

20: SIGNATURES

So far we have been concerned with how to send messages securely from one person to another. By encrypting the message we try to ensure that, if the message is intercepted, no-one except the intended recipient can decipher it. However, there are many other security concerns. When we receive a message we might want to know who sent it to protect us from forgeries. We might need to know that it has not been altered. We might need to prove that it came from a particular person. For all of these we can adapt the ciphers we have studied to give us the required reassurance.

20.1 Guarantees

Bob sends Alice a message. They may both be concerned about:

Secrecy

No third party can read their message. The ciphers we described in the past two lectures achieve this.

Integrity

No third party has altered the message. For example, if Bob is the customer of the bank owned by Alice, and sends an encrypted instruction to pay £100 to Charles, then Alice needs to be sure that the amount and recipient have not been altered.

The ciphers we have used do not automatically achieve this. For example, suppose that the message to the bank consisted of two numbers (a, m) the first specifying the recipient and the second being the number of pounds to pay them. These are encrypted using the RSA algorithm as $(c_0, c_1) = (a^e, m^e)$. If this message is intercepted, it may be impossible to decipher but it can still be sent again requesting further payments. Even more worryingly, the message could be altered to (c_0, c_1^3) and then sent. This last is called a *homomorphism attack*. It uses knowledge of the form of the cipher.

Authenticity

Alice can be sure that Bob sent the message. The usual signatures at the bottom of letters are intended to ensure that the recipient knows who sent it. We want to develop similar signatures for encrypted messages.

For example, if an enemy intercepts messages that end with a signature identifying the sender, he can swap the signatures to create chaos even if he can not understand the messages. We would like a signature that guarantees that the particular message comes from an identified sender.

Non-repudiation

Alice can prove that Bob sent a particular message. This is the role played by signatures on legal documents. A court can be convinced that a particular person signed the document and so accepted its contents. If electronic communications are to be used in law they require a similar guarantee.

All of these problems are interrelated and we may well require a combination of the guarantees described above.

20.2 Hash Functions

Suppose that we have a function $h : \mathcal{A}^* \rightarrow \mathcal{B}$ from all messages using the alphabet \mathcal{A} into another alphabet \mathcal{B} . There are infinitely many messages but \mathcal{B} is finite, so there are necessarily many messages with the same h -values. Nonetheless, it should be hard to find two messages with the same h -value — a *collision*. If it is simple to compute $h(m)$ for any message m but hard to find a collision, then h is called a *hash function*.

Hash functions are similar to ciphers but they compress the information. They reduce any message to a letter in a fixed alphabet. If the message is changed then it is very likely that the hash value will also change. So we can use the hash value to test the integrity of a message. In this sense it acts as a digest or summary of the message.

Example:

The md5sum of a computer file is a 128 bit binary sequence computed for a file. The value is usually represented as a hexadecimal string of length 32. For instance the md5sum of a text file containing the sentence “The quick brown fox jumped over the lazy dog’s back.” is

ed6761a9a0d26cbe6c7a9666e07bf08d .

Changes to the file will, with very high probability change the md5sum. For example, if we change the capital letter to a small letter we get:

9c9d6b537e1ccc8623ee953bb442d147

which is very different.

The md5sum is used to check the integrity of computer files. If I download a large file over the internet, it may be corrupted either accidentally or maliciously. So I compute its md5sum and compare it with the md5sum published by the provider of the file.

It is often simple to produce hash functions from ciphers. Consider, for example, the RSA cipher with public key (N, e) and private key d . For any letter a I can compute $a^e \pmod{N}$ and then apply a permutation to the binary bits of a^e to give $\sigma(a^e)$. For a message $m = a_1a_2a_3 \dots a_K$ I compute $\sigma_k(a_k^e)$ for a sequence of different permutations σ_k , and then add the results modulo N . Changes to letters have an effect on the hash value that is difficult to predict so it appears to be difficult to find collisions.

In practice hash functions tend to be produced by less algebraic means. A message is first broken up into blocks of fixed length M , say $B_1, B_2, B_3, \dots, B_K$, padding the last block if necessary. Then we choose a compression function (or a sequence of compression functions) $C : \mathcal{A}^M \times \mathcal{A}^M \rightarrow \mathcal{A}^M$. Starting with a fixed initial block J , we compute successively:

$$C_1 = J, C_2 = C(C_1, B_1), C_3 = C(C_2, B_2), \dots, \\ \dots, C_{n+1} = C(C_n, B_n), \dots, C_{K+1} = C(C_K, B_K) .$$

The final value C_{K+1} when we have processed the entire message is the hash value.

We will see that hash functions provide a convenient way to guarantee messages.

20.3 RSA Signatures

Bob wishes to sign a message to show that he sent it. To do this he can use his public key cipher. Suppose that e_B is his encrypting function, which is published for everyone, and d_B his decrypting function, which he keeps private. For simplicity we will suppose that we are dealing with a cipher where the encrypting function and the decrypting function are inverses of each other. So $e_B \circ d_B = I$ as well as $d_B \circ e_B = I$. This is true, for example, for the RSA cipher.

For a simple signature, Bob just takes his name n , computes $s = d_B(n)$ using his private decrypting function, and appends s to his message. Anyone receiving this signature can calculate $e_B(s)$ using Bob's public encrypting function and this is Bob's name $e_B(d_B(n)) = n$. No one else has access to Bob's private decrypting function, so they would find it very hard to sign the message in this way. Note, however, that there is nothing to prevent an enemy from separating Bob's signature from a message he intercepts and attaching it to others spuriously. To avoid this we need a signature that guarantees the integrity of the message.

Bob wishes to send the message m . He adds to this a signature $s = d_B(h(m))$ where h is a publicly known hash function. So he sends (m, s) .

When Alice receives a message (m, s) she can confirm that Bob sent it by using his public key to compute $e_B(s)$. Then she checks that $h(m) = e_B(s)$. Only Bob has the decrypting function d_B so only he could have signed the message with a signature satisfying $e_B(s) = h(m)$. Alice can also be confident that the message has not been altered since any alteration to m would require a corresponding alteration to $d_B(h(m))$.

Bob's signature can also be used to prove to a third party that the message was indeed from Bob. For, if we believe that the cipher we are using is secure, then no one except Bob could have sent a signed message (m, s) with $e_B(s) = h(m)$ without knowing Bob's private decrypting function d_B . Note, however, that Bob can invalidate this by telling others his private key and so allowing them to forge his signature. Furthermore, Alice can not alter the message m that she received without discovering how to make the corresponding change to $d_B(h(m))$.

The message in this example is not encrypted but, if it needs to be, we could use Alice's public key and send $(e_A(m), d_B(e_A(m)))$.

The hash function plays two important roles in this signature scheme. It reduces the length of the signature but it also makes it hard to forge. Suppose, for example, that Bob simply signed the message m with $s = d_B(m)$ and sent $(m, d_B(m))$. It is still possible for Alice to verify that the message is unaltered and came from Bob. However, any enemy can send $(e_B(c), c)$ using Bob's public key and these also appear to be correctly signed by Bob. However, the enemy has no way of knowing what the "message" $e_B(c)$ says or, indeed, if it makes any sense. (Such forged messages are known as *existential forgeries*.) The hash functions defeats this sort of forgery.

We also need to be confident that the message was signed and sent at the correct time, so we should insist that all messages are time stamped. This prevents messages being resent as if they were new.

20.4 The Elgamal Signature Scheme

The Elgamal cipher can also be used to sign messages. Let p be a large prime and γ a primitive root modulo p . Bob chooses $b \in \mathbb{Z}_{p-1}$ and publishes his public key $B = \gamma^b \in \mathbb{Z}_p^\times$. Let h be a hash function that takes values in \mathbb{Z}_{p-1} . To send a message m Bob chooses a random exponent k coprime to $p - 1$ and puts

$$r \equiv \gamma^k \in \mathbb{Z}_p^\times \quad s \equiv \frac{h(m) - br}{k} \pmod{p-1} .$$

(Since $(k, p - 1) = 1$, we know that k is invertible modulo $p - 1$.) Bob's signature is (r, s) so he sends the message (m, r, s) to Alice.

When Alice receives a triple (m, r, s) she checks whether $\gamma^{h(m)} \equiv B^r r^s \pmod{p}$. If the triple is the message (m, r, s) from Bob then we have

$$\gamma^{h(m)} \equiv \gamma^{br+ks} = (\gamma^b)^r (\gamma^k)^s \equiv B^r r^s \pmod{p}$$

so Alice's check succeeds. It is believed that the only way to forge this signature is to solve the discrete logarithm problem.

In the Elgamal signature scheme it is vital that a different random exponent k is chosen for each message. For suppose that two messages m_1, m_2 are signed using the same value for k , giving (m_1, r, s_1) and (m_2, r, s_2) . Then

$$h(m_1) - h(m_2) = k(s_1 - s_2) \pmod{p-1} .$$

Recall that $xs \equiv h \pmod{m}$ has either no solutions for x or else (m, s) solutions modulo m . Hence, there are $(p-1, s_1 - s_2)$ solutions for k modulo $p-1$. Choose the one that gives the correct value for $r \equiv \gamma^k \pmod{p}$.

Then $s_1 \equiv \frac{h(m_1) - br}{k} \pmod{p-1}$ implies that

$$br \equiv h(m_1) - ks_1 \pmod{p-1} .$$

This gives $(p-1, r)$ solutions for b . Choose the one that gives $B \equiv \gamma^b \pmod{p}$. In this way we have found Bob's private key b as well as the exponent k that he is using for signatures.