CPSC 467: Cryptography and Security

Michael J. Fischer

Lecture 18
November 3, 2020

Authentication While Preventing Impersonation

- Challenge-response authentication protocols

Zero Knowledge

- Secret cave protocol
- ZKIP for graph isomorphism
- Abstraction from two ZKIP examples

Authentication While Preventing Impersonation

Preventing impersonation

A fundamental problem with all of the password authentication schemes discussed so far is that Alice reveals her secret to Bob every time she authenticates herself.

This is fine when Alice trusts Bob but not otherwise.

After authenticating herself once to Bob, then Bob can masquerade as Alice and impersonate her to others.

Authentication requirement

When neither Alice nor Bob trust each other, there are two requirements that must be met:

1. Bob wants to make sure that an impostor cannot successfully masquerade as Alice.
2. Alice wants to make sure that her secret remains secure.

At first sight these seem contradictory, but there are ways for Alice to prove her identity to Bob without compromising her secret.

Challenge-Response Authentication Protocols

Challenge-response authentication protocols

In a challenge-response protocol, Bob presents Alice with a challenge that only the true Alice (or someone knowing Alice's secret) can answer.

Alice answers the challenge and sends her answer to Bob, who verifies that it is correct.

Bob learns the response to his challenge but Alice never reveals her secret.

If the protocol is properly designed, it will be hard for Bob to determine Alice's secret, even if he chooses the challenges with that end in mind.

Challenge-response protocol from a signature scheme

A challenge-response protocol can be built from a digital signature scheme (S_A, V_A).

(The same protocol can also be implemented using a symmetric cryptosystem with shared key k.)

	Alice		Bob
1.		$\xleftarrow{r}$	Choose random string r.
2.	Compute $s = S_A(r)$	$\xrightarrow{s}$	Check $V_A(r, s)$.

Requirements on underlying signature scheme

This protocol exposes Alice’s signature scheme to a chosen plaintext attack.

A malicious Bob can get Alice to sign any message of his choosing.

Alice had better have a different signing key for use with this protocol than she uses to sign contracts.

While we hope our cryptosystems are resistant to chosen plaintext attacks, such attacks are very powerful and are not easy to defend against.

Anything we can do to limit exposure to such attacks can only improve the security of the system.

Limiting exposure to chosen plaintext attack: try 1

We explore some ways that Alice might limit Bob's ability to carry out a chosen plaintext attack.

Instead of letting Bob choose the string r for Alice to sign, r is constructed from two parts, r_1 and r_2.

r_1 is chosen by Alice; r_2 is chosen by Bob. Alice chooses first.

	Alice		Bob
1.	Choose random string r_1	$\xrightarrow{r_1}$	
2.		$\xleftarrow{r_2}$	Choose random string r_2.
3.	Compute $r = r_1 \oplus r_2$		Compute $r = r_1 \oplus r_2$
4.	Compute $s = S_A(r)$	$\xrightarrow{s}$	Check $V_A(r, s)$.

Problem with try 1

The idea is that neither party should be able to control r.

Unfortunately, that idea does not work here because Bob gets r_1 before choosing r_2.

Instead of choosing r_2 randomly, a cheating Bob can choose $r_2 = r \oplus r_1$, where r is the string that he wants Alice to sign.

Thus, try 1 is no more secure against chosen plaintext attack than the original protocol.

Limiting exposure to chosen plaintext attack: try 2

Another possibility is to choose the random strings in the other order—Bob chooses first.

	Alice		Bob
1.		$\xleftarrow{r_2}$	Choose random string r_2.
2.	Choose random string r_1	$\xrightarrow{r_1}$	
3.	Compute $r = r_1 \oplus r_2$		Compute $r = r_1 \oplus r_2$
4.	Compute $s = S_A(r)$	$\xrightarrow{s}$	Check $V_A(r, s)$.

Try 2 stops chosen plaintext attack

Now Alice has complete control over r.

No matter how Bob chooses r_2, Alice's choice of a random string r_1 ensures that r is also random.

This thwarts Bob's chosen plaintext attack since r is completely random.

Thus, Alice only signs random messages.

Problem with try 2

Unfortunately, try 2 is totally insecure against active eavesdroppers. Why?

Suppose Mallory listens to a legitimate execution of the protocol between Alice and Bob.

From this, he easily acquires a valid signed message (r_0, s_0). How does this help Mallory?

Mallory sends $r_1 = r_0 \oplus r_2$ in step 2 and $s = s_0$ in step 4.

Bob computes $r = r_1 \oplus r_2 = r_0$ in step 3, so his verification in step 4 succeeds.

Thus, Mallory can successfully impersonate Alice to Bob.

Further improvements

Possible improvements to both protocols.

1. Let $r = r_1 \cdot r_2$ (concatenation).
2. Let $r = h(r_1 \cdot r_2)$, where h is a cryptographic hash function.

In both cases, neither party now has full control over r.

This weakens Bob's ability to launch a chosen plaintext attack if Alice chooses first.

This weakens Mallory's ability to impersonate Alice if Bob chooses first.

Zero Knowledge

Concept of zero knowledge

In all of the challenge-response protocols above, Alice releases some partial information about her secret by producing signatures that Bob could not compute by himself.

Zero knowledge protocols allows Alice to prove knowledge of her secret without revealing any information about the secret itself.

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

To paraphrase the famous show tune[1]: "Anything you [Alice] can do, I [Bob] can do better [without your help]".

[1] "Anything You Can Do (I Can Do Better)" is a show tune composed by Irving Berlin for the 1946 Broadway musical Annie Get Your Gun. [Wikipedia]

Basic ideas underlying zero knowledge

A zero knowledge protocol can be thought of as a special kind of challenge-response protocol. Here's how it works:

1. Alice and Bob agree on some global *parameters* p.
2. Alice chooses a *secret* s that is related to p.
3. Alice constructs a *puzzle* x with two *solutions* y_0 and y_1 that she knows how to find using s. Bob can check the solutions but not find them without knowing s.
4. Bob sends Alice a single *challenge* bit $b \in \{0, 1\}$ and asks her to show him solution y_b.
5. Alice responds with the requested solution y_b.
6. Bob checks that y_b is indeed a solution to puzzle x and that x is consistent with the global parameters p.

Security assumptions

- Alice's puzzles must be hard for anyone to solve without knowing s, but easy for anyone to check a solution.
- Bob learns nothing about Alice's secret from seeing just one solution. (If he sees both, the privacy of s is compromised.)
- An impostor Mallory cannot create a puzzle x for which she knows both solutions y_0 and y_1 without also knowing Alice's secret.
- Bob will catch Mallory cheating if he happens to request a solution that Mallory does not know. This will occur with probability $\geq 1/2$.
- Bob repeats this protocol t times to reduce his probability of accepting an impostor to $\leq 1/2^t$.

Feige-Fiat-Shamir authentication protocol

The Feige-Fiat-Shamir authentication protocol uses the number theory of *quadratic residues* to construct puzzles and solutions with the properties required for zero knowledge.

Quadratic residues are a fancy name for square roots in $\mathbf{Z}_n^*$.

We say that r is a *square root* of x modulo n iff $r^2 \equiv x \pmod{n}$. Not all numbers in $\mathbf{Z}_n^*$ have square roots.

The security assumptions of FFS depend on the fact that for a certain subset of numbers in $\mathbf{Z}_n^*$, determining whether or not a given number in that set has a square root is believed to be computationally difficult when n is the product of two distinct primes. This is called the *quadratic residuosity assumption*.

FFS preparation

- Alice chooses $n = pq$, where p and q are distinct large primes.
- Alice picks a random element $s \in \mathbf{Z}_n^*$ and computes $v = s^{-2} \bmod n$.
- She makes n and v public and keeps s private.

A simplified one-round FFS protocol

Here's a simplified one-round version.

	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 $y = rs^b \bmod n$.	$\xrightarrow{y}$	If $b = 0$, check $x = y^2 \bmod n$.
			If $b = 1$, check $x = y^2 v \bmod n$.

When both parties are honest, Bob accepts Alice because

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

This holds because

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

Road map to the full FFS authentication protocol

Before presenting the full FFE authentication protocol, we will explore zero knowledge interactive proofs (ZKIPs) and quadratic residues in greater detail, both of which have applications that go beyond the authentication problem.

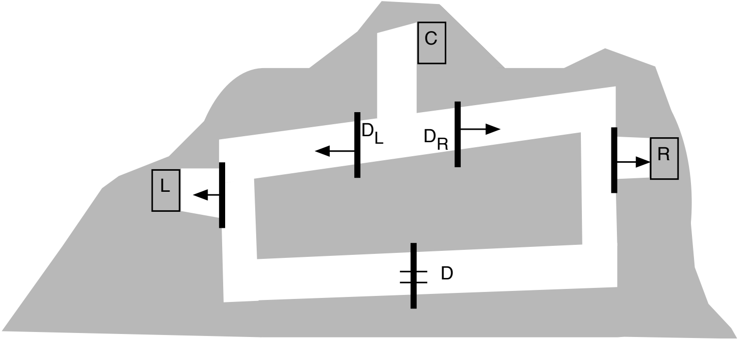
The Secret Cave Protocol

Zero knowledge proofs without number theory

While it might seem that zero knowledge proofs are intimately tied up with number theory, we present a purely physical illustration of zero knowledge, devoid of mathematics or number theory.

The secret cave

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 are one-way doors. 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 a door clank when it closes behind me. You won't know whether I went through D_L or D_R, but that doesn't matter. I'll be stuck in one side of the cave or the other."*
>
> *"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?"*

Agreement finally

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

Zero knowledge interactive proofs (continued)

We have seen 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

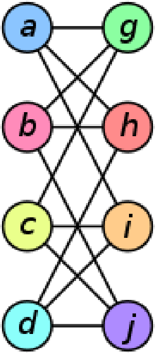
Two undirected graphs G and H are said to be *isomorphic* if there exists a bijection π from vertices of G to vertices of H that preserves edges.

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

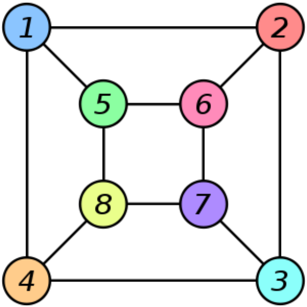
The *graph isomorphism problem* is, given graphs G and H, to determine whether or not G and H are isomorphic.

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

Testing versus finding

No polynomial time algorithm is known for **testing** if two graphs G and H 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?
If there were such a polynomial time algorithm, we could test isomorphism as follows:

> *Given G and H, use A to find an isomorphism π from G to H. If A succeeds, answer "yes"; otherwise answer "no".*

Complexity of graph isomorphism

László Babai claims that the graph isomorphism problem 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. This result is still unverified (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 $\pi : G_0 \to G_1$.

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-transferable*, 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 knowledge

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.

He could have obtained this information 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 $\sigma_0 : G_0 \to H$ and $\sigma_1 : G_1 \to H$ 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^{-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.

Another viewpoint

One way to view zero knowledge 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.

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-transferability 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.