Massachusetts Institute of Technology — Handout 24
6.857: Network and Computer Security — October 23, 2001
Professor Ronald L. Rivest

Problem Set 5 Solutions

Problem 5-1. I'm not going to tell

This solution was written by David Bailey, Eric Cholankeril, and Misha Zitser.

Our interactive ZK proof has three rounds as illustrated below:

P		V
$r \; \epsilon_R \; {Z_p}^*$	$u = g^r \ (mod\ p)$	
	$- - - - - - - - - - - - - - - - >$	
	c	$c \; \epsilon_R {Z^*}_p$
	$< - - - - - - - - - - - - - - - -$	
	$v = r - cx \ (mod\ p-1)$	
	$- - - - - - - - - - - - - - - - >$	
		Check: Is $u = g^v y^c$? If the above equality holds, V accepts proof, otherwise he rejects proof.

Completeness
Assume P and V are honest. If P knows an x such that $y = g^x \ (mod\ p)$, then $g^v y^c = g^{r-cx} g^{xc} = g^r = u$. So, with probability 1, V will accept P's proof.

Soundness We wish to show that if an adversary has no efficient algorithm for deriving the secret x (i.e algorithm that runs in probabilistic polynomial time with respect to the size k of p, where $|x| \approx |p| \approx k$), then he will manage to present a convincing proof to the verifier only a negligible fraction of times.

Proof: Suppose the adversary cannot efficiently compute all k bits of x. Let's say the adversary can efficiently compute only n bits out of the total k bits of x, where $0 \leq n < k$. The probability that he chooses the rest of the bits correctly is thus $\frac{1}{2^{k-n}}$. This probability must be negligible. In other words, $\frac{1}{2^{k-n}} < \frac{1}{Q(k)}$ for any polynomial Q and sufficiently large k. If the latter weren't true, the adversary could compute the remaining $k - n$ bits of x in polynomial time, and thus he would have computed all k bits of x in polynomial time, yielding a contradiction.

Zero Knowledge

We assume that the verifier is honest. First, let us show that V can simulate transcripts of valid proofs given by P, without P being present at all. A valid proof's transcript looks like (u, c, v), where u, c and v are such that $u = g^v y^c$. To simulate such a transcript, V first picks some random $v \, \epsilon_R \, \{0, ..., p-2\}$. He then chooses a random $c \, \epsilon_R \, Z^*{}_p$. Finally, he sets $u = g^v y^c \ (mod\, p)$. We now claim that such a simulated transcript is indistinguishable from a transcript of a valid interactive proof that could take place between P and V. Let us compare the simulated transcript (u, c, v) to a real transcript (u_r, c_r, v_r). Since V is honest, we can assume that c_r is independent of u_r, and thus c_r is uniformly distributed over $Z_p{}^*$. Therefore, c and c_r have the same distribution.

u is determined by v and c, both of which are uniform and independent of each other. Therefore, u's distribution corresponds to the distribution of a random variable that's equal to the product of two uniformly distributed random variables.(each distributed over $Z_p{}^*$) So, u and u_r are indistinguishable. Finally, it remains to be shown that v and v_r are indistinguishable. Since the verifier does not know x, $v = r - cx \ (mod\, p-1)$ is indistinguishable from a random element modulo $p-1$. Therefore, v is indistinguishable from v_r. We have shown that the simulated, valid transcripts are indistinguishable from the real valid transcripts. Therefore, it is impossible for the verifier to learn anything about x from his conversations with P that he could not already learn on his own.

(b) Signatures Using the technique demonstrated in lecture, we will replace the verifier with a hash function so that an actual third party verifier is not needed for the prover to sign the message. Once the verifier is removed from the signing process, anyone can verify that the signature is valid.

We first modify the converstation between the prover and verifier to be the following:

P		V
$m = message$		
$x = SecretKey$		
$H =$ MD5 hash fn.		
$r \, \epsilon_R \, Z_p{}^*$	$u = g^r \ (mod\, p),\, m$	
	$- - - - - - - - - - - - - - - - >$	
	$c = H(m, u)$	
	$< - - - - - - - - - - - - - - - -$	
	$v = r - cx \ (mod\, p-1)$	
	$- - - - - - - - - - - - - - - - >$	
		Check: Is $u = g^v y^c$?
		If the above equality holds,
		V accepts proof, otherwise
		he rejects proof.

The scheme still works, since although P can predict what V will choose as a value for c by computing the hash himself, it does not allow him to sign the message without knowing x, since the hash function is one-way. That is to say, there is no way for him to try to cheat by picking c first, and u afterward.

Having shown that this signature scheme is safe, we can simply have P compute the hash himself, and then publish m, $u = g^r$, $c = H(m, u)$, and $v = r - cx$.

(c) Equality of discrete Logs

Our interactive ZK proof has three rounds as illustrated below:

	P	V
$r \in_R Z_p{}^*$	$u = (gh)^r \ (mod\ p)$	
	$- - - - - - - - - - - - - - - - >$	
	c	$c \in_R Z^*{}_p$
	$< - - - - - - - - - - - - - - - -$	
	$v = r - cx \ (mod\ p-1)$	
	$- - - - - - - - - - - - - - - - >$	
		Check: Is $u = (gh)^v (y_1 y_2)^c$?
		If the above equality holds,
		V accepts proof, otherwise
		he rejects proof.

Completeness

Assume P and V are honest. If P knows an x such that $y_1 = g^x \ (mod\ p)$ and $y_2 = h^x \ (mod\ p)$, then $(gh)^v (y_1 y_2)^c = g^{r-cx} h^{r-cx} g^{xc} h^{xc} = g^r h^r = u$. So, with probability 1, V will accept P's proof.

Soundness

We wish to show that if an adversary has no efficient algorithm for deriving the secret x (i.e algorithm that runs in probabilistic polynomial time with respect to the size k of p, where $|x| \approx |p| \approx k$), then he will manage to present a convincing proof to the verifier only a negligible fraction of times.

Proof: Suppose the adversary cannot efficiently compute all k bits of x. Let's say the adversary can efficiently compute only n bits out of the total k bits of x, where $0 \leq n < k$. The probability that he chooses the rest of the bits correctly is thus $\frac{1}{2^{k-n}}$. This probability must be negligible. In other words, $\frac{1}{2^{k-n}} < \frac{1}{Q(k)}$ for any polynomial Q and sufficiently large k. If the latter weren't true, the adversary could compute the remaining $k-n$ bits of x in polynomial time, and thus he would have computed all k bits of x in polynomial time, yielding a contradiction.

Zero Knowledge

We assume that the verifier is honest. First, let us show that V can simulate transcripts of valid proofs given by P, without P being present at all. A valid proof's transcript looks like (u, c, v), where u, c and v are such that $u = (gh)^v (y_1 y_2)^c$. To simulate such a transcript, V first picks some random $v \in_R \{0, ..., p-2\}$. He then chooses a random $c \in_R Z^*{}_p$. Finally, he sets $u = (gh)^v (y_1 y_2)^c \ (mod\, p)$. We now claim that such a simulated transcript is indistinguishable from a transcript of a valid interactive proof that could take place between P and V. Let us compare the simulated transcript (u, c, v) to a real transcript (u_r, c_r, v_r). Since V is honest, we can assume that c_r is independent of u_r, and thus c_r is uniformly distributed over $Z_p{}^*$. Therefore, c and c_r have the same distribution.

Now, we note that g and h are generators of $Z^*{}_p$, but the log base g of h is not known to anyone. Since the verifier does not know the relationship between g and h, he cannot distinguish between $g^r h^r$ and the product of two random elements of $Z_p{}^*$. Or, in other words, the verifier cannot distinguish the distribution of u_r from a distribution obtained by multiplying two random elements of $Z_p{}^*$.

u is determined by v and c, both of which are uniform and independent of each other. Therefore, u's distribution corresponds to the distribution of a random variable that's equal to the product of two uniformly distributed random variables.(each distributed over $Z_p{}^*$) So, u and u_r are indistinguishable. Finally, it remains to be shown that v and v_r are indistinguishable. Since the verifier does not know x, $v = r - cx \ (mod\, p-1)$ is indistinguishable from a random element modulo $p-1$. Therefore, v is indistinguishable from v_r. We have shown that the simulated, valid transcripts are indistinguishable from the real valid transcripts. Therefore, it is impossible for the verifier to learn anything about x from his conversations with P that he could not already learn on his own.

(d) Non-interactive version

A non-interactive version of the protocol in part (c) can be obtained by replacing the verifier with a public one-way hash function. So, instead of receiving a random c from the verifier, the prover will instead generate c himself, by setting $c = h(u) \ (mod\ p)$. The verifier will add an extra step to his checking procedure. In addition to checking that $u = (gh)^v (y_1 y_2)^c$, he will also check that

$h(u) = c$. Since h is one-way, the prover is prevented from cheating by trying to find a particular value of r such that u will hash to a "nice" value.

Problem 5-2. Keystroke Logging Bypass Mechanisms

Patrick Cody, Edward Cotler, Joseph Hastings, and Pavel Langer submitted this solution.

We came up with two different mechanisms to prevent keystroke loggers from compromising passwords.

The first method to guarantee keyboard security is basically a one-way pad for keyboard input. When a user is given a prompt for a password, he is also given a randomly generated conversion table on the screen. This conversion table is dynamically generated for each character, with random reassignments for all keyboard letters to screen letters. The user then enters the correct password according to the corresponding conversion table. For example, if the user's password is "FooBar", then the password the user would type on the keyboard might be something like "Eks9*l" (note that the conversion value of 'o' changes between keystrokes). This mechanism effectively acts as a one-way pad for each keyboard entry, allowing a login program to recover and verify the password while rendering pattern analysis of keyboard strokes useless. In addition, to keep the length of the password secret, the user may type any number of random characters at the end of his password, which the login system would ignore. With the conversion tables changing with each login, with the length of each password randomly padded, and with the keystroke recorder not knowing the contents of the display, pattern analysis of keyboard trends would yield nothing about contents of the password or its length. With this system, a password entered by keyboard would be secure (as long as the contents of the screen are also not known).

A second method to avoid keystroke logging is to not use the keyboard to enter the password. Instead, the password would be entered by using the mouse to click on a virtual keypad on the screen. Given that almost every computer in existence has a mouse already, we feel this bypass is reasonable. To prevent a mouse-logging application (which I have never heard of, but is conceivable) from discovering the password, the virtual keyboard on the screen has its buttons moved around randomly for each instance of a password. Since the only way to recover the password from a record of mouse movements is to know the contents of the screen at the time, a mouse-based password entry system would also be secure from an input-logging system[1].

The reason we listed two methods of bypassing a logging system is that both solutions have different levels of usability. A different system for entering passwords has to be acceptable to users, and multiple approaches should satisfy a larger group of users than any one solution. However, both of our approaches rely on the fact that an intercept program can only record the input device, if both the screen and the input device can be recorded, then we believe that there is no simple solution to the problem.

[1]However, this system would allow someone who records the monitor output to record the password, a definite disadvantage.