CPSC 367: Cryptography and Security

Michael J. Fischer

Lecture 18
April 2, 2019

Authentication Using Passwords (cont.)

Secure password storage

Dictionary attacks

Authentication While Preventing Impersonation

Challenge-response authentication protocols

Authentication using zero knowledge

Zero Knowledge Interactive Proof [ZKIP]

Secret cave protocol

Authentication Using Passwords (cont.)

Secure password storage

Another issue with traditional password authentication schemes is the need to store the passwords on the server for later verification.

- The file in which passwords are store is highly sensitive.
- Operating system protections can (and should) be used to protect it, but they are not sufficient.
- Legitimate sysadmins might use passwords found there to log into users' accounts at other sites.
- Hackers who manage to break into the computer and obtain root privileges can do the same thing.
- Backup copies may not be subject to the same system protections, so someone with access to a backup device might read everybody's password from it.

Storing encrypted passwords

Rather than store passwords in the clear, it is usual to store "encrypted" passwords.

That is, the hash value of the password under some cryptographic hash function is stored instead of the password itself.

Using encrypted passwords

The authentication function

- takes the cleartext password from the user,
- computes its hash value,
- compares the computed hash value with the stored value.

Since the password file does not contain the actual password, and it is computationally difficult to invert a cryptographic hash function, knowledge of the hash value does not allow an attacker to easily find the password.

Dictionary attacks on encrypted passwords

Access to an encrypted password file opens up the possibility of a *dictionary attack*.

Many users choose weak passwords—words that appear in an English dictionary or in other available sources of text.

If one has access to the password hashes of legitimate users on the computer (such as is contained in `/etc/passwd` on Unix), an attacker can hash every word in the dictionary and then look for matches with the password file entries.

Harm from dictionary attacks

A dictionary attack is quite likely to succeed in compromising at least a few accounts on a typical system.

Even one compromised account is enough to allow the hacker to log into the system as a legitimate user, from which other kinds of attacks are possible that cannot be carried out from the outside.

Salt

Salt is a way to make dictionary attacks more expensive.

- *Salt* is a random number that is stored along with the hashed password in the password file.
- The hash function takes two arguments, password and salt, and produces a hash value.
- Because the salt is stored (in the clear) in the password file, the user's password can be easily verified.
- The same password hashes differently depending on the salt.
- A successful dictionary attack now has to encrypt the entire dictionary with every salt value that appears in the password file being attacked.
- This increases the cost of the attack by orders of magnitude.

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.

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 the help of Alice he could have computed by himself without Alice’s help.

Authentication using zero knowledge

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

In a single round of the 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.

Zero Knowledge Interactive Proof [ZKIP]

Zero knowledge using number theory

The Feige-Fiat-Shamir authentication protocol is based on a zero knowledge proof that the client knows a secret number s.

To understand it requires some number theory concerning the difficulty of computing the square root of certain numbers modulo an integer n that is the product of two distinct primes.

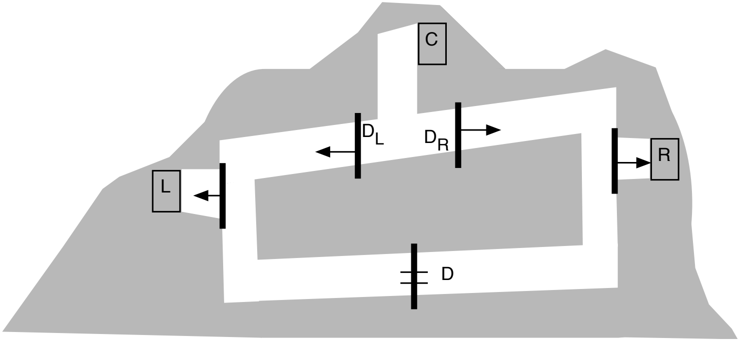
Before presenting FFE authentication and the number theory needed to understand it, we explore the fundamental ideas behind any zero knowledge protocol.

While it might seem that zero knowledge interactive 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 Protocol

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.