# CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture 18
April 2, 2013

## 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.

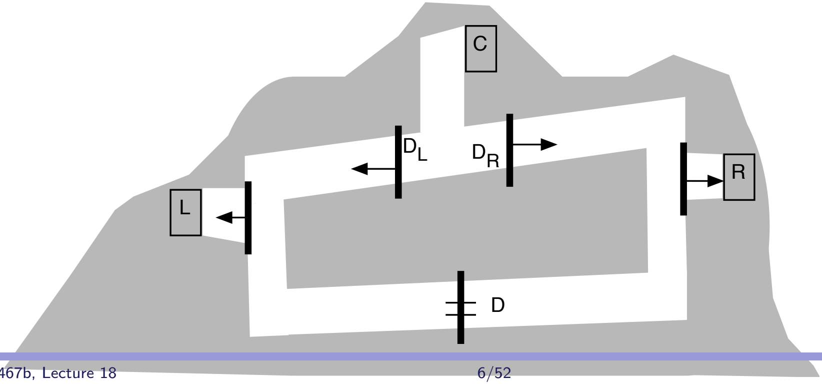
We now consider zero knowledge proofs in greater detail.

## 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 $\pi$ 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 $\pi$ 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 $\pi$ from $G_0$ to $G_1$, without revealing any information about $\pi$.

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 $\tau$ in step 3, and Bob checks that $\tau$ is an isomorphism from $G_0$ to $H$.
- When $b = 1$, the function $\sigma$ that Alice computes is an isomorphism from $G_1$ to $H$. This is because $\pi^{-1}$ is an isomorphism from $G_1$ to $G_0$ and $\tau$ 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 $\pi$ 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 $\pi$ 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 $\pi$.

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 $G_1$.

## 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

## 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, $f(z)$.

## A Mallory-simulator

A Mallory-simulator, whom we'll call Sam, is a program like Mallory except he is not on the internet and can't talk to Alice.

Alice's protocol is *zero knowledge* if for every Mallory, there is a Mallory-simulator Sam that computes the same random function $f(z)$ as Mallory.

In other words, whatever Mallory does with the help of Alice, Sam can do alone.

## The logical connection with knowledge

If Mallory computes some function with Alice's help (such as writing a square root of $v$ to the output tape), then Sam can also do that without Alice's help.

Under the assumption that taking square roots is hard, Sam couldn't do that; hence Mallory also couldn't do that, even after talking with Alice.

We conclude that Alice doesn't release information that would help Mallory to compute her secret; hence her secret is secure.

## Constructing a simulator

To show a particular interactive protocol is zero knowledge, it is necessary to show how to construct Sam for an arbitrary program Mallory.

Here's a sketch of how to generate a triple $(x, b, y)$ for the FFS protocol.

- $b = 0$: Sam generates $x$ and $y$ the same way Alice does—by taking $x = r^2 \bmod n$ and $y = r \bmod n$.
- $b = 1$: Sam chooses $y$ at random and computes $x = y^2 v \bmod n$.

What he can't do (without knowing Alice's secret) is to generate both triples for the same value $x$.

## A simulator for FFS

Here's the code for Sam:

1. Simulate Mallory until he requests a value from Alice.
2. Save Mallory's state as $Q$.
3. Choose a random value $\hat{b} \in \{0, 1\}$.
4. Generate a valid random triple $(x, \hat{b}, y)$.
5. Pretend that Alice sent $x$ to Mallory.
6. Continue simulating Mallory until he is about to send a value $b$ to Alice.
7. If $b \neq \hat{b}$, reset Mallory to state $Q$ and return to step 3.
8. Otherwise, continue simulating Mallory until he requests another value from Alice. Pretend that Alice sent him $y$ and continue.
9. Continue simulating Mallory until he halts.

## Properties of the simulator

The probability that $b = \hat{b}$ in step 7 is $1/2$; hence, the expected number of times Sam executes lines 3–7 is only 2.

Sam outputs the same answers as Mallory with the same probability distribution. Requires some work to show.

Hence, the FFS protocol is zero knowledge.

Note that this proof depends on Sam's ability to generate triples of both kinds without knowing Alice's secret.

## Composing Zero-Knowledge Proofs

## Serial composition

One round of the simplified FFS protocol has probability 0.5 of error. That is, Mallory can fool Bob half the time.

This is unacceptably high for most applications.

Repeating the protocol $t$ times reduces error probability to $1/2^t$.

Taking $t = 20$, for example, reduces the probability of error to less than on in a million.

The downside of such *serial repetition* is that it also requires $t$ round trip messages between Alice and Bob (plus a final message from Alice to Bob).

## Parallel composition of zero-knowledge proofs

One could run $t$ executions of the protocol in parallel.

Let $(x_i, b_i, y_i)$ be the messages exchanged during the $i^{\text{th}}$ execution of the simplified FFS protocol, $1 \leq i \leq t$.

In a *parallel execution*,

- Alice sends $(x_1, \ldots, x_t)$ to Bob,
- Bob sends $(b_1, \ldots, b_t)$ to Alice,
- Alice sends $(y_1, \ldots, y_t)$ to Bob,
- Bob checks the $t$ sets of values he has received and accepts only if all checks pass.

## Simulation proof does not extend to parallel execution

A parallel execution is certainly attractive in practice, for it reduces the number of round-trip messages to only $1\frac{1}{2}$.

The downside is that the resulting protocol may not be zero knowledge by our definition.

Intuitively, the important difference is that Bob gets to know all of the $x_i$'s before choosing the $b_i$'s.

## Problem extending the simulator to the parallel case

While it seems implausible that this would actually help a cheating Bob to compromise Alice secret, the simulation proof used to show that a protocol is zero knowledge no longer works.

To extend the simulator construction to the parallel composition:

- First Sam would have to guess $(\hat{b}_1, \ldots \hat{b}_t)$.
- He would construct the $x_i$'s and $y_i$'s as before.
- When Mallory's program reaches the point that Mallory generates the $b_i$'s, the chance is very high that Sam's initial guesses were wrong and he will be forced to start over again. Indeed, the probability that all $t$ initial guesses are correct is only $1/2^t$.

## Full Feige-Fiat-Shamir Authentication Protocol

## Full FFS overview

The full Feige-Fiat-Shamir Authentication Protocol combines ideas of serial and parallel execution to get a protocol that exhibits some of the properties of both.

A *Blum prime* is a prime $p$ such that $p \equiv 3 \pmod 4$.

A *Blum integer* is a number $n = pq$, where $p$ and $q$ are Blum primes.

If $p$ is a Blum prime, then $-1 \in \mathrm{QNR}_p$, so $\left(\frac{-1}{p}\right) = -1$. This follows from the Euler criterion, since $\frac{p-1}{2}$ is odd, so

$$(-1)^{\frac{p-1}{2}} = \left(\frac{-1}{p}\right) = -1.$$

If $n$ is a Blum integer, then $-1 \in \mathrm{QNR}_n$ but $\left(\frac{-1}{n}\right) = 1$.

## Square roots of Blum integers

Let $n = pq$ be a Blum integer and $a \in \mathrm{QR}_n$. Exactly one of $a$'s four square roots modulo $n$ is a quadratic residue.

Consider $\mathbf{Z}_p^*$ and $\mathbf{Z}_q^*$. $a \in \mathrm{QR}_p$ and $a \in \mathrm{QR}_q$.

Let $\{b, -b\} = \sqrt{a} \pmod{p}$ and apply the Euler Criterion to both. Since

$$(-1)^{(p-1)/2} = -1 \quad \text{and} \quad b^{(p-1)/2} \in \{\pm 1\},$$

then either $b^{(p-1)/2} = 1$ or $(-b)^{(p-1)/2} = 1$.
Hence, either $b \in \mathrm{QR}_p$ or $-b \in \mathrm{QR}_p$. Call that number $b_p$.

Similarly, one of the square roots of $a \pmod{q}$ is in $\mathrm{QR}_q$, say $b_q$.

Applying the Chinese Remainder Theorem, it follows that exactly one of $a$'s four square roots modulo $n$ is a quadratic residue.

## Full FFS key generation

Here's how Alice generates the public and private keys of the full FFS protocol.

- She chooses a Blum integer $n$.
- She chooses random numbers $s_1, \dots, s_k \in \mathbf{Z}_n^*$ and random bits $c_1, \dots, c_k \in \{0, 1\}$.
- She computes $v_i = (-1)^{c_i} s_i^{-2} \bmod n$, for $i = 1, \dots, k$.
- She makes $(n, v_1, \dots, v_k)$ public and keeps $(n, s_1, \dots, s_k)$ private.

Notice that every $v_i$ is either a quadratic residue or the negation of a quadratic residue.

It is easily shown that all of the $v_i$ have Jacobi symbol 1 modulo $n$.

## One round of the full FFS authentication protocol.

A round of the protocol itself is shown below. The protocol is repeated for a total of $t$ rounds.

| | Alice | | Bob |
|---|---|---|---|
| 1. | Choose random $r \in \mathbf{Z}_n - \{0\}$, $c \in \{0,1\}$. $x = (-1)^c r^2 \bmod n$ | $\xrightarrow{x}$. | |
| 2. | | $\xleftarrow{b_1,\ldots,b_k}$ | Choose random $b_1, \ldots, b_k \in \{0,1\}$. |
| 3. | $y = r s_1^{b_1} \cdots s_k^{b_k} \bmod n$. | $\xrightarrow{y}$ | |
| 4. | | | $z = y^2 v_1^{b_1} \cdots v_k^{b_k} \bmod n$. Check $z \equiv \pm x \pmod{n}$ and $z \neq 0$. |

## Correctness of full FFS authentication protocol

When both Alice and Bob are honest, Bob computes

$$z = r^2(s_1^{2b_1} \cdots s_k^{2b_k})(v_1^{b_1} \cdots v_k^{b_k}) \bmod n.$$

Since $v_i = (-1)^{c_i} s_k^{-2}$, it follows that $s_i^2 v_i = (-1)^{c_i}$. Hence,

$$\begin{aligned} z &\equiv r^2(s_1^2 v_1)^{b_1} \cdots (s_k^2 v_k)^{b_k} \\ &\equiv x(-1)^c(-1)^{c_1 b_1} \cdots (-1)^{c_k b_k} \equiv \pm x \pmod{n}. \end{aligned}$$

Moreover, since $x \neq 0$, then also $z \neq 0$. Hence, Bob's checks succeed.

The chance that a bad Alice can fool Bob is only $1/2^{kt}$. The authors recommend $k = 5$ and $t = 4$ for a failure probability of $1/2^{20}$.

## Zero knowledge property

Theorem

*The full FFS protocol is a zero knowledge proof of knowledge of the* $s_j$ *for* $k = O(\log \log n)$ *and* $t = O(\log n)$.

Proof.

See U. Fiege, A. Fiat,and A. Shamir, *Zero knowledge proofs of identity*, ACM Symp. on Theory of Computing, 1987, 210–217. □