CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture 16
March 19, 2012

Authentication While Preventing Impersonation

- Challenge-response authentication protocols
- Feige-Fiat-Shamir Authentication Protocol

Zero Knowledge Interactive Proofs (ZKIP)

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

Public Key Infrastructure (PKI) and Trust

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.

Feige-Fiat-Shamir Authentication Protocol

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.

The Feige-Fiat-Shamir protocol allows Alice to prove knowledge of her secret *without revealing any information about the secret itself*.

Such protocols are called *zero knowledge*, which we will discuss shortly.

Feige-Fiat-Shamir protocol: overview

Alice authenticates herself by successfully completing several rounds of a protocol that requires knowledge of a secret s.

In a single round, protocol, Bob has at least a 50% chance of catching an impostor Mallory.

By repeating the protocol t times, the error probability (that is, the probability that Bob fails to catch Mallory) drops to $1/2^t$.

This can be made acceptably low by choosing t to be large enough.

For example, if $t = 20$, then Mallory has only one chance in a million of successfully impersonating Alice.

Feige-Fiat-Shamir protocol: preparation

The Feige-Fiat-Shamir protocol is based on the difficulty of computing square roots modulo composite numbers.

- Alice chooses $n = pq$, where p and q are distinct large primes.
- Next she picks a quadratic residue $v \in \mathrm{QR}_n$ (which she can easily do by choosing a random element $u \in \mathbf{Z}_n^*$ and letting $v = u^2 \bmod n$).
- Finally, she chooses s to be the smallest square root of $v^{-1} \pmod{n}$.[1] She can do this since she knows the factorization of n.

She makes n and v public and keeps s private.

[1] Note that if v is a quadratic residue, then so is $v^{-1} \pmod{n}$.

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 $y = rs^b \bmod n$.	$\xrightarrow{y}$	Check $x = y^2 v^b \bmod n$.

When both parties are honest, Bob accepts Alice because

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

This holds because

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

A dishonest Alice

We now turn to the security properties of the protocol when "Alice" is dishonest, that is, when Mallory is attempting to impersonate the real Alice.

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 that Mallory can't successfully cheat

Proof.

In order for Mallory to successfully fool Bob, he 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 he does not know which value of b to expect.

When Mallory receives b, he 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 he was also able to compute a square root of v^{-1}, contradicting the assumption that he doesn't "know" a square root of v^{-1}. $\square$

Successful cheating with probability 1/2

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

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

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

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

A dishonest Bob

We now consider the case of a dishonest Mallory impersonating 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 (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 he 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 he can compute such pairs by himself without involving Alice.

If having such pairs would allow him find a square root of v^{-1}, then he 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 he 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 $r \rightarrow rs \bmod n$ is one-to-one for all $s \in \mathbf{Z}_n^*$. Hence, as r ranges through all possible values, so does $rs \bmod n$.

What does Mallory learn from x?

Nothing that he could not have computed himself knowing y, for $x = y^2 v \bmod n$.

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

Mallory learns nothing from (x, y)

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

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

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

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.

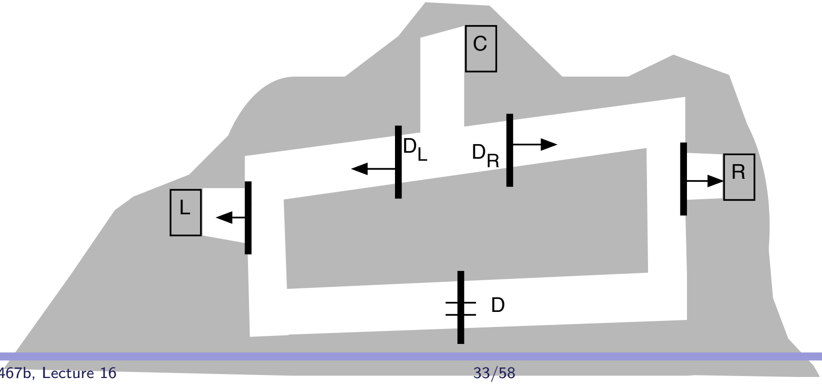
We now consider zero knowledge proofs in greater detail.

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.

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.