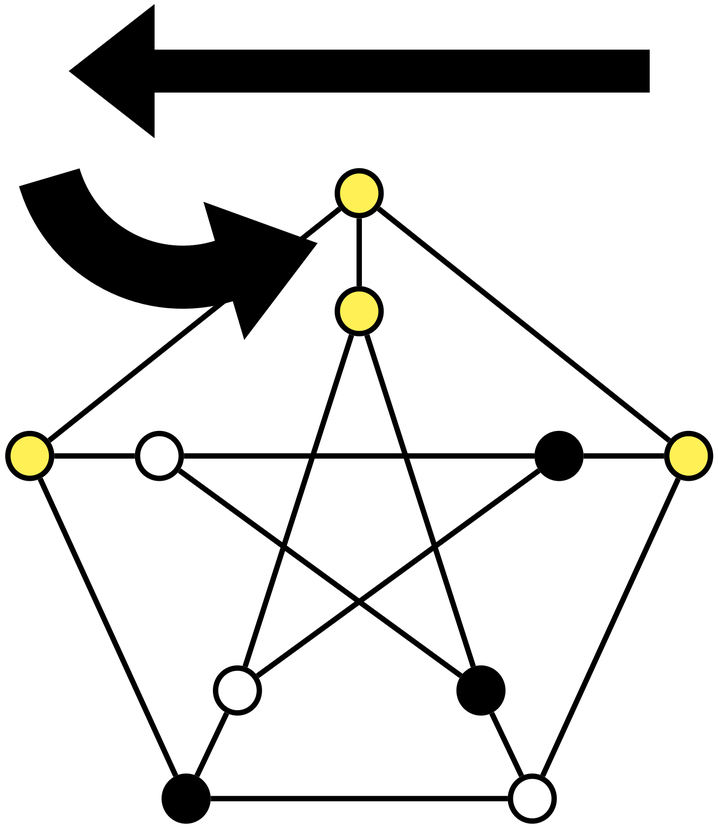
If parties repeat r times, P wins with probability at most

$$\left(\frac{E-1}{E}\right)^r$$

Graph 3-Coloring



V



P

Soundness: P who does not know a coloring must cheat on some edge

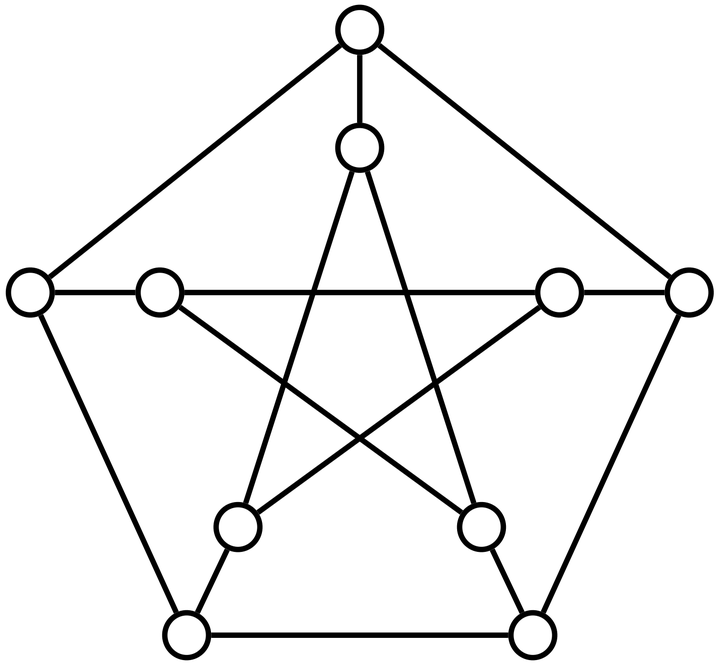
V chooses that edge with probability $\frac{1}{E}$

If parties repeat r times, P wins with probability at most

$\left(\frac{E-1}{E}\right)^r$ *Negligible in r*

Graph 3-Coloring
Zero Knowledge

V

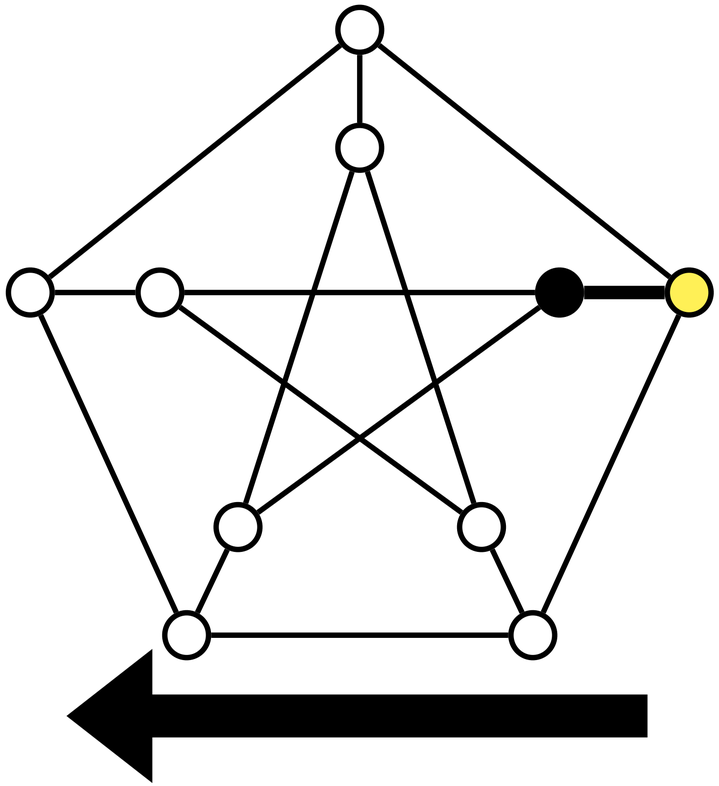


$\mathcal{S}$

Intuition: Adversarial V is just seeing two different random colors

Graph 3-Coloring
Zero Knowledge

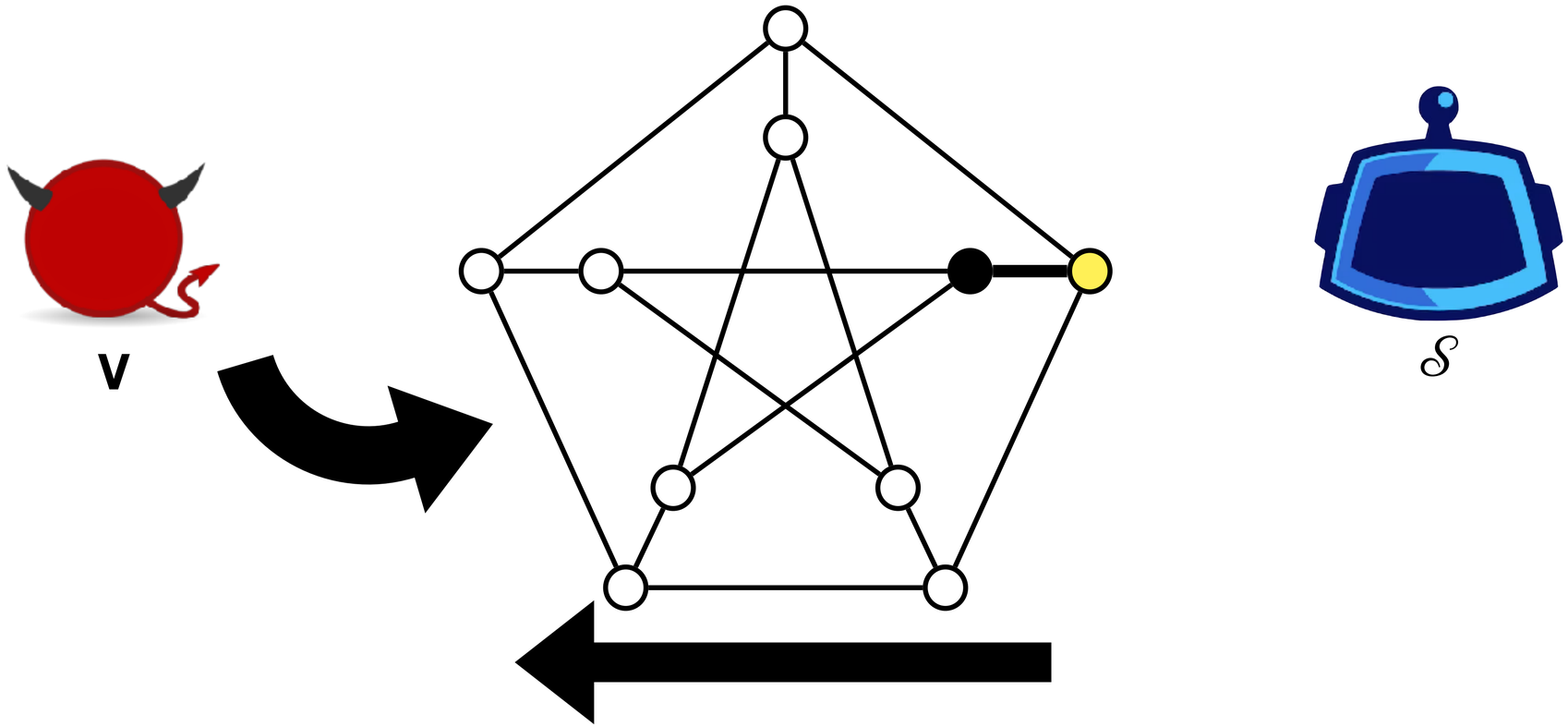
V



$\mathcal{S}$

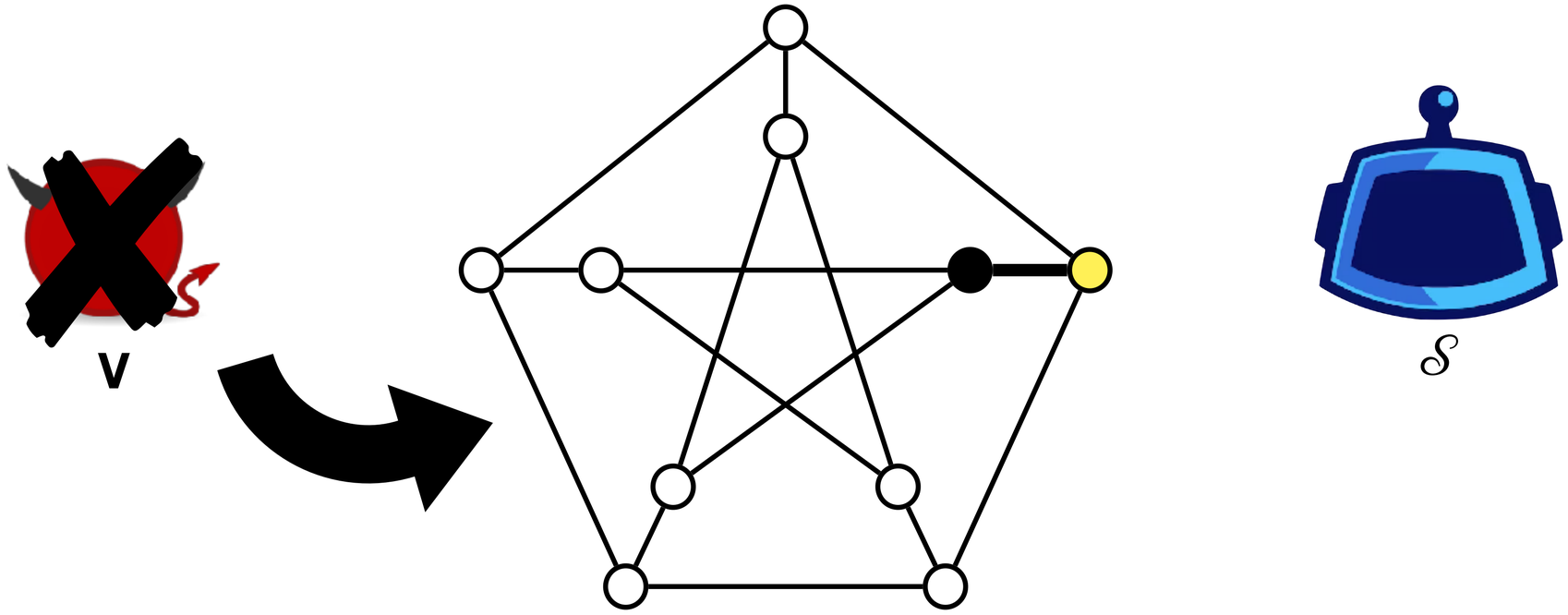
Intuition: Adversarial V is just seeing two different random colors

Graph 3-Coloring
Zero Knowledge



Intuition: Adversarial V is just seeing two different random colors

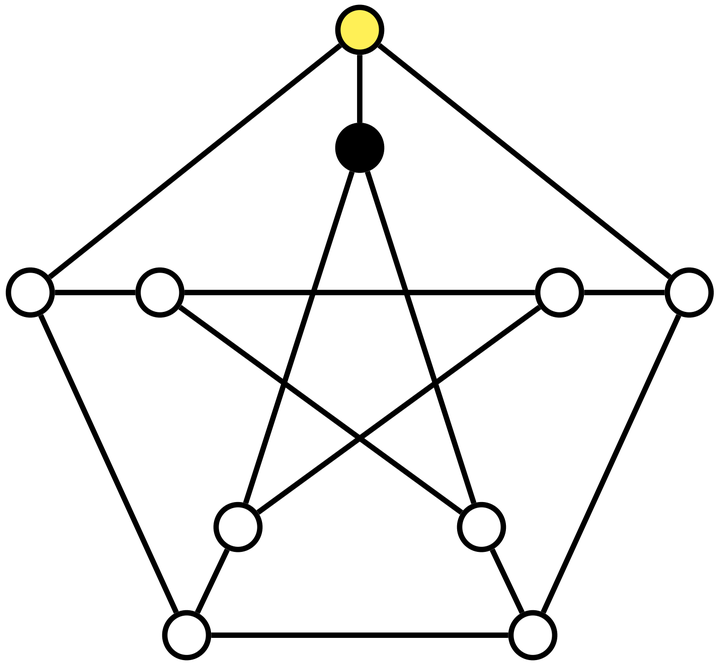
Graph 3-Coloring
Zero Knowledge



Intuition: Adversarial V is just seeing two different random colors

Graph 3-Coloring
Zero Knowledge

V

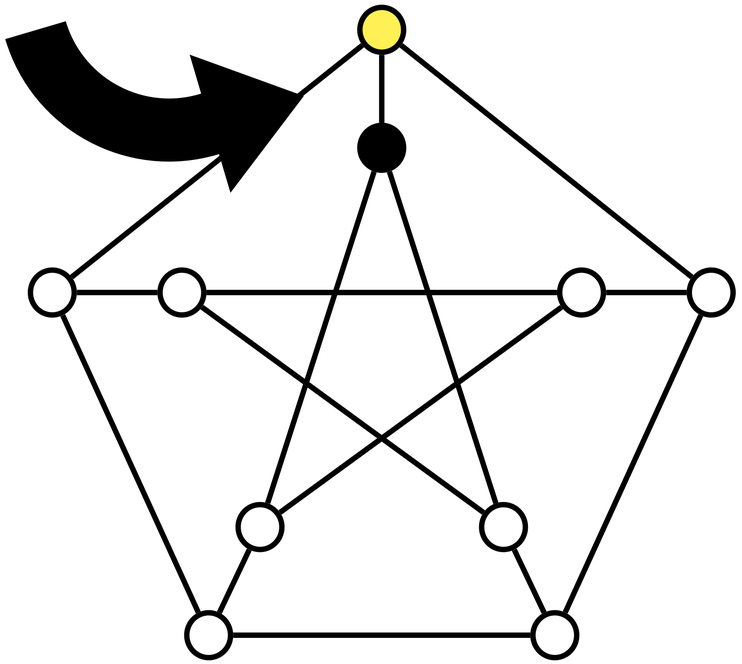


$\mathcal{S}$

Intuition: Adversarial V is just seeing two different random colors

Graph 3-Coloring
Zero Knowledge

V

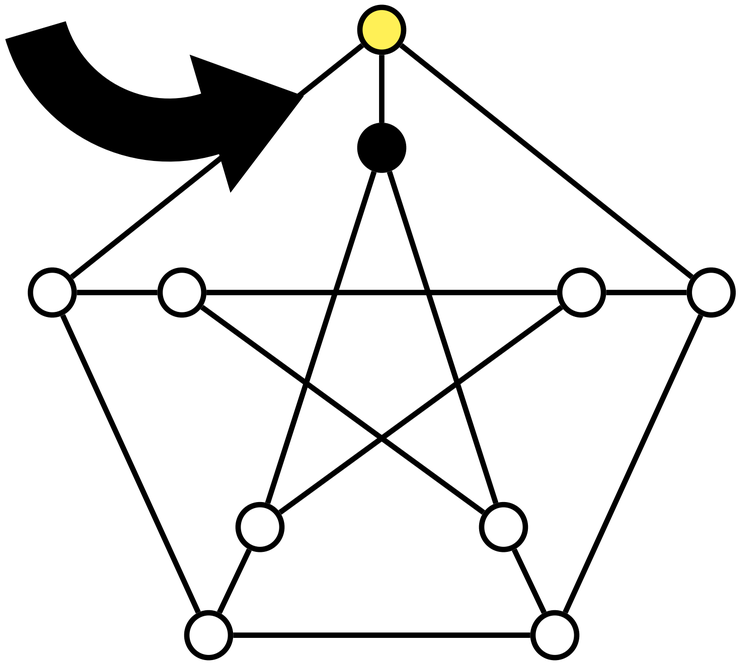


$\mathcal{S}$

Intuition: Adversarial V is just seeing two different random colors

Graph 3-Coloring
Zero Knowledge

V

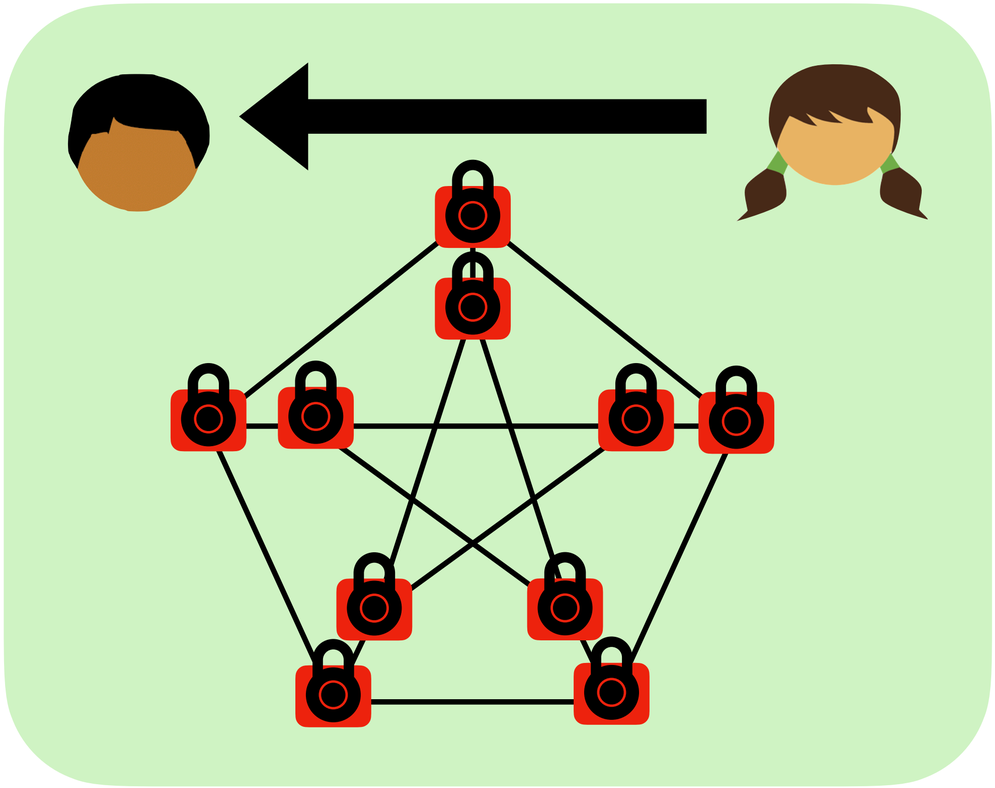


$\mathcal{S}$

Adversarial V will soon (in polynomial tries) choose the chosen edge

After polynomial iterations, the proof interaction will terminate,

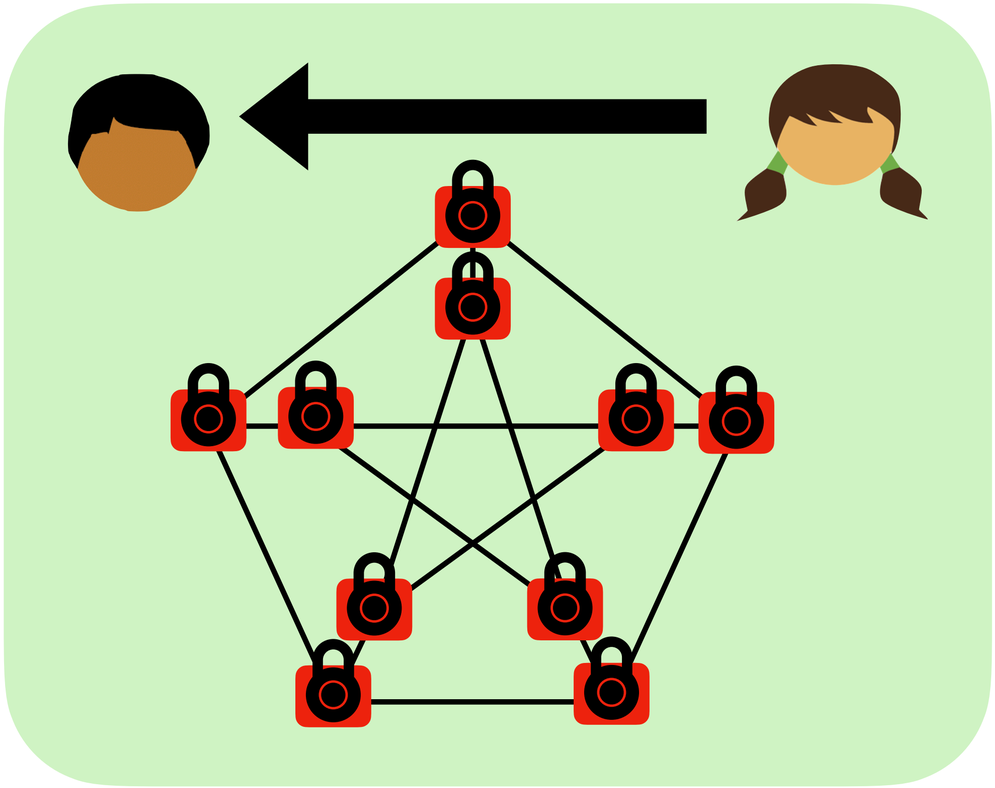
and $\mathcal{S}$ can output whatever V outputs



Not a practical ZK protocol

Soundness is very bad: we need many, many repetitions

Still, demonstrates feasibility: we can prove any NP statement!



Not a practical ZK protocol

Soundness is very bad: we need many, many repetitions

Still, demonstrates feasibility: we can prove any NP statement!

An interesting NP statement:

"While running protocol Π, and given the messages you have sent me so far, I constructed **this** message according to Π."

HOW TO PLAY ANY MENTAL GAME

or

A Completeness Theorem for Protocols with Honest Majority

(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

Abstract

We present a polynomial-time algorithm that, given as a input the description of a game with incomplete information and any number of players, produces a protocol for playing the game that leaks no partial information, provided the majority of the players is honest.

Our algorithm automatically solves all the multi-party protocol problems addressed in complexity-based cryptography during the last 10 years. It actually is a *completeness theorem* for the class of distributed protocols with honest majority. Such completeness theorem is optimal in the sense that, if the majority of the players is not honest, some protocol problems have no efficient solution[2].

1. Introduction

Before discussing how to "make playable" a general game with incomplete information (which we do in section 6) let us address the problem of making playable a special class of games, the *Turing machine games* (*Tm-games* for short).

Informally, n parties, respectively and individually owning secret inputs $x_1,...,x_n$, would like to *correctly* run a given Turing machine M on these x_i's while keeping the maximum possible *privacy* about them. That is, they want to compute $y=M(x_1,...,x_n)$ without revealing more about the x_i's than it is already contained in the value y itself. For instance, if M computes the sum of the x_i's, every single player should not be able to learn more than the sum of the inputs of the other parties. Here M may very well be a probabilistic Turing machine. In this case, all players want to agree on a single string y, selected with the right probability distribution, as M's output.

The correctness and privacy constraint of a Tm-game can be easily met with the help of an extra, trusted party P. Each player i simply gives his secret input x_i to P. P will privately run the prescribed Turing machine, M, on these inputs and publically announce M's output. Making a Tm-game playable essentially means that the correctness and privacy constraints can be satisfied by the n players themselves, without invoking any extra party. Proving that Tm-games are playable retains most of the flavor and difficulties of our general theorem.

2. Preliminary Definitions

2.1 Notation and Conventions for Probabilistic Algorithms.

We emphasize the number of inputs received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$", if it receives two inputs we write $A(\cdot,\cdot)$ and so on.

RV will stand for "random variable"; in this paper we only consider RVs that assume values in

Work partially supported by NSF grants DCR-8509905 and DCR-8413577, an IBM post-doctoral fellowship and an IBM faculty development award. The work was done when the first author was at the Laboratory for Computer Science at MIT; and the second author at the mathematical Sciences Research Institute at UC-Berkeley.

GMW Compiler:

Automatically upgrades any semi-honest protocol to malicious security

Prove in ZK that each protocol message is honestly constructed

An interesting NP statement:

"While running protocol Π, and given the messages you have sent me so far, I constructed **this** message according to Π."

HOW TO PLAY ANY MENTAL GAME

or

A Completeness Theorem for Protocols with Honest Majority

(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

Abstract

We present a polynomial-time algorithm that, given as a input the description of a game with incomplete information and any number of players, produces a protocol for playing the game that leaks no partial information, provided the majority of the players is honest.

Our algorithm automatically solves all the multi-party protocol problems addressed in complexity-based cryptography during the last 10 years. It actually is a *completeness theorem* for the class of distributed protocols with honest majority. Such completeness theorem is optimal in the sense that, if the majority of the players is not honest, some protocol problems have no efficient solution[2].

1. Introduction

Before discussing how to "make playable" a general game with incomplete information (which we do in section 6) let us address the problem of making playable a special class of games, the *Turing machine games* (*Tm-games* for short).

Informally, n parties, respectively and individually owning secret inputs $x_1,...,x_n$, would like to *correctly* run a given Turing machine M on these x_i's while keeping the maximum possible *privacy* about them. That is, they want to compute $y=M(x_1,...,x_n)$ without revealing more about the x_i's than it is already contained in the value y itself. For instance, if M computes the sum of the x_i's, every single player should not be able to learn more than the sum of the inputs of the other parties. Here M may very well be a probabilistic Turing machine. In this case, all players want to agree on a single string y, selected with the right probability distribution, as M's output.

The correctness and privacy constraint of a Tm-game can be easily met with the help of an extra, trusted party P. Each player i simply gives his secret input x_i to P. P will privately run the prescribed Turing machine, M, on these inputs and publically announce M's output. Making a Tm-game playable essentially means that the correctness and privacy constraints can be satisfied by the n players themselves, without invoking any extra party. Proving that Tm-games are playable retains most of the flavor and difficulties of our general theorem.

2. Preliminary Definitions

2.1 Notation and Conventions for Probabilistic Algorithms.

We emphasize the number of inputs received by an algorithm as follows. If algorithm A receives only one input we write "$A(\cdot)$", if it receives two inputs we write $A(\cdot,\cdot)$ and so on.

RV will stand for "random variable"; in this paper we only consider RVs that assume values in

Work partially supported by NSF grants DCR-8509905 and DCR-8413577, an IBM post-doctoral fellowship and an IBM faculty development award. The work was done when the first author was at the Laboratory for Computer Science at MIT; and the second author at the mathematical Sciences Research Institute at UC-Berkeley.

GMW Compiler:

Automatically upgrades any semi-honest protocol to malicious security

Prove in ZK that each protocol message is honestly constructed

Note: Primarily a feasibility result

An interesting NP statement:

"While running protocol Π, and given the messages you have sent me so far, I constructed **this** message according to Π."

Setting

Semi-honest Security

Malicious Security

General-Purpose Tools

GMW Protocol

- Multi-party
- Multi-round

Garbled Circuit

- Constant Round
- Two Party

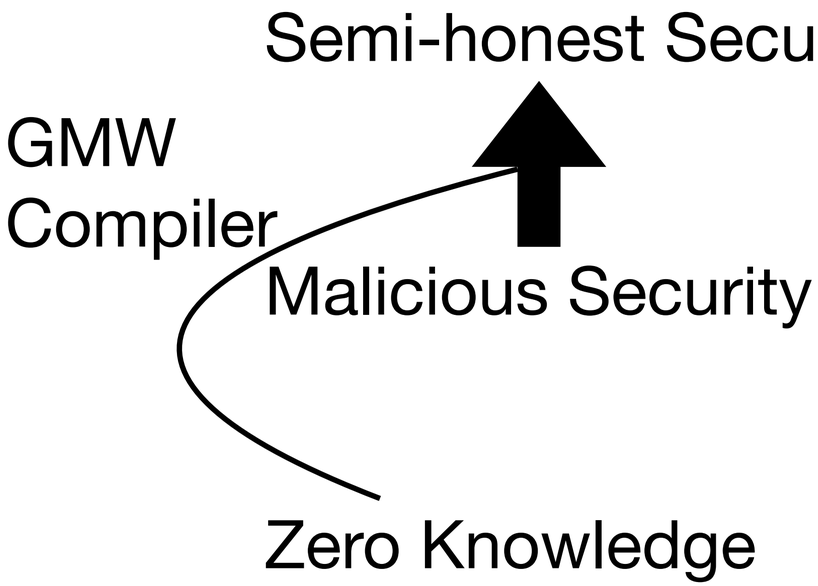
Primitives

Oblivious Transfer

Pseudorandom functions/encryption

Commitments

Setting



General-Purpose Tools

GMW Protocol

- Multi-party
- Multi-round

Garbled Circuit

- Constant Round
- Two Party

Primitives

Oblivious Transfer

Pseudorandom functions/encryption

Commitments

Today's objectives

Introduce Zero Knowledge Proofs

See a feasibility result for ZK of arbitrary statements in NP

Discuss how to upgrade semi-honest protocols to malicious