CPSC 467: Cryptography and Computer Security

Michael J. Fischer

Lecture 17
October 30, 2013

Zero Knowledge Interactive Proofs (ZKIP)
    Secret cave protocol
    ZKIP for graph isomorphism
    Abstraction from two ZKIP examples

Public Key Infrastructure (PKI) and Trust

Formalizing Zero Knowledge
    Computational Knowledge

Zero Knowledge Interactive Proofs (ZKIP)

Zero knowlege interactive proofs (ZKIP)

A round of the simplified Feige-Fiat-Shamir protocol is an example
of a so-called *zero-knowledge interactive proof*.

These are protocols where Bob provably learns nothing about
Alice's secret.

Here, "learns" means computational knowledge: Anything that
Bob could have computed with the help of Alice he could have
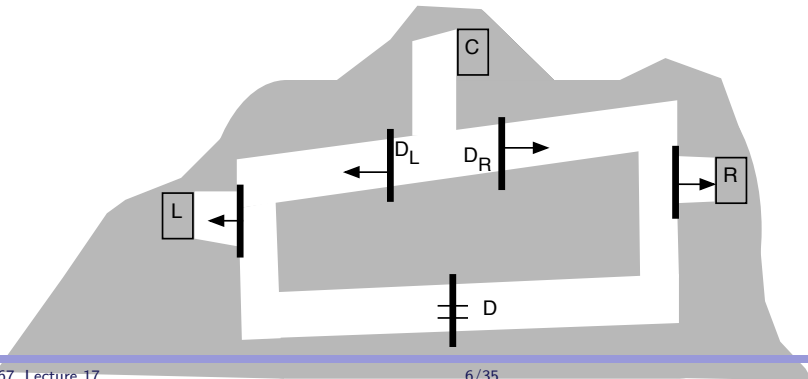computed by himself without Alice's help.

We now consider zero knowledge proofs in greater detail.

The Secret Cave Protocol

The secret cave protocol

The secret cave protocol illustrates the fundamental ideas behind zero knowledge without any reference to number theory or hardness of computation.

Image a cave with tunnels and doors as shown below.

Secret cave protocol (cont.)

There are three openings to the cave: L, C, and R.

L and R are blocked by exit doors, like at a movie theater, which can be opened from the inside but are locked from the outside. The only way into the cave is through passage C.

The cave itself consists of a U-shaped tunnel that runs between L and R. There is a locked door D in the middle of this tunnel, dividing it into a left part and a right part.

A short tunnel from C leads to a pair of doors D_L and D_R through which one can enter left and right parts of the cave, respectively. These doors are also one-way doors that allow passage from C into either the left or right parts of the cave, but once one passes through, the door locks behind and one cannot return to C.

Alice's proposition

Alice approaches Bob, tells him that she has a key that opens door D, and offers to sell it to him.

Bob would really like such a key, as he often goes into the cave to collect mushrooms and would like easy access to both sides of the cave without having to return to the surface to get into the other side.

However, he doesn't trust Alice that the key really works, and Alice doesn't trust him with her key until she gets paid.

Their conversation

Bob tells Alice.

"Give me the key so I can go down into the cave and try it to make sure that it really works."

Alice retorts,

"I'm not that dumb. If I give you the key and you disappear into the cave, I'll probably never see either you or my key again. Pay me first and then try the key."

Bob answers,

"If I do that, then you'll disappear with my money, and I'm likely to be stuck with a non-working key."

How do they resolve their dilemma?

They think about this problem for awhile, and then Alice suggests,

> "Here's an idea: I'll enter the cave through door C, go into the left part of the cave, open D with my key, go through it into the right part of the cave, and then come out door R. When you see me come out R, you'll know I've succeeded in opening the door."

Bob thinks about this and then asks,

> "How do I know you'll go into the left part of the cave? Maybe you'll just go into the right part and come out door R and never go through D."

Alice's plan

Alice says,

> "OK. I'll go into either the left or right side of the cave.
> You'll know I'm there because you'll hear door D_L or D_R
> clank when it closes behind me. You then yell down into
> the cave which door you want me to come out—L or
> R—and I'll do so. If I'm on the opposite side from what
> you request, then I'll have no choice but to unlock D in
> order to pass through to the other side."

Bob's hesitation

Bob is beginning to be satisfied, but he hesitates.

> "Well, yes, that's true, but if you're lucky and happen to
> be on the side I call out, then you don't have to use your
> key at all, and I still won't know that it works."

Alice answers,

> "Well, I might be lucky once, but I surely won't be lucky
> 20 times in a row, so I'll agree to do this 20 times. If I
> succeed in coming out the side you request all 20 times,
> do you agree to buy my key?"

Bob agrees, and they spend the rest of the afternoon climbing in
and out of the cave and shouting.

ZKIP for graph isomorphism

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.

No known polynomial time algorithm decides, given two graphs G and H, whether they are isomorphic, 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?

A zero-knowledge proof for isomorphism

Now, suppose G_0 and G_1 are public graphs and Alice knows an isomorphism $\pi : G_0 \to G_1$.

There is a zero knowledge proof whereby Alice can convince Bob that she knows an isomorphism π from G_0 to G_1, without revealing any information about π.

In particular, she can convince Bob that the graphs really are isomorphic, but 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.

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.

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.

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.

Outline	ZKIP	PKI	Formalizing ZK
	●●●●●●●●●●●●●●●●●●●○●○		○○○

Abstraction

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.

Public Key Infrastructure (PKI) and Trust

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 $s = s_1 \oplus s_2$.

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.

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.

Formalizing Zero Knowledge

Outline
ZKIP
0000000000000000000
PKI
Formalizing ZK
●○○

Computational Knowledge

Computational Knowledge

What does Bob learn from Alice?

We have seen several examples of zero knowledge proofs but no careful definition of what it means to be "zero knowledge".

The intuition that "Bob learns nothing from Alice" surely isn't true.

After running the FFS protocol, for example, Bob learns the quadratic residue x that Alice computed in the first step.

He didn't know x before, nor did he and Alice know any quadratic residues in common other than the public number v.

By zero knowledge, we want to capture the notion that Bob learns nothing that might be useful in turning an intractable computation into a tractable one.

A general client process for interacting with Alice

Consider an arbitrary algorithm for performing some computation, i.e., suppose Mallory is trying to compute some function $f(z)$.

We regard Mallory as a probabilistic Turing machine with input tape and output tape.

- z is placed on the input tape at the beginning.
- If Mallory halts, the contents of the output tape is the answer.
- Mallory can also play Bob's role in some zero-knowledge protocol, say FFS for definiteness.
- During the computation, Mallory can read the number x that Alice sends at the start of FFS.
- Later, he can send a bit b to Alice.
- Later still, he can read the response y from Alice.
- After that, he computes and produces the answer, $f(z)$.