Web Services Security

Jan Jusko

ATG, FEE, CTU

November 1, 2012

1. Motivation
2. Cryptography
3. Web Services Security

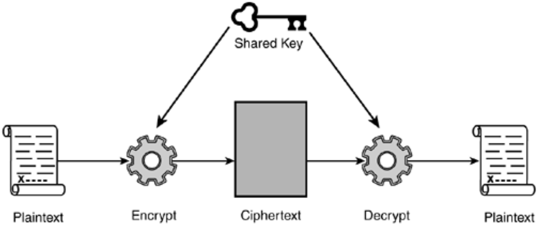
- **Integrity** — messages are not duplicated, modified, reordered, replayed, etc.
- **Confidentiality** — protects communication and data from passive attacks as eavesdropping, traffic analysis, and disclosure
- **Authentication** — allows agents to prove their identity each other, i.e. to verify whether the counterpart is what it claims to be
- **Non-repudiation** — someone who received confidential information cannot deny that she received it

Cryptography

- Address the needs to communicate in secure, private, and reliable ways
- translate a message M into its encrypted form, the cipher-text H, and then decrypts it back into its original form
- symmetric/asymmetric cryptography
- algorithms are well known

Symmetric Cryptography

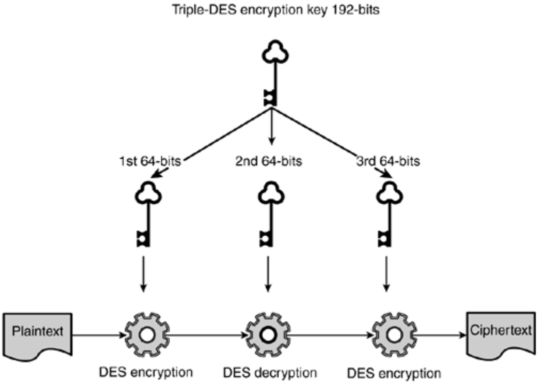
- uses the same key to both encrypt and decrypt the message
- the key needs to be exchanged out of band



Symmetric Cryptography

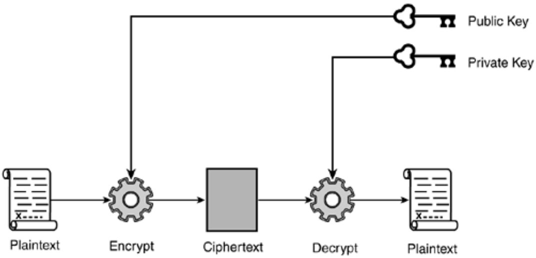
- DES is an example of symmetric encryption algorithm (uses 64-bit key)
- new version of DES is 3DES , which uses 192-bit key
- the newest algorithm is AES, also known as *Rijndael*, uses keys of length up to 256 bits
- all of them have hardware implementations

3DES schematics

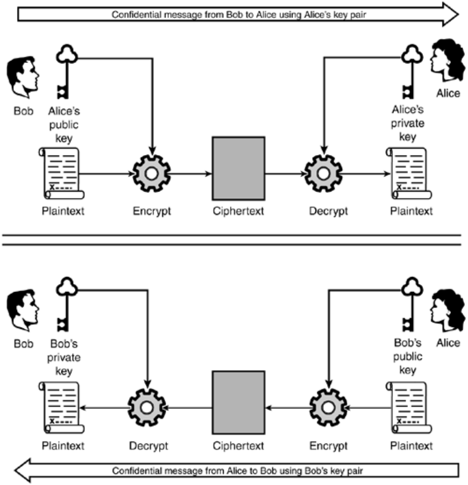


Asymmetric Cryptography

- uses a pair of public/private key
- private key is bigger
- message encrypted by public key can be decrypted only by private key and vice-versa
- RSA, DSA



Conversation Using Asymmetric Cryptography



Symmetric vs. Asymmetric Cryptography

- symmetric is much faster
  - in hardware implementations, RSA is 1000 times slower than DES
- symmetric crypto can be used on data of arbitrary size
- asymmetric crypto has a size limitation on the data to encrypt
- key exchange is much easier for asymmetric crypto
- asymmetric crypto can be used to exchange the key for symmetric crypto

Hybrid Cryptography Scheme

- both parties exchange their public keys
- exchange of the encrypted *secret* keys
- in further communication, symmetric encryption is used

Hash Function

- A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h (that is, h = H(m)).
- The basic requirements for a cryptographic hash function are:
  - the input can be of any length
  - the output has a fixed length
  - H(x) is relatively easy to compute for any given x
  - H(x) is one-way
  - H(x) is collision-free

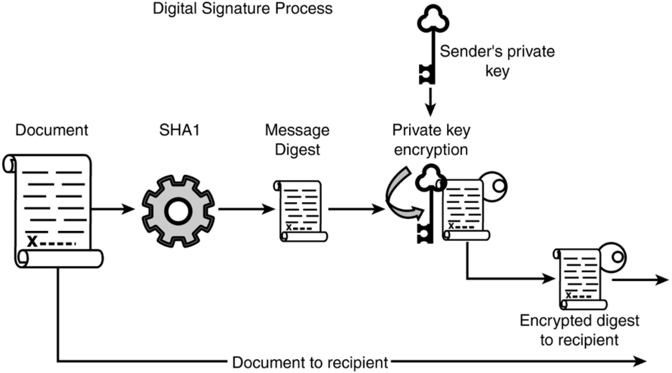
Hash Function

- A hash function H is one-way if it is hard to invert, where "hard to invert" means that given a hash value h, it is computationally infeasible to find some input x such that H(x) = h
- Given a message x, it is computationally infeasible to find a message y not equal to x such that H(x) = H(y) then H is said to be a weakly collision-free hash function
- A strongly collision-free hash function H is collision-free for any x, y

Signing

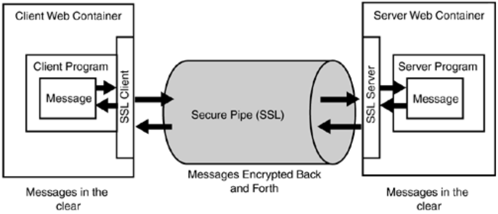
- uses hash function in conjunction with *private key*
- first calculates the digest and encrypts it afterwards

Digital Signature Process



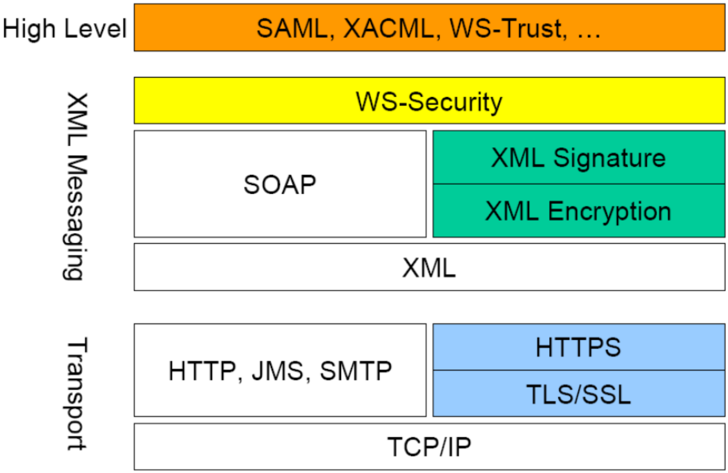
Web Services Security

- point-to-point



  - well established protocols
  - intermediaries can see the plain-text message
- end-to-end
  - SOAP message self-protection

Web Services Security Framework



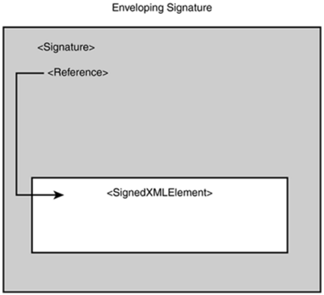
Web Services Security

- **XML Signature (XMLDSIG)** — Message Integrity and Sender/Receiver Identification
- **XML Encryption (XMLENC)** — Message Confidentiality
- **WS-Security (WSS)** — Securing SOAP Messages

XML Signature

- guarantees identity (authentication) & integrity
- used to sign parts of the XML document or the document itself
- a digest is calculated for the part of the message to be signed and this digest is encrypted
- adds `Signature` element to the XML
- can be of three types
    - enveloping
    - enveloped
    - detached

Enveloping XML Signature

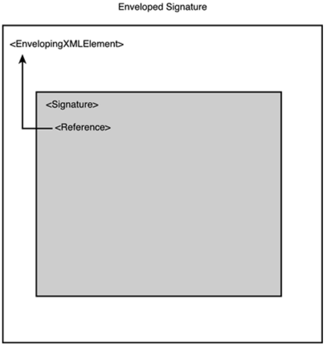


Enveloping XML Signature

Example

```
<Signature>
  <SignedInfo>
    <Reference URI="#111" />
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
  <Object>
     <SignedItem id="111">
        Stuff to be signed
     </SignedItem>
  </Object>
</Signature>
```

Enveloped XML Signature

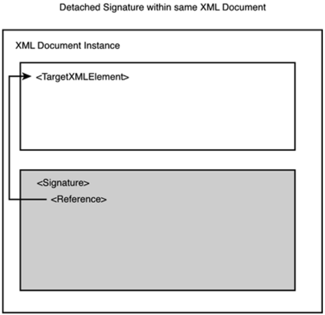


Enveloped XML Signature

Example

```
<PurchaseOrder id=”po1”>
  <SKU>125356</SKU>
  <Quantity>17</Quantity>
  <Signature>
    <SignedInfo>
      <Reference URI=”#po1” />
    </SignedInfo>
    <SignatureValue>...</SignatureValue>
    <KeyInfo>...</KeyInfo>
  </Signature>
</PurchaseOrder>
```

Detached XML Signature



Detached XML Signature

Example

```
<PurchaseOrderDocument>
  <PurchaseOrder id=”po1”>
    <SKU>12366</SKU>
    <Quantity>17</SKU>
  </PurchaseOrder>
  <Signature>
  <SignedInfo>
    <Reference URI=”#po1” />
  </SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```

XML Signature

- `KeyInfo` elements contains information about key that was used to encrypt the digest
- `SignatureValue` is actually a signature of `SignedInfo` element
- one Signature can contain several References
- Reference URI can point to any document with valid URI (i.e. even external)
- digests are calculated on raw data, thus we need canonization (e.g. to get rid of various line endings on Windows/Linux)

XML Encryption

- guarantees confidentiality
- used to encrypt parts of the XML document or the document itself
- unlike transport layer encryption, this is end-to-end protection
- can use both symmetric and asymmetric encryption — in reality, a combination of both is used
- different security mechanisms can be applied to request and response
- can be **only enveloping**

Original Data

Example

```
<Employee>
    <EmployeeID>512-34-4567</EmployeeID>
    <Manager>Fred Jones</Manager>
    <Salary>$50,000</Salary>
</Employee>
```

Encrypted Data

Example

```
<Employee>
    <EmployeeID>
        <EncryptedData>...</EncryptedData>
    </EmployeeID>
    <Manager>Fred Jones</Manager>
    <EncryptedData>...</EncryptedData>
</Employee>
```

XML Encryption

- can encrypt element contents or the element itself
  - `EmployeeID` has encrypted data
  - element `Salary` was encrypted as a whole
- EncryptedData contains base64 encoded value

Using XML Signature and Encryption together

- three strategies of encryption & signing
  - first encrypt, then sign
  - first sign, then encrypt
  - sign, encrypt, sign again
- all approaches have trade-offs

WS-Security

- used to protect SOAP messages
- describes three main mechanisms
  - how to sign SOAP messages to assure integrity
  - how to encrypt SOAP messages to assure confidentiality
  - how to attach security tokens to ascertain the sender's identity
- takes advantage of XML Signature and Encryption
- security information is put into the SOAP header

WS-Security

Example

```
<Security>
  <!-- Security Token -->
  <UsernameToken>...</UsernameToken>
  <!-- XML Signature -->
  <Signature>
  <Reference URI="#body">
  <Signature>
  <!-- XML Encryption Reference List -->
  <ReferenceList>
    <DataReference URI="#body"/>
  </ReferenceList>
</wsse:Security>
```