Zero knowledge proof

- **Problem**: I know that P 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 imply P is true
- **Example**: P : 26781 is not a prime since $26781 = 113 \times 237$
- Given this factorization, other than that you are convinced that P is true, you gained some knowledge (the factorization)
- In a *Zero Knowledge Proof*, Alice will prove to Bob that a statement P is true. Bob will be completely convinced that P is true, but will not learn anything as a result of this process. That is, Bob will gain zero knowledge

Bob被确信P是正确的，但并未获得更多知识。。

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". Alice can prove in *zero knowledge* that she followed the instructions.

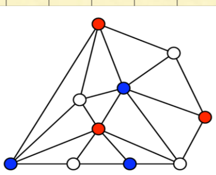
Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design

(Extended Abstract)

Oded Goldreich	*Silvio Micali*	*Avi Wigderson*
Dept. of Computer Sc.	Lab. for Computer Sc.	Inst. of Math. and CS
Technion	MIT	Hebrew University
Haifa, Israel	Cambridge, MA 02139	Jerusalem, Israel

- *Identification scheme*. How should Alice prove to Bob that she is who she claimed to be? For example, how to design a control access system to the CSE dept.?
- A direct solution is to have a box on the door and give authorized people a secret PIN number. However, a drawback is that the box remains outside all the time and if someone could examine the box, they would perhaps be able to view its memory and extract the secrets keys of all people.
- **Ideas using ZKPs**:
  - Let the box contain an *instance* of a hard problem.
  - Give the authorized people the *solution* to the instance.
  - The authorized people will *prove* to the box that they know the solution in zero knowledge.

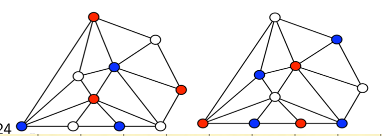
An example



- Alice knows how to 3-color a graph: no two adjacent vertices have the same color; this is an NPC problem.
  - can impress your friends
  - useful for identification

- How can Alice convince Bob that she can 3-color the graph without
  - letting him steal her work?
  - letting him impersonate her?
  - Bob is convinced that Alice can do this.
  - Bob has no idea how to do it himself.

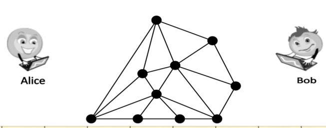
Alice may permute the vertex colors.



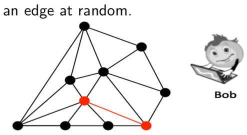
24

每次Alice都要重排

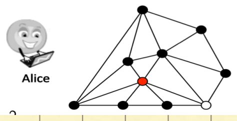
- Alice then encrypts all vertex colors (one key per vertex), and sends the graph to Bob.



- Bob picks an edge at random.



Alice reveals colors of those two keys.



- Repeat as much as needed:
  - Alice permutes graph coloring
  - Alice enrypts all vertices with distinct keys
  - Alice sends permuted encrypted colors to Bob
  - Bob picks an edge
  - Alice sends keys for two vertices
  - Bob checks whether these two colors are distinct

If Alice is lying, with probability $\frac{1}{|E|}$ she will be caught.
If she is telling the truth, she will never be caught.
After k repetitions, the probability she fools Bob is $(1 - \frac{1}{|E|})^k$.

- What does Bob see?
  - randomly-generated keys
  - randomly-generated colors

Because Bob could have generated those keys and colors by himself, he learns nothing from the graph coloring.

Claim. Every NP-statement can be proven in zero-knowledge.

Zero knowledge proof

- A standard mathematical proof is like:

  Given *axioms* and *inference rules*, we give the proof for P that derives P from the axioms using the inference rules.

- A proof system is *sound* if you can never derive **false** statements using it.

  A proof system is *complete* if you can prove all **true** statements using it.

- *Interactive probabilistic proofs*: the verifier does not convince with absolute certainty that P is true, but with high certainty.

soundness 健壮性：证明系统无法得出错误结论
completeness 完备性：证明系统能证得所有正确结论

在zero knowledge proof system 中需保证
- soundness
- completeness
- zero knowledge

Interactive Probabilistic Proofs

- *Interactive probabilistic proofs*: the verifier does not convince with absolute certainty that P is true, but with high certainty.

  No matter what the prover does, and how she tries to to cheat, if the statement P is false, she will fail with this probability.

- **Example.** Alice can distinguish between Coke and Pepsi: Alice turns her back, Bob flips a coin and puts either Coke and Pepsi into a paper cup according the result, Alice tastes and announces whether she thinks it was Coke or Pepsi.
- If they repeat this k times, and Alice always answers correctly, then Bob can conclude with $1 - 2^{-k}$ probability that she really can tell the difference.

Protocol QR

Recall If n is an integer, then $x \in \mathbb{Z}_n^*$ is a *quadratic residue* modulo n if there is some s such that $x = s^2 \pmod{n}$.

It is believed to be hard to tell whether x is a QR modulo n without knowing the factorization of n.

Some useful **facts**:

◇ if n is prime, then $\mathbb{Z}_n^*$ has a generator g and x is a QR iff $x = g^i$ for an even i.

◇ All the QRs form a *group*. If x is a QR, and y is a random QR, then xy is a random QR. For every $z \in QR_n$,

$$\Pr[xy = z] = 1/|QR_n|.$$

Statement P: x is a QR modulo n
Public input: x, n; Prover – Alice; Verifier – Bob
Prover's private input: w such that $x = w^2 \pmod{n}$

$P \to V$: Alice chooses random $u \leftarrow_R \mathbb{Z}_n^*$ and sends $y = u^2$ to Bob
$P \leftarrow V$: Bob chooses $b \leftarrow_R \{0,1\}$
$P \to V$: If $b = 0$, Alice sends u to Bob. If $b = 1$, Alice sends $w \cdot u$.

Verification: Let z denote the number sent by Alice.
Bob *accepts* the proof in the case $b = 0$, $z^2 = y \pmod{n}$, and in the case $b = 1$, $z^2 = xy \pmod{n}$.

We will analyze this protocol in *completeness, soundness, zero knowledge*.

Completeness

Completeness: Whenever x is really a QR, Alice is given s such that $x = s^2 \pmod{n}$, and Alice and Bob follow the protocol, then Bob will *accept* the proof with probability 1.

Alice : w, u

Bob: $z = \begin{cases} u, & b=0 \\ wu, & b=1 \end{cases}$ $\Rightarrow z^2 = u^2 = y$
$\Rightarrow z^2 = (wu)^2 = w^2u^2 = xy$

Saundness

Soundness: If x is not a QR, then regardless of what Alice does, Bob will reject the proof with probability at least $1/2$.

Alice may not follow the instructions in this protocol, and may possibly cheat. We model her strategy as a function P^*. We think of P^* as follows: on input the empty word, it gives a string y, and on input b, it gives a string z.

Lemma 15.1 For every (possibly not efficiently computable) P^*, and (x, n) such that x is not a QR modulo n, we have

$$\Pr_{b \leftarrow \{0,1\}}[out_V\langle P^*, V_{x,b}\rangle = accept] \le 1/2.$$

If two interactive algorithms A and B are running a protocol, we denote this execution by $\langle A, B\rangle$.

$out_A\langle A, B\rangle$ – the output of A after this interaction is finished.
$view_A\langle A, B\rangle$ – the *view* of A in the interaction: messages it received.

Proof. Suppose that $x \notin QR_n$. Denote by y the output of P^* on the empty input.

First, we cannot assume that y is a QR.
Second, y is the output before P^* sees the query b, and then y is independent of b.

Alice会欺骗Bob

Case 1: $y \in QR_n$. That is, $y = u^2 \pmod{n}$. With probability $1/2$, Bob sends $b = 1$. Let $z = P^*(1)$. Bob will accept only if $z^2 = xy$. This is impossible since $z^2y^{-1} = z^2u^{-2} = xyy^{-1} = x$, but $x \notin QR_n$.

Case 2: $y \notin QR_n$. With probability $1/2$, Bob sends $b = 0$. However, if $b = 0$, Alice has to come up with some z such that $z^2 = y$, impossible! Bob will also reject with probability $\ge 1/2$.

Zero knowledge

从证明中学到的知识，自学也能学到

- We think of a possibly cheating verifier V^*. He can only sends either $b = 0$ or $b = 1$. Our goal is to show that: in both cases, he gets a random element in $\mathbb{Z}_n^*$, leaking no info about the QR of x.

  Idea: A proof is *zero knowledge* if whatever Bob learns, he could have learned by himself without any interaction with Alice.

- **Definition 15.2** A prove strategy P is (T, ϵ)-zero knowledge if for every T-time cheating strategy V^* there exists a *poly*(T)-time non-interactive algorithm S (called the *simulator* for V^*) such that for every valid public input x and private input w, the following two random variables are (T, ϵ)-computationally indistinguishable:
  - $view_{V^*}\langle P_{U_m,x,w}, V^*\rangle$, where m is the number of random coins P uses.
  - $S(x)$. Note that S can be probabilistic and so this is a r.v.

  The *simulator* S only gets the public input and has no interaction with P, but still manages to output something indistinguishable from whatever V^* learned in the interaction.

- **Lemma 15.3** The prover of Protocol QR is $(\infty, 2^{-|x|})$ - zero knowledge.

  Proof. Let V^* be a possibly cheating verifier. The simulator S will do the following (S can depend on V^*):
  1. **Input**: x, n such that $x \in QR_n$.
  2. Choose $b' \leftarrow_R \{0, 1\}$.
  3. Choose $z \leftarrow_R QR_n$.
  4. If $b' = 0$, compute $y = z^2$. Otherwise ($b' = 1$), compute $y = z^2x^{-1}$.
  5. Invoke V^* on the message y to obtain a bit b.
  6. If $b = b'$, then output $\langle y, z\rangle$. Otherwise, go back to Step 2.

  We do not even know whether this algorithm loops forever or not.

- **Lemma 15.4** In both case $b' = 0$ and $b' = 1$, the message y is a random element in QR_n.

  Proof. In the case $b' = 0$, this is obvious.

  In the case $b' = 1$, $y = x^{-1}z^2$, where z^2 is a random element in QR_n. So is y since $x \in QR_n$.

  This implies that y is independent of b'. We have already known that $b = V^*(y)$ is also independent of b' and hence we have $\Pr[b = b'] = 1/2$. If we run the algorithm for k steps, we will halt with very high probability ($1 - 2^{-k}$).

  Lemma 15.5 The output of the simulator S is distributed identically to the view of V^* in an interaction with an honest prover.

  Proof. For both the prover and the simulator, if $b = 0$, then z is a random root of y; if $b = 1$, then z is a random root of xy.

Schnorr's identification protocol

- Statement P: Alice knows DL of h, w.r.t. g, these are in group $G = \mathbb{Z}_p$.

  Public input: g, h; Prover – Alice; Verifier – Bob

  Prover's private input: x such that $h = g^x$

  $P \to V$: Alice chooses random $r \leftarrow_R \mathbb{Z}_p$ and sends $a = g^r$ to Bob

  $P \leftarrow V$: Bob chooses $b \leftarrow_R \mathbb{Z}_p$ and sends b to Alice

  $P \to V$: Alice sends $c = r + xb \pmod p$ to Bob.

  Verification: Bob verifies that $ah^b = g^c$.

- *Completeness*: obvious

  Proof of knowledge: if $b \neq b'$ then given a and $b \neq b'$ and $c \neq c'$ such that $ah^b = g^c$ and $ah^{b'} = g^{c'}$, we get $h^{b-b'} = g^{c-c'}$. Since we know b and b', we can take this to the power $(b - b')^{-1} \pmod p$ to get an equation $h = g^x$.

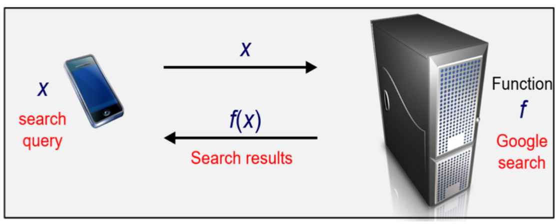
  Honest verifier zero knowledge: The simulator S does the following: choose $b, c \leftarrow_R \mathbb{Z}_p$, choose a as $h^{-b} g^c$.

Homomorphic encryption

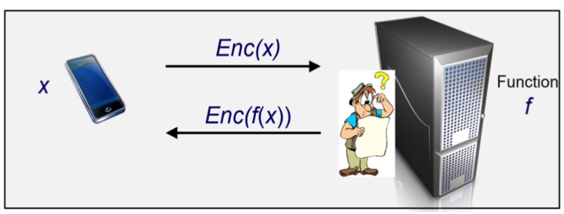
- **Definition 15.6** (*Group homomorphism*)

  Two groups G and G' are *homomorphic* if there exists a function (*homomorphism*) $f : G \to G'$ such that for all $x, y \in G$, $f(x +_G y) = f(x) +_{G'} f(y)$.

  Why do we need *homomorphic encryption*?



想让服务器算出f(x)，但又不想服务器得知x.



若满足全同态，则$f(Enc(x)) = Enc(f(x))$

Computing on encrypted data

- Recall RSA encryption

  $E(m_1) = m_1^e \bmod n$, $E(m_2) = m_2^e \bmod n$

  $E(m_1) \cdot E(m_2) = m_1^e \cdot m_2^e = (m_1 \cdot m_2)^e = E(m_1 \cdot m_2)$

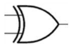
  RSA is multiplicatively homomorphic, but not additively homomorphic.

  We need **both**!

  What people really wanted was the ability to do arbitrary computing on encrypted data, and this requires the abibility to compute both sums and products.

- Why SUMs and PRODUCTs?

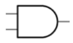
SUM



XOR

$x + y \bmod 2$

PRODUCT



AND

$x \cdot y \bmod 2$

{XOR, AND} is complete, i.e.,
any function is a combination of XOR and AND. (e.g., OR)

Example
$x \; OR \; y = x + y + x \cdot y \bmod 2$.

- Because {XOR, AND} is *complete*, if we can compute SUMs and PRODUCTs on encrypted bits, we can compute any function on encrypted inputs.

Fully-homomorphic encryption!

We can delegate arbitrary processing of data without giving away access to it.

Applications: *private cloud computing, private information retrieval, multi-party secure computation, encrypted search,* ...