Secret Codes

Example 1
CSE 791: E-1

Title: Logic, SML, & HOL90
SML files: SecretCode.sml
Objective: Basic Cryptographic Notions

1 Cryptography

Cryptography serves privacy needs by encryption. It serves source authentication and non-repudiation needs through the use of secrets. It serves integrity through message integrity codes (MIC) for secret key cryptography or digital signatures for public key cryptography.

1.1 Types of Cryptographic Functions

There are three kinds of cryptographic functions: secret key functions, public key functions and hash functions. Public key cryptography uses two keys. Secret key cryptography uses one key. Hash functions uses no keys.

1.1.1 Secret Key Cryptography

In *Secret key* or *symmetric* cryptography, the same key $s$ is used for both encryption and decryption, as shown in Figure 1. Ciphertext is obtained by applying the encryption function to both plaintext and the secret key. To retrieve the original plaintext, decryption function is applied to the ciphertext and the same secret key. A message $m$ encrypted with secret key $s$ is denoted as $[m]_s$.

Ideally secret key cryptography has following property: a message encrypted with secret key $k$ can only be retrieved (decrypted) with the same secret key. When an initial vector (IV) is utilized in the cryptographic algorithm, it must be the same for both encryption and decryption. This can be formalized as:

\[
\forall msg \; key \; IV.
\]

\[
( decryptS ( encryptS \; msg \; key \; IV) \; key \; IV = msg ) \land
( \forall msg2 \; key2. \; ( decryptS \; msg2 \; key2 \; IV =
\quad decryptS \; msg2 \; key2 \; IV ) = key = key2)
\]  \tag{1}

The secret key scheme can be used to generate a fixed-length cryptographic checksum associated with a message, as shown in Figure 2; this message integrity code (MIC) can be used to check the integrity of the message sent along with it (see section 1.4).
1.1.2 Public Key Cryptography

In public key or asymmetric cryptography, each individual has a pair of keys: a private key \( d \) only known to the owner, and a corresponding public key \( e \) that is accessible by the world. The public key is used for encryption and the private key is used for decryption. This is shown in Figure 3. A message \( m \) encrypted using public key \( e \) is denoted as \( \{ m \}_e \).

Public key cryptography has following property: a message encrypted with public key \( e_k \) can only be retrieved (decrypted) with an unique private key \( d_k \); on the other hand, a message encrypted with private key \( d_k \) can only be retrieved (decrypted) with the unique public key \( e_k \). This can be formalized as:

\[
\forall msg \ eKEY \ dKEY.
\]
\[
\begin{align*}
((\text{decryptP} (\text{encryptP} \ msg \ eKEY) \ dKEY = msg) = \text{true}) \land \\
((\text{encryptP} (\text{decryptP} \ msg \ dKEY) \ eKEY = msg)) \land \\
((\forall dk. (\text{decryptP} (\text{encryptP} \ msg \ eKEY) \ dk = msg) \lor dk = dKEY) \land \\
(\forall ek. (\text{encryptP} (\text{decryptP} \ msg \ dKEY) \ ek = msg) \lor ek = eKEY))
\end{align*}
\]

Public key cryptography can be used to generate signature on any message. The signature can be verified by anyone who knows the public key of the signer, and can only be generated by the one who knows the corresponding private key. This is shown in Figure 4. These two properties can be formalized as follows:
\( \forall m_1 m_2 \text{dkey}_1 \text{dkey}_2. (\text{sign } m_1 \text{dkey}_1 = \text{sign } m_2 \text{dkey}_2) \)
\( \supset (m_1 = m_2) \land (\text{dkey}_1 = \text{dkey}_2) \)  \hspace{1cm} (3)

\( \forall \text{msg eKEY dKEY. verify } \text{msg } (\text{sign msg dKEY}) \text{eKEY } \supset 
(\forall m_1 m_2. \text{verify } m_1 m_2 \text{eKEY } = (m_2 = \text{sign } m_1 \text{dKEY})) \)  \hspace{1cm} (4)

The counterpart of MICs for public key cryptography are digital signatures as shown in Figure 5. They are used to check integrity.
1.1.3 Hash Functions

Hash functions are message digests or one-way transformations. A cryptographic hash function is a mathematical transformation that takes a message of arbitrary length and computes from it a fixed length number.

Hash functions have the following properties:

- If $h(m_0)$ denotes the hash of the message $m_0$, there is no substantially easier way to find an $m$ whose hash is $h(m_0)$ without going through all values of $m$ to search for $h(m_0)$.
- It is computationally infeasible to find two values of $m$ which hash to the same value.

Essentially, hash functions behave like one-to-one functions, i.e.,

$$\forall m \neq m'. h(m) = h(m') \Rightarrow m = m'$$

(5)

1.2 Privacy

Privacy is obtained through encryption. If Emily wants to send Benjamin a mail that only Benjamin can read, she will choose a random secret key $S$ to be used only for encrypting that one message $m$. She encrypts the message with $S$ to get $[m]_S$, encrypts $S$ with Benjamin’s public key $c_B$ to get $\{S\}_{c_B}$ (if public key cryptography is used) or with the secret key she shares with Benjamin $K_{EB}$ to get $[S]_{K_{EB}}$ (if secret key cryptography is used), and transmits both to Benjamin.

Privacy in PEM is gotten by using any of the following cryptographic functions: DES-CBC for secret key encryption of messages; DES-EDE for secret key encryption of Data Encryption Keys (DEKs); DES-ECB for secret key encryption of DEKs; RSA for public key encryption of DEKs and signatures. Summaries of each of the encryption algorithms mentioned here are found in [1].

1.3 Authentication

Authentication verifies the identity of the communicating party. Encryption is used to prove the knowledge of secrets, hence to verify identities. The means for doing so are variations on a challenge/response protocol. A challenge is issued by the party wishing to verify the identity of the other principal. The principal, whose identity is being checked, issues a response based on the use of a secret key or public key cryptography.
In secret key cryptography, if Emily wants to verify the identity of Benjamin, she issues a challenge, a random picked number \( r \), and sends it to Benjamin. Benjamin encrypts the \( r \) with the the secret key \( K_{EB} \) he shares with Emily and sends it back to Emily. Emily decrypts the response with \( K_{EB} \) and checks to see if she got back \( r \) (see Figure 6).

![Figure 6: Secret Key Authentication](image)

If public key cryptography is used, Emily chooses a random number \( r \), encrypts it with Benjamin’s public key \( e_B \) and sends the result to Benjamin. Benjamin proves he knows his private key \( d_B \) by decrypting the message and sending \( r \) back to Emily (see Figure 7).

![Figure 7: Public Key Authentication](image)

1.4 Integrity

Integrity of a message is maintained by using either a MIC (in secret key cryptography) or a digital signature (in public key cryptography) shown in Figure 2 and Figure 5.

For secret key cryptography, a MIC is computed by using a secret key with a known checksum algorithm. It is included as part of the header sent along with the message to the recipients. The recipients compute the MIC for the message they receive and compare it to the MIC received in the header. If the MICs match, then the message is genuine (see Figure 8).

For public key cryptography, integrity is protected by digital signatures. If Emily wants to send Benjamin a message which is integrity protected, she generates the digital signature of the message using her private key, and send it along with the message to Benjamin. When Benjamin receives the message with its digital signature, he verifies the digital signature with Emily’s public key (see Figure 9).
Hash functions are used with public keys for integrity protection (see Figure 9). Signing a message digest is much quicker than signing a message itself. When the signature of the message digest is sent with the message to recipients, the recipients generate the message digest from the message, and verify the signature of the digest to check the integrity of the message.

1.5 Non-repudiation

Non-repudiation is the ability of the recipient to prove to a third party that the sender really did send the message. It comes automatically with public key cryptography as only the person who knows the private key can generate the signature. Comparing the message digest with the signature decrypted using originator’s public key is all that is required.

2 Caesar Cipher

The Caesar Cipher maps characters of plain text into cipher text by translating each character by a constant amount as specified by a key. For example, if the key were 2, then A would become C.

We can implement a modern equivalent of the Caesar Cipher using the ASCII character set. Each ASCII character has a numerical code associated with it. For example, A 🜢 65. There are 128 ASCII characters. The following program shows how it is implemented.
Now is the time for all people \nto come to the defense of the nation. 

where \n is a “new line” is encrypted as shown below.

The Caesar code implemented on
val m1 = "Now is the time for all people \nto come to the defense of the nation.\n"
val m1 = "Now is the time for all people \nto come to the defense of the nation.\n"
print m1;
val k1 = 56;
val k1 = 56 : int
val em1 = Caesar k1 m1;
val em1 = "\F'/X!+X, \029X,!%\029X\030'\&X\~Y$X\029'(\$\029X\B,’\029%\029X,\’X, \029 X\028\029\030\029&+\029X'\030X, \029X&\~Y,!+'&fB#"
val dm1 = invertCaesar k1 em1;
val dm1 = "Now is the time for all people \nto come to the defense of the nation.\n"

References