Problem Set #8

CMSC/Math 456
Instructor: Daniel Gottesman

Due on Gradscope, Thursday, Dec. 1, 11:59 PM

Problem #1. Modified Discrete-Log-Based Identification and Signature (30 pts.)

In this problem, we will investigate the need for α in DSA and the associated identification scheme we saw in class.

a) (10 points) Consider a modification of the discrete-log-based identification protocol that we saw in class: The (private key, public key) pair is still $(x, y = g^x \bmod p)$, working over $\mathbb{Z}_p^*$ with prime subgroup of order q and generator g.

1. The initial message from Alice is $I = g^k \bmod p$ for some random $k \in \mathbb{Z}_q^*$.
2. Then Bob returns a challenge, which in this case just random $r \in \mathbb{Z}_q$.
3. Alice returns the response $s = k^{-1}xr \bmod q$.
4. Bob accepts if $s \neq 0$ and $y^{rs^{-1}} = I \bmod p$.

Show that this protocol is *correct*, i.e. that Bob accepts (except with negligible probability) the identity of an honest Alice who has the private key and acts according to the protocol.

b) (10 points) For the same protocol in part a, show that Eve can successfully impersonate Alice with high probability if Eve has seen the transcript (publically transmitted information) from just one previous identification protocol with the true Alice.

c) (10 points) Consider the a modified DSA signature scheme where the private and public keys are as above and the signature of a message m is (r, σ), where $r = g^k \bmod p$ for random $k \in \mathbb{Z}_q^*$ and

$$\sigma = k^{-1}(H(m) + r)x \bmod q. \tag{1}$$

Here $H(m)$ is a hash function, and you may assume that it well approximates a random oracle.

To verify a signature (r, s) with $r, s \neq 0 \bmod q$, determine if

$$r = y^{(H(m)+r)s^{-1}} \bmod p. \tag{2}$$

If so, accept the signature, and if not, reject it.

Do you think this signature protocol is secure? Why or why not?

Problem #2. Old Versions of TLS (30 pts.)

In this problem, you will be exploring security flaws in earlier versions of TLS.

a) (10 points) For the key exchange, TLS 1.1 (which is now deprecated) offered either RSA or Diffie-Hellman, and either RSA or DSS for signatures. For the cipher, the options were RC4 with 128-bit keys, DES in CBC mode, and Triple-DES in CBC mode. (There was also another option for a cipher

called IDEA, which we didn't cover in class and is not included in this question.) The MAC was HMAC and the hash function was either MD5 or SHA-1.

Which of these cryptographic protocols are now considered insecure and which are still considered secure?

Important Note: You can answer this question completely using only material covered in class (although some of the required information was mentioned only in passing). Alternatively, you can find the relevant information on this on the internet, which is acceptable as well. As a reminder, if you do use sources other than the textbook and class notes, you must cite your source. You also must use your own words in your answer.

b) (10 points) TLS 1.2 and earlier allowed compression in conjunction with the encryption. Specifically, the handshake could specify a lossless data compression algorithm, which was then applied to the plaintext in the record layer before encryption. A compression algorithm is an algorithm $C(m)$ which when given a message m attempts to make it shorter by removing redundancy. A lossless data compression algorithm is a compression algorithm which is always invertible, so that for any output it always is possible to find the unique input m which leads to that output. Both the compression algorithm and its inverse should be efficiently computable. Any lossless compression algorithm must reduce the length of some messages and increase the length of others, but the useful algorithms reduce the length of common messages and increase the length of rare ones.

As a concrete example of a lossless compression algorithm, consider a situation where the message consists of strings of 0 to 15 1's separated by single 0's, starting with the first string of 1's and ending with a 0. Let the compression algorithm $C(m)$ be a sequence of 4-bit numbers representing the number of 1's in each string. For instance, if $m = 11111111111011111010111$, then $C(m) = 1010010100010011$. In this case, $|m| = 22$ and $|C(m)| = 16$. If m has a sequence of more than 15 1's in a row (which is not normally allowed for valid messages), $C(m)$ compresses the first 15 1's with 1111 and then continues to compress the remaining message as if it didn't have those 15 1's. For instance, the message $m = 1111111111111111111110111$ compresses to $C(m) = 111101010011$. In this case, the compression function is not invertible, since it does not distinguish between the message with a long string of 1's and the message with a 0 inserted after each 15 1's.

Suppose that the compression algorithm described above is used on messages of the above specified form and then the result is encrypted with a block cipher with 64-bit block sizes (the size used in TLS 1.1) with appropriate padding. Explain how by observing the number of blocks transmitted, Eve can learn information about the original message m and reliably distinguish some pairs of messages m that have the same length.

c) (10 points) Now consider a situation where we use the compression and encryption algorithms as in part b, but Eve has the additional ability to add extra bits of her choice to the beginning of a message m unknown to Eve and the ability to force multiple resends of the same message with potentially different extra bits added to the beginning. (For instance, perhaps this ability is due to Eve managing to get some malware with very limited access onto Alice's device.) Find a strategy whereby Eve can learn the number of 1's at the start of the unknown message m.

Comment: A more sophisticated version of this attack has been used to break TLS, which is why compression is now disallowed in TLS 1.3. The concept of the attack is that authentication information is sent in a cookie which is compressed as part of TLS, and this kind of compression attack can then learn the contents of the cookie.