Message Integrity
CS276 Class Notes for Lecture 8 (second half)

Leslie Ikemoto (lesliei@eecs)

February 18, 2004

1 Introduction

Up until this point, the class has been concerned with *message secrecy*, which attempts to prevent attackers from learning the content of a message. Now, we are going to begin talking about *message integrity*, which is concerned with preventing attackers from changing a message. Message integrity is also known as *authenticity*.

2 The General Scenario

Imagine a person Alice who would like to send a message m to Bob. Alice would like to send her message such that if an attacker Mallet modifies the message in any way, Bob will know that Alice's message has been tampered with.

Let's define S, the sending algorithm, and R the receiving algorithm, such that:

- $S : K \times \{0,1\}^\phi \rightarrow \{0,1\}^\phi$
- $R : K \times \{0,1\}^\phi \rightarrow \{0,1\}^\phi \cup \perp$

where K is the key, ϕ is the message length, and $\perp$ signals that R declares the ciphertext invalid. We also require that $R_k(S_k(m)) = m$ must hold at all times.

Alice encrypts her message m to form a ciphertext $S(m)$. Mallet controls the communication channel between Alice and Bob, and will either pass $S(m)$

through unchanged, or will tamper with it. Bob then receives $S(m)'$, which may or may not be the same as $S(m)$. We would like it to be the case that when Bob runs R_k on $S(m)'$ that R_k will return m if the message was unchanged, and $\perp$ if Mallet tampered with it.

This scenario allows Bob to effectively detect tampering, since if a message decrypts to $\perp$, then it must not be authentic. (Note however that in this scenario Bob cannot prevent message tampering.)

3 Definitions

Given the above scenario, informally we can say that if Bob decrypts a ciphertext and gets a (valid) message m, that Alice must have sent m.

More formally, we define *integrity of plain-text* and *integrity of cipher-text* as follows.

INT-PTXT (integrity of plain text):

$$Adv^{INT-PTXT}A = Pr_k[A^{S_k,R_k} \text{ forges }]$$

A forges if:

- (x_i, y_i) is i^{th} (query, response) to S_k
- (y_i', x_i') is i^{th} (query, response) to R_k

$\exists i$ such that $x_i' \neq \perp$ and $(\forall j\ x_j \neq x_i')$

In other words, an attacker A is given S_k and R_k, oracles for the sending and receiving sides, which allow him to mount chosen plain-text attacks. An attack succeeds when Bob gets a message m that Alice did not send. (In other words, the ciphertext was actually a forgery, but Bob's decryption module did not return $\perp$.)

In addition, we define integrity of cipher-text as follows.

INT-CTXT (integrity of cipher-text):

$$Adv^{INT-CTXT}A = Pr_k[A^{S_k,R_k} \text{ forges }]$$

A forges if:

- (x_i, y_i) is i^{th} (query, response) to S_k
- (y'_i, x'_i) is i^{th} (query, response) to R_k

$\exists i$ such that $x'_i \neq \perp$ and $(\forall j \ y_j \neq y'_i)$

Intuitively, a scheme has INT-PTXT if it is difficult to make a valid ciphertext for a new plaintext, and a scheme has INT-CTXT if it is difficult to make a new, valid ciphertext. INT-CTXT implies INT-PTXT, but INT-PTXT does not imply INT-CTXT. Informally, this is because under INT-CTXT, an attacker cannot tamper with the cipher-text, and thus cannot tamper with the plain-text. However, under INT-PTXT, an attacker cannot tamper with the plain-text, but that does not preclude tampering with the ciphertext.

4 Encryption does not provide integrity

How do we achieve message integrity? One idea we might try is to use an encryption scheme for data authenticity. In other words, we could make S_k an encryption algorithm and R_k a decryption algorithm. The argument for this scheme is that an attacker does not know the key, so it would be hard for the attacker to impersonate the sender.

However, generally in such a scheme, all ciphertexts c of the proper length decode to a valid message m successfully. m may or may not appear as a garbage message to the receiver Bob, but since the forged ciphertext did not decode to $\perp$, we consider this a successful attack.

Let's now suppose we add redundancy to the message by, for example, appending 32-bits of zeros. However, an attacker could easily break this scheme as well by simply changing the first few bits of the cipher-text. As long as the last 32-bits remained unchanged, the decryption algorithm would return a valid message m.

We could continue to try to make this scheme more robust by adding, for example, a check-sum to the message. But an attacker could robustly break this scheme as well.

Thus, we begin to see that encryption is not a good technique for message integrity.