Digital Signature

A digital signature is a construct that authenticates both the origin and contents of a message in a manner that is provable to a disinterested third party.

Provides a service of nonrepudiation

Case Study 1: Is \{m\} k a Signature?

- Alice sends Bob \( m \) k
  - k is the shared secret key between Alice and Bob
  - Bob deciphers \( m \) k and compares with m
    - Thus verify that the message is from Alice
    - And the contents are not modified
  - But Bob cannot prove to a third party that \( m \) k is not created by Bob himself!
  - So \( m \) k is not a signature of m

Case Study 2: Is \( m \) \( d_{Alice} \) a Signature?

- Alice sends Bob \( m \) \( d_{Alice} \)
  - \( d_{Alice} \) is the private key of Alice
  - Bob deciphers \( m \) \( d_{Alice} \) and compares with m
    - Thus verify that the message is from Alice
    - And the contents are not modified
  - A judge can verify \( m \) \( d_{Alice} \) is indeed signed by Alice if:
    \( m = (m \) \( d_{Alice} \) \( s_{Alice} \)

Classical Signature

Let Cathy be a trusted third party
- Alice shared a secret key \( k_{AC} \) with Cathy
- Bob shared a secret key \( k_{BC} \) with Cathy

1. Alice→Bob: \( m \) \( k_{AC} \)
2. Bob→Cathy: \( m \) \( k_{BC} \)
   - Cathy decrypts with \( k_{AC} \) and re-encrypts with \( k_{BC} \)
3. Cathy→Bob: \( m \) \( k_{BC} \)
   - Bob then gets m

Classical Signature Verification

- Verification question: is m created by Alice?

- Verification method:
  - Judge Takes the disputed messages \( m \) \( k_{AC} \) & \( m \) \( k_{BC} \)
  - Ask Cathy to decrypt \( m \) \( k_{AC} \) using \( k_{AC} \) and \( m \) \( k_{BC} \) using \( k_{BC} \)
  - And compare \( (m \) \( k_{AC} \) \( k_{AC} = ((m \) \( k_{BC} \) \( k_{BC} \) \)?
Public Key Signature

- Instead of using \((m) d_{\text{Alice}}\), Alice actually signs the message as \((h(m)) d_{\text{Alice}}\), where \(h\) is a cryptographic hash function.
- And sends Bob \(m (h(m)) d_{\text{Alice}}\).
- Q: how does Bob verify the signature?

Cipher Techniques

Intro

- Cipher techniques must be used wisely
  - Very sensitive to the environment
- A mathematically strong cryptosystem is vulnerable when used incorrectly

Examples

- Precomputing the possible messages
- Misordered Blocks
- Statistical Regularities

Precomputing Possible Messages

- Simon’s Attack: “Forward search” technique
- Alice will send Bob one of two messages: BUY or SELL, enciphered with \(e_{\text{Bob}}\):
  - Eve does not know which one, but
  - Eve knows it’s one of the two
- Eve precomputes the (“BUY”) \(s_{\text{Alice}}\) and (“SELL”) \(s_{\text{Bob}}\)
- When Alice sends Bob a message, Eve intercept it and compare with the precomputed ciphertext
  - Then Eve knows what’s the plaintext!
- Problem: the set of plaintext if small

Misordered Blocks

- Denning: part of ciphertext can be deleted, replayed, or reordered

  - LIVE \[\rightarrow 44 \ 57 \ 21 \ 16 \rightarrow 44 \ 57 \ 16 \rightarrow \text{EVE} \]
  - LIVE \[\rightarrow 44 \ 57 \ 21 \ 16 \rightarrow 44 \ 57 \ 16 \rightarrow \text{LIE} \]
- Each part can be signed
  - But if signed separately, will it work?
- Problem: each part is encrypted independently
Statistical Regularities

- When each part is enciphered separately, the same plaintext will produce the same ciphertext
  - Thus regularity arises
  - Making cryptanalysis easy

- This type of encryption is called **code book mode**

So . . .

- How to use cipher techniques?

Block Cipher

- **E**: an encipherment algorithm
- **E_k(b)**: encipherment of msg **b** with key **k**
- Message **m** = **b_1** **b_2**..., where each **b_i** is of fixed length

- **Block cipher**: **E_k(m) = E_k(b_1) E_k(b_2) . . .**
- Q: is DES a block cipher?

Stream Cipher

- **E**: an encipherment algorithm
- **E_k(b)**: encipherment of msg **b** with key **k**
- Message **m** = **b_1** **b_2**..., Key **k** = **k_1** **k_2**..., where each **b_i** is of fixed length

- **Stream cipher**: **E_k(m) = E_k(b_1) E_k(b_2) . . .**
- Q: is Vigenere a stream cipher?

Periodic Stream Cipher

- **Periodic (stream) cipher**: the key stream **k** of a stream cipher repeats itself

- Q: is Vigenere cipher a periodic cipher?
- Q: is one-time pad a periodic cipher?

Synchronous Stream Ciphers

- Generates bits of a key from a particular source
  - Not from the message itself
  - Hopefully the newly generated key is random and long
- Several techniques
  - LFSR (Linear feedback shift register)
  - NLFSR (Nonlinear feedback shift register)
  - Output feedback mode
  - Counter method
LFSR (linear feedback shift register)

- An n-bit register \( r = r_{n-1} \ldots r_0 \) (a variable)
- An n-bit tap sequence \( t = t_{n-1} \ldots t_0 \) (a constant)
- Use \( r_0 \) as current key bit
- Right shift \( r \), and \( r_{n-1} = (r_{n-1} \cdot t_{n-1}) \oplus \ldots \oplus (r_0 \cdot t_0) \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>current reg</th>
<th>key</th>
<th>new ( r_{n-1} ) bit</th>
<th>new reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>000</td>
</tr>
<tr>
<td>000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1000</td>
</tr>
</tbody>
</table>

The key stream can have a period of \( 2^n - 1 \) (maximal value)

NLFSR (nonlinear feedback shift register)

- New bit is a function of current register bits
  - No tap sequence used

\[
\begin{array}{ccc}
\text{current reg} & \text{key} & \text{new } r_{n-1} \text{ bit} & \text{new reg} \\
0010 & 0 & f(0,0,1,0)=0 & 0001 \\
0001 & 1 & f(0,0,0,1)=0 & 0000 \\
0000 & & & 0000 \\
\end{array}
\]

\[ f = r_3 \text{or } (r_2 \text{ and } r_0) \]

Output Feedback Mode

\( m \): the message to encrypt
\( E \): encipherment function
\( k \): cryptography key to generate
\( r \): a register (never shifted)
- \( r = E_k( r ) \)
- \( k_i = r_0 \) (\( r \)'s rightmost bit)
- \( c_i = m_i \oplus k_i \)

Counter Method

\( m \): the message to encrypt
\( E \): encipherment function
\( k \): cryptography key to generate
\( i_0 \): initial value of a counter
- \( k_i = (i+i_0)'s \) rightmost bit \( (\text{for } i=0, 1, 2, \ldots) \)
- \( c_i = m_i \oplus k_i \)

Self-Synchronous Stream Ciphers

- Generate a key from the message itself
  - Could be from plaintext, could be from ciphertext
  - Also called autokey cipher

<table>
<thead>
<tr>
<th>Key</th>
<th>Plaintext</th>
<th>CipherText</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{THEBOYHASTHEBA} )</td>
<td>( \text{THEBOYHASTHEEDAG} )</td>
<td>( \text{QALPPHHNLIPLCT} )</td>
</tr>
<tr>
<td>( \text{QXBCQOVVNGNRTT} )</td>
<td>( \text{THEBOYHASTHECAT} )</td>
<td>( \text{QXBCQOVVNGR8TM} )</td>
</tr>
</tbody>
</table>

(cont’d)

- If using plaintext, key selection is an issue
  - Key will display same statistical regularities as it’s derived from plaintext
- If using ciphertext, weak
  - A character in ciphertext \( = f(X, \text{a previous character in ciphertext}) \)
Cipher Feedback Mode

- \( m \): the message to encrypt
- \( E \): encipherment function
- \( k \): cryptography key to generate
- \( r \): a register
  - \( x = E_k(r) \)
  - \( c_i = m_i \oplus x_0 \) (is \( x \)'s rightmost bit)
  - \( r = x_n \oplus r_{n-1} \oplus \ldots \oplus r_1 \)

Block Cipher

- Multiple bits each time
  - Faster than stream cipher in software implementations
- But an identical plaintext block will produce an identical ciphertext block
  - If using the same key

Strengthening Block Cipher

1. Insert extra bits into a block, often related to block position
   - Sequence number of a block
   - Bits from preceding ciphertext block
2. Cipher block chaining (CBC)
   - \( c_0 = E_k(m_0 \oplus I) \)
   - \( c_i = E_k(m_i \oplus c_{i-1}) \) for \( i > 0 \)

Three Common Modes of DES

- CBC: Cipher Block Chaining
- EDE: Encrypt-Decrypt-Encrypt
- Triple DES: DES-DES-DES

(cont’d)

3. Encrypt-Decrypt-Encrypt (EDE)
   \( c = E_k(D_k(E_k(m))) \)
4. Triple Encryption Mode
   \( c = E_k(E_k(E_k(m))) \)
EDE

- Two 64-bit keys: $k$ and $k'$

$$c = DES_k(DES_{k'}^{-1}(DES_m(m)))$$

Triple DES

- Three 64-bit keys: $k$, $k'$, and $k''$

$$c = DES_k(DES_{k'}(DES_{k''}(m)))$$