18733: Applied Cryptography S17

Homework 5

due May 1, 2017, 11:59pm EST

1 Commitment Schemes [4 points]

Suppose that there exists an online auction where users bid on items, and at the end of the auction, the user who bids the highest amount pays the second highest bid. If every users' bids were publicly available, an adversarial user A who observes the bids everyone has placed, can 'cheat' the auction by submitting \$0.01 higher than the current highest bid when he is willing to pay the hightest bid. To prevent this from happening, you wish to develop an auction which each participant's bid is hidden from everyone else but the auction organizer.

(a) **[1 point]** Describe a simple solution for this problem using techniques that we have studied so far in class. Does your solution work even when the auction organizer is 'dishonest'? If not, how would you modify your scheme work even when the auction organizer cannot be trusted?

Another possible way to solve this problem is through the use of commitment schemes. Commitment schemes are useful in the application of auctions and zero-knowledge proofs.

Let A and B be two parties, and A wants to send some value x (ex. a bid for an action) to B in such a way that it is secret to everyone (including B), but that at a later date may be revealed in such a way that B is convinced A did not change the value between the time of commitment and the time of reveal. These intuitions are captured by the following definitions:

- **Perfectly Hiding:** If B is given a commitment c, then each value of x is equally likely, i.e. given c, B cannot figure out what x is.
- **Computationally Binding:** A sender A should not be able to find an x and x' that produce the same commitment c.

(b) **[1 point]** Describe a solution for the auction problem using commitment schemes.

Now that we have motivated with the use of commitment schemes, we will consider a specific commitment scheme based on discrete log problem.

Let $p = 2q + 1$ for p, q n-bit prime numbers. Also, let G_q be the quadratic residue subgroup of $\mathbb{Z}_p^*$ which we have shown before to have order q, namely $|G_q| = |QR(\mathbb{Z}_p)| = q$. Finally let g be a randomly selected generator of G_q which is trivial to find.

The commitment scheme will begin with the receiver B, selecting $a \leftarrow_R \mathbb{Z}_q$ and computing $h = g^a \ mod\ p$. Then the receiver makes the values (p, q, g, h) public. A user A who wants to commit to a value x, first selects $r \leftarrow_R \mathbb{Z}_q$ and computes $c = g^x h^r \ mod\ p$ and sends c to B. At the time of verification, he later reveals the commitment to B by sending the values of (x, r). B checks the validity of the commitment by computing $c \overset{?}{=} g^x h^r \ mod\ p$.

(c) **[1 point]** Prove that this scheme is perfectly hiding.

(d) **[1 point]** Using a reduction to discrete log, prove that this scheme is computationally binding.

2 TOR Guard Nodes and Attacks [6 points]

Recall the TOR overlay network that we observed in class which is used for anonymizing HTTP and web communication.

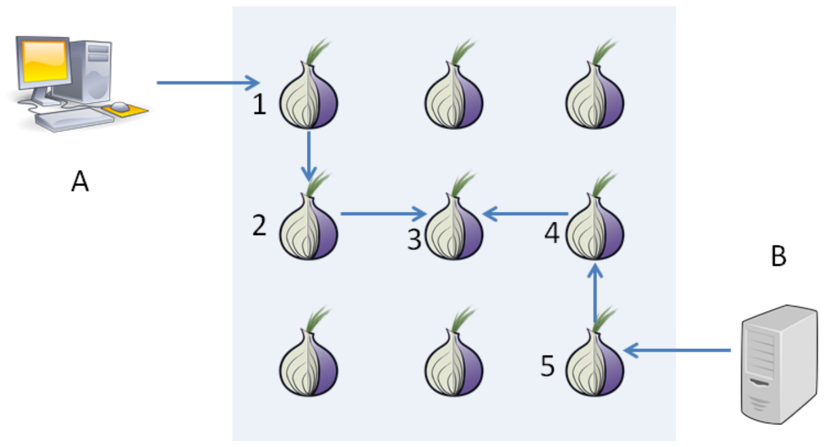

Figure 1: Diagram of a user A connecting to a Tor hidden service (.onion) B using node 3 as a rendezvous point.

Suppose that we have an adversarial client A that wants to discover the IP address of a Tor Hidden Service B. A and B construct a circuit through the Tor network with a rendezvous point at node 3 in order to communicate. The circuit works as follows, nodes 1, 2, 3 behave as we have seen in the example in class, except instead of removing the last layer of encryption and forwarding the message, node 3 will re-encrypt the message and continue to forward it along its eventual path to B.

(a) [**2 points**] Assume that A and B rebuild their circuit with 5 randomly selected nodes every 10 minutes. Describe an attack where A can learn the IP address of B. Clearly state your assumptions about the actions of A and his capabilities. Your attack should be realistic, i.e. you should not assume things like the ability for A to break strong crypto.

It is often the case that the operator of hidden service B who is worried about this type of attack will not randomly select the nodes for his circuits, but instead will use the same long-term node as the first hop in his circuits which is called an **entry guard**. This entry guard is a node that B feels can be trusted more than random nodes in the network.

(b) [**1 point**] Does the use of an entry guard change the success of your attack described in part a? If so, how?

Suppose now that you have found a vulnerability in Tor such as the 'The Sniper Attack' `https://www.nrl.navy.mil/itd/chacs/sites/edit-www.nrl.navy.mil.itd.chacs/files/pdfs/13-1231-3743.pdf` which allows you to perform an efficient DDOS attack on Tor relay nodes, effectively taking them off-line.

(c) [**3 points**] Describe an attack against Tor using DDOS attacks that would allow A to find the IP address of B. Suppose that performing DDOS against a node has a non-trivial cost, say \$1,000 and takes 1 day to perform, and hosting your own node has a cost of say \$100. Roughly, how expensive would your attack be on a Tor network consisting of 10,000 nodes if it was desired that the attack could be performed in 1 week? What is the relationship between the cost of performing the attack and the time to perform the attack? Note: Assume that the adverasry has a way to detect which node in the network is the guard node for B.

3 Anonymous Credential Systems [4 points]

Read pages 7-49 of https://groups.csail.mit.edu/cis/theses/anna-sm.pdf and answer the following questions:

(a) **[1 point]** Describe at a high level what the anonymous credential system is designed to be used for and some of its limitations. Be sure to enumerate all parties involved in the system and and the roles that they play.

(b) **[1 point]** Consider protocol Π for proving equality of discrete logarithms on page 39. Argue in detail that the protocol does not leak any information about the value of x to the verifier V, even if V is allowed to choose c in an adversarial way.

(c) **[1 point]** The paper states that protocol Π can be made non-interactive by using a strong hash function. By non-interactive, the paper means that the prover P can publish some information and go off-line, and then the verifier V can download that information and be convinced that P knows x such that $h = g^x, \tilde{h} = \tilde{g}^x$ without any further interactions. Explicitly write a non-interactive version of protocol Π and argue that V should be convinced that P knows x and that V does not learn anything about x.

(d) **[1 point]** The key that makes the pseudonym system work is that a user's nym with an organization $N = (a, b)$ utilizes the users master secret key x corresponding to their master public key g^x. How are these things related and how are they used when creating and transferring credentials? Why is this relation important for the security of the system?

4 (Programming) Elliptic Curve Cryptography [6 points]

In this problem you are going to be implementing ECDH (Elliptic Curve Diffie Hellman) and ECDSA (Elliptic Curve Digitial Signature Algorithm) for `secp256k1` Elliptic Curve standard.

A supplementary file `ecc_template.py` has been provided which provides function prototypes for the following operations:

i. **Point Addition:**
Take in two points on the standard elliptic curve and return their sum which is also a point on the curve, $P_1 + P_2 = P_3$.

ii. **Point Negation:**
Take in a single point on the standard elliptic curve and return the negation of it, $-P_1 = P_2$.

iii. **Scalar Multiplication:**
Take in a single point on the standard elliptic curve group, and an integer in $\mathbb{Z}_n$ were n is the order of the curve subgroup and return the product $n * P_1 = P_2$.

iv. **Sign Message:**
Take in a message m and a private key sk and return a valid signature for m under sk using ECDSA.

v. **Gen Keypair:**
Generate a random public key pk and secret key sk in the elliptic curve setting.

vi. **Generate Forgery:**
Take in two messages and their corresponding signatures that were generated with the same value for k and output a valid signature to a third message using ECDSA.

After implementing the functions that are described, the script will interact with the functions and test them by simulating a key exchange between two parties using ECDH. The shared key will then be used for passing a secret message using a symmetric key encryption scheme, namely AES.

Lastly, the script will interact with a server and download two random messages that are signed with the same value of k. A forgery for a third message will be generated and uploaded to the server where it will be checked for correctness.

To test your code, you can use the following values:
P1.x : 63395642421589016740518975608504846303065672135176650115036476193363423546538
P1.y : 29236048674093813394523910922582374630829081423043497254162533033164154049666

P2.x : 77392770812506936202877798844009338869624245327085260572871517211271361583330
P2.y : 90261830859533359318610421233633301520021539114853433274871094992247021776 27

(P1+P2).x : 68371932353456608033485239330452622297101222794078601004197016497389550492992
(P1+P2).y : 65955754001338186103800512907259643334116137798062372375547840395888950468931

(P1*1234).x : 81574911045610021626036912602689970416286038339612559273196319574968698887196
(P1*1234).y : 52263545852653473929036915641879203268840159261994938359379069356280950009005

Alternatively you can select any x coordinate and calculate a value for y using algebra such that the point lies on the curve. Youc can then add the point to its self, and with high probability, if your solution is incorrect it will not lie on the curve.

You will be graded on the following:

(a) **[3 points]** Attach your code and all output for implementing all functions except **Generate Forgery**.

(b) **[3 points]** Attach your code and all output for implementating all function including **Generate Forgery** as well as any special output from visitng the server webpage with your browser.