CPSC 467: Cryptography and Computer Security

Michael J. Fischer

Lecture 19
November 8, 2017

Zero Knowledge Interactive Proofs (ZKIP)

- ZKIP for graph isomorphism
- Feige-Fiat-Shamir Authentication Protocol
- Abstraction from two ZKIP examples

Information Splitting

Public Key Infrastructure (PKI) and Trust

Zero Knowledge Interactive Proofs (ZKIP)

Zero knowledge interactive proofs (continued)

Last time we saw two examples of zero knowledge interactive proofs:

- Simplified Feige-Fiat-Shamir authentication protocol.
- Secret cave protocol.

We now look at ZKIP's in greater detail.

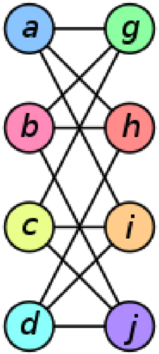
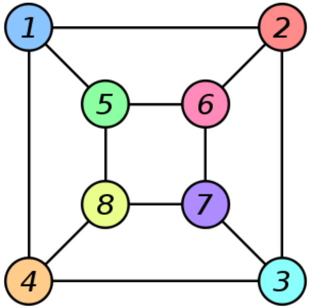
Graph isomorphism problem

Two undirected graphs G and H are said to be *isomorphic* if there exists a bijection π from vertices of G to vertices of H that preserves edges.

That is, $\{x, y\}$ is an edge of G iff $\{\pi(x), \pi(y)\}$ is an edge of H.

The *graph isomorphism problem* is, given graphs G and H, to determine whether or not G and H are isomorphic.

Graph Isomorphism

Graph G	Graph H	Isomorphism π
		$\pi(a) = 1$ $\pi(b) = 6$ $\pi(c) = 8$ $\pi(d) = 3$ $\pi(g) = 5$ $\pi(h) = 2$ $\pi(i) = 4$ $\pi(j) = 7$

From Wikipedia, https://en.wikipedia.org/wiki/Graph_isomorphism

Complexity of graph isomorphism

No polynomial time algorithm for solving graph isomorphism is known, but this problem is also not known to be *NP*-hard.

It follows that there is no known polynomial time algorithm for finding the isomorphism π given two isomorphic graphs G and H. Why?

László Babai claims that graph isomorphism is in *quasipolynomial time*, that is, time of the form

$$2^{O(\log(n)^c)}$$

for some constant c. This is a huge improvement over the best prior result. (See László Babai Graph Isomorphism.)

A zero-knowledge proof for isomorphism

Suppose G_0 and G_1 are public graphs, and Alice knows an isomorphism $\pi : G_0 \rightarrow G_1$.

Using a zero-knowledge proof, Alice can prove to Bob that she knows π *without revealing any information about* π. In particular, she convinces Bob that the graphs really are isomorphic.

However, the proof is *non-transferrable*, so Bob cannot turn around and convince Carol of that fact.

Interactive proof of graph isomorphism

	Alice		Bob
1.	Simultaneously choose a random isomorphic copy H of G_0 and an isomorphism $\tau : G_0 \to H$.		
		$\xrightarrow{H}$	
2.		$\xleftarrow{b}$	Choose random $b \in \{0, 1\}$.
3.	If $b = 0$, let $\sigma = \tau$.		
	If $b = 1$, let $\sigma = \tau \circ \pi^{-1}$.	$\xrightarrow{\sigma}$	Check $\sigma(G_b) = H$.

Validity of isomorphism IP

The protocol is similar to the simplified Feige-Fiat-Shamir protocol

If both Alice and Bob follow this protocol, Bob's check always succeeds.

- When $b = 0$, Alice send τ in step 3, and Bob checks that τ is an isomorphism from G_0 to H.
- When $b = 1$, the function σ that Alice computes is an isomorphism from G_1 to H. This is because π^{-1} is an isomorphism from G_1 to G_0 and τ is an isomorphism from G_0 to H. Composing them gives an isomorphism from G_1 to H, so again Bob's check succeeds.

Isomorphism IP is zero knowlege

The protocol is zero knowledge (at least informally) because all Bob learns is a random isomorphic copy H of either G_0 or G_1 and the corresponding isomorphism.

This is information that he could have obtained by himself without Alice's help.

What convinces him that Alice really knows π is that in order to repeatedly pass his checks, the graph H of step 1 must be isomorphic to *both* G_0 and G_1.

Moreover, Alice knows isomorphisms $\sigma_0 : G_0 \rightarrow H$ and $\sigma_1 : G_1 \rightarrow H$ since she can produce them upon demand.

Hence, she also knows an isomorphism π from G_0 to G_1, since $\sigma_1^{-1} \circ \sigma_0$ is such a function.

Reprise: A simplified one-round FFS protocol

Recall:

- $n = pq$, where p and q are distinct large primes.
- $v \in \mathrm{QR}_n$.
- s is the smallest square root of $v^{-1} \pmod{n}$.
- n and v are public. s is Alice's secret.

FFS protocol:

	Alice		Bob
1.	Choose random $r \in \mathbf{Z}_n$.		
	Compute $x = r^2 \bmod n$.	$\xrightarrow{x}$	
2.		$\xleftarrow{b}$	Choose random $b \in \{0, 1\}$.
3.	Compute $y = rs^b \bmod n$.	$\xrightarrow{y}$	Check $x = y^2 v^b \bmod n$.

Properties of FFS protocol

We make three claims for the FFS protocol.

1. [Completeness] When both Alice and Bob are honest, Bob's check always succeeds.
2. [Soundness] If Mallory attempts to impersonate Alice without knowing her secret s, Bob's check will fail with probability at least $1/2$.
3. [Zero knowledge] Anything that Mallory can compute while interacting with Alice in the FFS protocol could also be computed without Alice's involvement. In particular, if Mallory can find Alice's secret s after running the FFS protocol, then he could have found s without ever talking to Alice.

Completeness

We claimed last time that when both parties are honest, Bob's check

$$x = y^2 v^b \bmod n.$$

succeeds because

$$y^2 v^b \equiv (rs^b)^2 v^b \equiv r^2 (s^2 v)^b \equiv x (v^{-1} v)^b \equiv x \pmod{n}.$$

In a little more detail, we consider the two cases separately:

- $b = 0$: Then $y = r$ and $y^2 \equiv r^2 \equiv x \pmod{n}$.
- $b = 1$: Then $y \equiv rs \pmod{n}$ and $s^2 \equiv v^{-1} \pmod{n}$, so

$$y^2 v \equiv r^2 s^2 v \equiv r^2 v^{-1} v \equiv r^2 \equiv x \pmod{n}.$$

Soundness

Theorem

Suppose Mallory doesn't know a square root of v^{-1}. Then Bob's verification will fail with probability at least 1/2.

Proof.

To successfully fool Bob, Mallory must come up with x in step 1 and y in step 3 satisfying $x = y^2 v^b \bmod n$.

Mallory sends x in step 1 before Bob chooses b, so she does not know which value of b to expect.

When Mallory receives b, she responds by sending a value y_b to Bob.

We consider two cases.

(continued. . .)

Proof: case 1

Proof (continued).

Case 1: There is at least one $b \in \{0, 1\}$ for which y_b *fails* to satisfy

$$x = y^2 v^b \bmod n.$$

Since $b = 0$ and $b = 1$ each occur with probability $1/2$, this means that Bob's verification will fail with probability at least $1/2$, as desired.

(continued...)

Proof: case 2

Proof (continued).

Case 2: y_0 and y_1 both satisfy the verification equation, so $x = y_0^2 \bmod n$ and $x = y_1^2 v \bmod n$.

We can solve these equations for v^{-1} to get

$$v^{-1} \equiv y_1^2 x^{-1} \equiv y_1^2 y_0^{-2} \pmod{n}$$

But then $y_1 y_0^{-1} \bmod n$ is a square root of v^{-1}.

Since Mallory was able to compute both y_1 and y_0, then she was also able to compute a square root of v^{-1}, contradicting the assumption that she doesn't "know" a square root of v^{-1}. □

Successful cheating with probability 1/2

We remark that it *is* possible for Mallory to cheat with success probability $1/2$.

- She guesses the bit b that Bob will send her in step 2 and generates a pair (x, y).
- If she guesses $b = 0$, then she chooses $x = r^2 \bmod n$ and $y = r \bmod n$, just as Alice would have done.
- If she guesses $b = 1$, then she chooses y arbitrarily and $x = y^2 v \bmod n$.

She proceeds to send x in step 1 and y in step 3.

The pair (x, y) is accepted by Bob if Mallory's guess of b turns out to be correct, which will happen with probability $1/2$.

Zero knowledge

We now consider the case of an honest Alice interacting with a dishonest Mallory pretending to be Bob, or simply a dishonest Bob who wants to capture Alice's secret.

Alice would like assurance that her secret is protected if she follows the protocol, regardless of what Mallory (or Bob) does.

Consider what Mallory knows at the end of the protocol.

Mallory sends $b = 0$

Suppose Mallory sends $b = 0$ in step 2.

Then she ends up with a pair (x, y), where y is a random number and x is its square modulo n.

Neither of these numbers depend in any way on Alice secret s, so Mallory gets no direct information about s.

It's also of no conceivable use to Mallory in trying to find s by other means, for she can compute such pairs by herself without involving Alice.

If having such pairs would allow her find a square root of v^{-1}, then she was already able to compute square roots, contrary to the assumption that finding square roots modulo n is difficult.

Mallory sends $b = 1$

Suppose Mallory sends $b = 1$ in step 2.

Now she ends up with the pair (x, y), where $x = r^2 \bmod n$ and $y = rs \bmod n$.

While y might seem to give information about s, observe that y itself is just a random element of $\mathbf{Z}_n$. This is because r is random, and the mapping $r \to rs \bmod n$ is one-to-one for all $s \in \mathbf{Z}_n^*$. Hence, as r ranges through all possible values, so does $y = rs \bmod n$.

Mallory learns nothing from x that she could not have computed herself knowing y, for $x = y^2 v \bmod n$.

Again, all she ends up with is a random number (y in this case) and a quadratic residue x that she can compute knowing y.

Mallory learns nothing from (x, y)

In both cases, Mallory ends up with information that she could have computed without interacting with Alice.

Hence, if she could have discovered Alice's secret by talking to Alice, then she could have also done so on her own, contradicting the hardness assumption for computing square roots.

This is the sense in which Alice's protocol releases zero knowledge about her secret.

FFS authentication and isomorphism IP

We have seen two examples of zero knowledge interactive proofs of knowledge of a secret.

In the simplified Feige-Fiat-Shamir authentication scheme, Alice's secret is a square root of v^{-1}.

In the graph isomorphism protocol, her secret is the isomorphism π.

In both cases, the protocol has the form that Alice sends Bob a "commitment" string x, Bob sends a query bit b, and Alice replies with a response y_b.

Bob then checks the triple (x, b, y_b) for validity.

FFS/Isomorphism comparison (continued)

In both protocols, neither triple $(x, 0, y_0)$ nor $(x, 1, y_1)$ alone give any information about Alice's secret, but y_0 and y_1 can be combined to reveal her secret.

In the FFS protocol, $y_1 y_0^{-1} \bmod n$ is a square root of v^{-1}.
(Note: Since v^{-1} has four square roots, the revealed square root might not be the same as Alice's secret, but it is equally valid as a means of impersonating Alice.)

In the graph isomorphism protocol, $y_1^{-1} \circ y_0$ is an isomorphism mapping G_0 to G_1.

Other materials on zero knowledge

Here are some links to other interesting materials on zero knowledge.

- How to explain zero-knowledge protocols to your children gives a different version of the Secret Cave protocol along with other stories illustrating other aspects of zero knowledge, such as non-transferrability of proof.
- Using a zero-knowledge protocol to prove you can solve a sudoku is a video of a Skype session in which Katie Steckles proves her sudoku-solving ability to Christian Perfect.
- Cryptographic and Physical Zero-Knowledge Proof Systems for Solutions of Sudoku Puzzles is the paper describing the sudoku solution protocol upon which the video above is based.

Information Splitting

Another viewpoint

One way to view these protocols is that Alice splits her secret into two parts, y_0 and y_1.

By randomization, Alice is able to convince Bob that she really has (or could produce on demand) both parts, but in doing so, she is only forced to reveal one of them.

Each part by itself is statistically independent of the secret and hence gives Bob no information about the secret.

Together, they can be used to recover the secret.

Secret splitting

This is an example of *secret splitting* or *secret sharing*, an important topic in its own right. We have already seen other examples of secret sharing.

In the one-time pad cryptosystem, the message m is split into two parts: the key k are the ciphertext $c = m \oplus k$.

Bob, knowing both k and c, recovers m from by computing $c \oplus k$.

Assuming k is picked randomly, then both k and c are uniformly distributed random bit strings, which is why Eve learns nothing about m from k or c alone.

What's different with zero knowledge proofs is that Bob has a way to check the validity of the parts that he gets during the protocol.

The Big Picture

Much of cryptography is concerned with splitting a piece of information s into a collection of *shares* $s_1, \ldots, s_r$.

Certain subsets of shares allow s to be easily recovered; other subsets are insufficient to allow any useful information about s to be easily obtained.

In the simplest form, s is split into two shares a and b. Neither share alone gives useful information about s, but together they reveal s.

Examples of information splitting

- One-time pad: s is broken into a key k and a ciphertext c, where $|k| = |c|$ and $s = k \oplus c$.
- AES: s is broken into a short key k and a long ciphertext c, where $s = D_k(c)$.
- Secret splitting: s is broken into equal-length *shares* s_1 and s_2, where $s = s_1 \oplus s_2$.

Public Key Infrastructure (PKI) and Trust

Share distribution

A key problem (pun intended) in any use of cryptography is how the various parties to a protocol obtain their respective shares.

For conventional symmetric cryptography, this is known as the *key distribution problem*.

For public key systems, the public shares are provided through a *public key infrastructure (PKI)*.

Desired properties of a PKI

A PKI should allow any user to obtain the correct public key (and perhaps other information) for a user.

The information provided must be correct.

The user must trust that it is correct.

Centralized PKI

The first idea for a PKI is a centralized database run by a trusted 3rd party, e.g., the government.

Problems:

- Centralized systems are brittle.
- Difficult to find a single entity that is universally trusted.

Hierarchy of trust

The most widely used PKI today is based on *X.509 certificates* and the *hierarchy of trust*.

A certificate is a package of information that binds a user to a public key.

To be *trusted*, a certificate must be signed by a trusted *certificate authority (CA)*.

To validate the signature, one must obtain and validate a trusted certificate of the signing CA.

Where does trust stop?

The roots of the PKI hierarchy are trusted CA's that are well-known.

Your browser is distributed with trusted certificates for the root CA's.

Any other certificate can be valided by obtaining a *chain of trust* leading to a root certificate.

For this scheme to work, one relies on the CA's to take reasonable care not to issue bogus certificates.

Recent revelations show that not all root CA's are trustworthy. Even a single bad root CA can break the entire system.

Web of trust

A variant PKI is the *web of trust*.

Here, the trust relationship is a graph, not a tree.

The basic rule is to trust a certificate if it is signed by one or more trusted parties.

Anyone can act as a CA, so one must only trust the signatures of trustworthy signers.