

Cryptography and Network Security (B20.3157)

Solutions to In-Class Exercises

September 12, 2005

1 First case - shared keys

In the first example, we have a local and remote site that communicate data over the Internet. The local site shares a key with the remote host. Suppose that we will communicate sensitive documents between the two sites. What are the threats, and how do we use cryptography to address these threats?

Let us first consider what we want to protect against. Because we are communicating to a remote site, there is the possibility of an adversary intercepting the communications (think Walsingham in the Mary, Queen of Scots example). Since this question used the Internet as the communications medium, people in the class brought up using IPsec and SSL as examples of solutions to the problem. You will learn about these technologies during the lectures on network and web security. Suffice to say, we need a way to ensure *secrecy*. Additionally, we want to ensure that the information's *fidelity* is retained; that is, we want the remote site to be assured of having the same information that we sent it. We optimally want to demonstrate that we are the ones that sent the message.

What sort of cryptographic constructions do we have at hand that would allow us to maintain these properties? We assume that we share a key k with the remote site (let L be the local site and R be the remote site) and would like to transmit plaintext p . What we need to be able to do in order to enforce secrecy of the communication is to use *encryption*. Recall that for shared (or symmetric) key encryption,

$$c = E(p, k) \tag{1}$$

and

$$p = D(c, k) \tag{2}$$

such that $D(E(p, k), k) = p$, where E and D are the encryption and decryption functions, respectively, of a symmetric-key algorithm (e.g, DES). Now as the local site, we send

$$L : c_L = E(p_L, k) \tag{3}$$

and receive

$$R : c_R = E(p_R, k) \tag{4}$$

which we can decrypt with the decryption function D .

Now that we have preserved secrecy with our messages, are we finished? Unfortunately, our adversary could have tampered with the message even without knowing the details of what is in it. If even a single bit is modified, the message will decrypt to random characters. If we are expecting regular English text, we might be able to immediately tell that this is not right, but if we were expecting a random value, we might never know that modifications happened in transit. We therefore need a method to protect the fidelity of the message, and we accomplish this through a *cryptographic hash*. These functions take in an arbitrarily large amount of input, and output a fixed-size bit string (128 bits for MD5, 160 bits for SHA-1) that is unique for the given input.

Is this secure? We've preserved fidelity, but we're placing a hash of the plaintext at the end of the message in a fixed spot (we need to know where it is so we can separate it from the message, in order to hash for verification). If the adversary knows or guesses the hash algorithm in use, they can precompute a dictionary of words, phrases etc. that they think might be used (e.g., "ATTACK AT DAWN"). If they can match the precomputed string to the hash, they will have a way of matching the plaintext to the ciphertext. (This is described in your textbook as a *known plaintext* attack.) We can use a *salt*, or special number added at the beginning of the message in order to make the numbers hash differently, but both parties would have to know about this value. We already have a value that we share: the symmetric key! So why don't we use the key in order to provide protection against *dictionary attacks* and act as authentication that it was us that sent the message? That's exactly what we do when we use an HMAC. Recall that for some text x ,

$$HMAC(x) = H(x + k) \tag{5}$$

where $H(x)$ is the hashing function and the key is appended to the message prior to the HMAC taking place. Now the flow that we have, which demonstrates secrecy, fidelity, and authenticity, is

$$L : c_L, HMAC(p_L)$$

$$R : c_R, HMAC(p_R)$$

What does the HMAC buy us above and beyond encryption alone when it comes to authenticity? After all, if we have the shared key then by decrypting the ciphertext, we will be able to show that the message was sent by the encryptor, because it will decrypt

to the correct value. However, if we were passing random data, we would not be able to differentiate what the correct decrypted message should be. The HMAC is the one way we are assured that someone with possession on the shared key was the originator of the message.

2 Receipt Generation with Public Keys

We are a bank, and want to design a system that gives account transaction receipts to customers. Assume that the transaction could have taken place through any means (not only online, but over the phone or mail); we are not interested in the actual transaction details, just providing a receipt.

Recall that we have the public key of the client, and the client has our public key. Recall that encryption can be performed in the following manner, assuming the public key k^+ and the private key k^- :

$$c = E(p, k^+) \tag{6}$$

and

$$p = D(c, k^-) \tag{7}$$

Now to send the result to the client while enforcing confidentiality, we send the client the receipt using public key encryption - this ensures that nobody other than the client will be able to decrypt the message, as only the client possesses the private key. To enforce authenticity, we use a *digital signature*. Take the receipt (before encryption), hash it, and sign it with the bank's private key. Now the client, who has our public key, will be able to validate the signature by decrypting it with the key and comparing the hash to the value they computed themselves.

Formally, assume the receipt (along with details such as its ID value and the details of the transaction) are contained in message x . Now we send the client

$$E(x, k_{client}^+), S(x) = E(H(x), k_{bank}^-) \tag{8}$$

where $S(x)$ is the signature as previously defined. To validate the message, the client first decrypts the encrypted receipt message to get the plaintext:

$$p = D(x, k_{client}^-) \tag{9}$$

and hashes the message body:

$$h = H(p) \tag{10}$$

The client then uses the bank's public key to decrypt the signature as follows:

$$v = D(S(x), k_{bank}^+) \tag{11}$$

If the decrypted signature v matches the hash of the message h then the receipt is genuine.