18733: Applied Cryptography S17

Homework 4

due April 3, 2017, 11:59pm EST

1 Probabilistic Full Domain Hash [3 points]

In class, we discussed Full Domain Hash (FDH) signing scheme developed by Bellare and Rogaway. We discussed that uf-cma security of FDH relied on the security of invertibility of RSA. Specifically, given a uf-cma attacker A making q_S signing queries and q_H queries to the Random Oracle hash H, an RSA inverter I (against one-wayness of RSA) can be construct such that:

$$Adv^{uf-cma}_{FDH}(A) \leq (q_S + q_H + 1) \cdot Adv^{owf}_{RSA}(I)$$

Please see 'Section 3: The Full-Domain-Hash Scheme' of the following paper from Bellare and Rogaway, and try to understand how the exact reduction steps look like. You may also refer to the lecture slides for the proof. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin

Now, consider the following probabilistic variant signature scheme $S = (G, S, V)$ of FDH. This scheme operates on message space M, and uses a hash function $H : M \times R \to Z_n$:

$$\begin{aligned} G() &:= \{(n, d, e) \leftarrow^R RSAGen, \quad pk := (n, e), \quad sk := (n, d), \quad \text{output } (pk, sk)\}; \\ S(sk, m) &:= \left\{r \leftarrow^R R, \quad y \leftarrow H(m, r), \quad \sigma \leftarrow y^d \in Z_n, \quad \text{output } (\sigma, r)\right\} \\ V\,(pk, m, (\sigma, r)) &:= \{y \leftarrow H(m, r), \quad \text{accept if } y = \sigma^e \text{ and reject otherwise}\} \end{aligned}$$

Using similar reduction steps as in the above paper, show that this signature is secure if the RSA is not invertible, the quantity $1/|R|$ is negligible, and H is modeled as a random oracle. Moreover, show that the reduction to inverting RSA is tight.

2 Key Exchange Protocols [5 points]

We want to build a protocol for that allows a client C and a server S to agree on a shared secret key k. We demand a protocol that has the following properties:

i. **Completeness:** At the end of the protocol, an honest client and server agree on the same key.

ii. **Security:** No PPT active adversary can learn the shared secret key with non negligible probability.

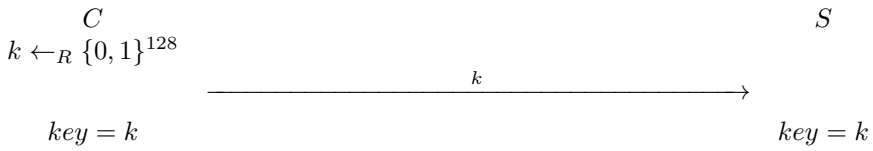
iii. **Privacy:** No malicious active adversary can learn the identity of the client other than their IP address.

iv. **Perfect Forward Secrecy:** An adversary that in the future learns a secret key of some party cannot break the secrecy of saved past conversations.

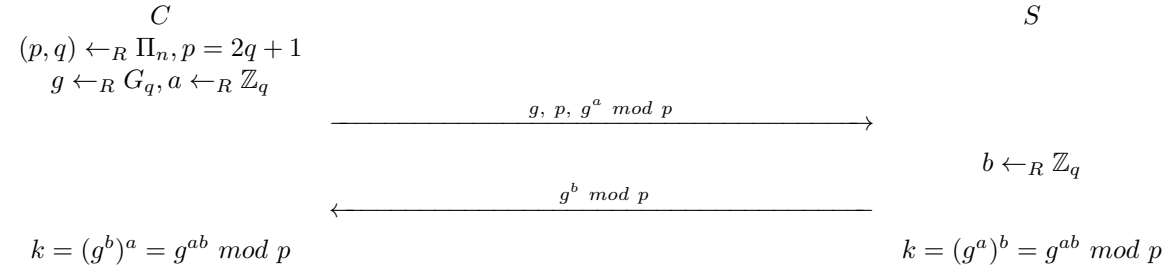
v. **DDOS Resistant:** The server is able to resist DDOS attacks, i.e. running the protocol should not make the server much more prone to DDOS.

For each of the following protocols, state the properties that are satisfied. For each of the properties that are not satisfied, briefly describe an adversary that performs an attack against it. Also, for this problem assume that there is no need to negotiate things like which groups, encryption or hash functions are supported.

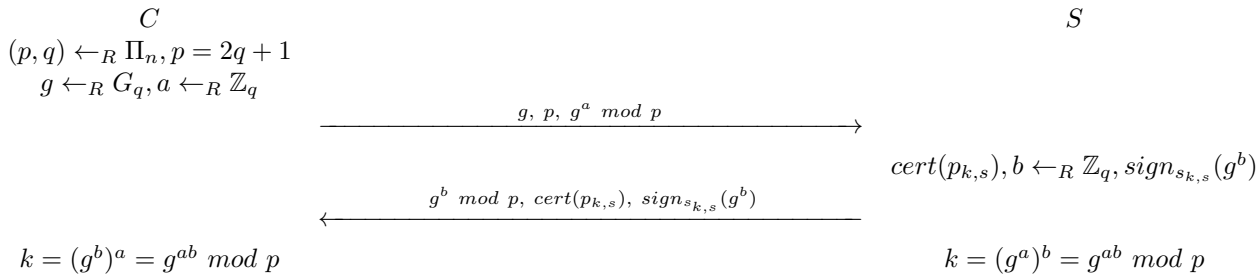
(a) **[0.5 points]**



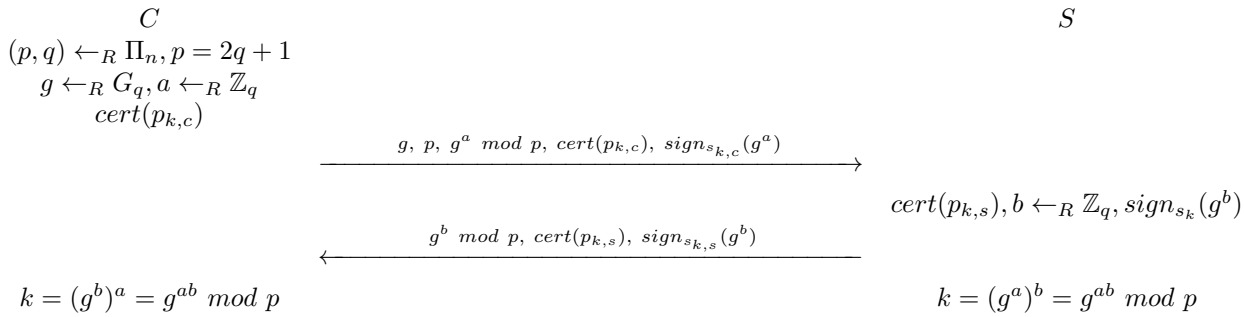
(b) **[0.5 points]**



(c) **[0.5 points]**



(d) **[0.5 points]**



(e) **[1 point]** Is it possible to simultaneously satisfy reciever authentication, client authentication, and client privacy in a protocol with two rounds using the format that we have discussed? Explain.

(f) **[1 point]** Observe the 4 round JFKi protocol in the following paper:
https://www.cs.columbia.edu/ angelos/Papers/jfk-tissec.pdf
Notice that in round 1, the nonce N_I' is sent, and in round 3 the nonce N_I is sent, what is the reason for this? Describe an attack against the protocol if the same nonce is used twice.

(g) **[1 point]** Notice that in the JFKi protocol, in round 2, the receiver sends a commitment of all its protocol decisions, and in round 3 the initiator sends back all of the receivers choices and the commitment. This is done to eliminate the need for the receiver to keep state which mitigates particular types of DDOS attacks. Suppose that the recipient wanted to further mitigate DDOS, describe a general way to modify any key exchange protocol to further mitigate DDOS attacks.

3 SSL / TLS [3 points]

Suppose that there is a company network that connects to the Internet via some gateway as shown in Figure 1. The administrator of the network wants the ability to spy on traffic between the hosts and the server. In order to do this, G will perform a man in the middle attack where it can decrypt TLS traffic between a host (say H1) and S, and then re-encrypt it once it has inspected its contents by creating a fake certificate for the server S.

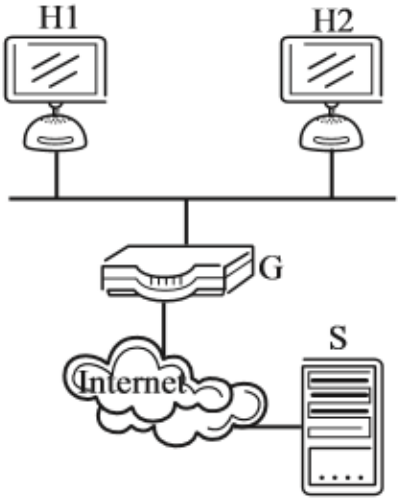


Figure 1: Diagram of a company network consisting of hosts that connect to a server S on the internet through a gateway G.

Suppose H_1 initiates a TLS connection with S:

(a) **[0.5 points]** How would you modify H_1 so that its browser will not complain about the invalid certificate for S.

(b) **[0.5 points]** Can S detect this attack when using server authentication only?

(c) **[0.5 points]** Can S detect this attack when using server and client authentication?

(d) **[0.5 points]** An employee at the company suspects that the company is spying on his traffic, what can the employee do to detect such an attack?

(e) **[1 point]** Suppose that a server that performs the TLS protocol supports the following feature. Instead of re-negotiating the TLS protocol, a client may instead simply provide a unique session identifier based off of a previous session identifier calculated as **session_id = H(old_session_id** $\|$ **server_public_key)**. Will this server be vulnerable to session hijack attacks with respect to an adversary that can eavesdrop on their communication? Explain.

4 SSL / TLS Part 2 [3 points]

Suppose that Gihyuk and Professor Datta have long term key-pairs. Additionally, suppose they communicated the solutions to homework over an SSL protected channel, however it is suspected that after the solutions were communicated, one of the students managed to compromise Gihyuk's long term key. Suppose that the adversary can break strong collision resistance but not one-wayness or weak collision resistance of MD5. For each of the following, describe if the instructors should be worried that the solutions have been leaked.

(a) **[0.5 points]** Anonymous Diffie-Hellman was used as key exchange, 128-bit AES and 128-bit MD5 MAC.

(b) **[0.5 points]** Ephemeral Diffie-Hellman was used as key exchange, 128-bit AES and 128-bit MD5 MAC.

(c) **[0.5 points]** Ephemeral Diffie-Hellman was used as key exchange, 40-bit DES and 0-bit MD5 MAC.

(d) **[0.75 points]** Fixed Diffie-Hellman was used as key exchange, 128-bit AES, 128-bit MD5 MAC.

(e) **[0.75 points]** Fixed Diffie-Hellman was used as key exchange, 40-bit DES, 128-bit MD5 MAC.

5 (Programming) Discrete Log Meet-in-the-Middle Attack [6 points]

The previous assignment gave some practice with the discrete log assumption and showed that an oracle which implemented the $half$ function can be used to efficiently compute discrete logs. In this assignment we are going to look at reducing the complexity of computing discrete logs using a meet in the middle attack.

Let g be some element in $\mathbb{Z}_p^*$, and let $y \in \mathbb{Z}_p^* = g^x$ for some secret x. Further, suppose it is known that x lies in some range, say $1 \leq x \leq 2^{40}$. The goal of this problem is to recover x.

An obvious solution would be to formulate guesses for x, say $x\prime$ and check if $g^{x\prime} = y$. The runtime of this algorithm is $O(2^n)$ where n is the number of bits in the range of possible values for x.

A better solution is to perform a meet in the middle attack which has a runtime of $O(\sqrt{2^n})$ which works as follows. Let $A = 2^{20}$, the secret value x can be expressed as $x = x_0 A + x_1$ where $x_0, x_1 \in [0, A-1]$. Using this expression for x we can re-write $y = g^x = g^{x_0 A + x_1} = (g^A)^{x_0} g^{x_1}$. We can always divide, so we get $\frac{y}{g^{x_1}} = (g^A)^{x_0}$.

The key insight here is that $x_0, x_1 \in [0, A-1]$, so enumerating each side of the equation takes only $O(\sqrt{2^n})$ time. Additionally, since we have this equality relationship, it suffices to find values of x_0, x_1 such that the sides are equal, and we will have found x. To do this, build a hash table and perform 2^{20} multiplications and 2^{20} lookups.

The values for the problem are:

p = 17480715736546509273132356167852223654917350291331787539356496312333028105252468745075491024000992015452563532520952698743383378549938420481917954954410649849158983419586000890687503941868419125253760412312965974672161440234644913519583295579381570913605319820771251183875391960889409590773209931313944629984 3

g = 41899070570517490692126143234857256603477072005476801644745865627893958675820606802876173648371028044404957307185876963051595214534530501331532626624926034521316281025445575243636197258111995884364277423716373007329751928366973332463469104730271236078593527144954324116802080620822212777139186990364810367977

y = 34609266042424639469111447953379872548007680946125118447901412192017942541926039974368791130114250363746053947167838800492643936410021540239167586584541747784025881023472072355868675284705005950466533189974857284371083228488348644524414630393174353452254650096369721977302017953750238841572076375438504612 2

(a) **[4 points]** Find the value of x, and attach your code for doing so.

(b) **[1 point]** Does this attack present a problem for practical instantiations of the discrete log problem?

(c) **[1 point]** Suppose that computation cycles and storage space have asymmetric costs such that it is more expensive to store data than it is to compute it. If you wanted to perform a meet in the middle attack against discrete logs in such a way as to minimize the cost, how would your solution change?