A4M36AOS – Architektury orientované na služby

6. Services and Security

Jiří Vokřínek

Agent Technology Center
Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University in Prague

jiri.vokrinek@fel.cvut.cz http://agents.fel.cvut.cz

Services and Security

- Why we need security?
- Threads and risks
- Cryptography
- Web services security

Why we need security?

Services and Security

- SOA aims to large-scale deployment
- Often used in security critical applications, such as e-commerce, e-health, social networks
- Dynamic environment with heterogeneous platforms
- SOA provides high data availability, dynamic service configuration, loose-coupled autonomous services

Services and Security

- SOA aims to large-scale deployment
- Often used in security critical applications, such as e-commerce, e-health, social networks
- Dynamic environment with heterogeneous platforms
- SOA provides high data availability, dynamic service configuration, loose-coupled autonomous services

→ Results in big **security risk**

Services and Security

- Oriented to servers, clients, messages, ...
- **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.

Messages Security

- Covered by cryptography standards
- Symmetric (DES, AES) and asymmetric (RSA) cryptography
- Digital signatures
- Certificates
- “Secured” communication channel
- Reliable protocols, error-detection (hash, etc.)

Client Security

- Service discovery risk
- Phishing risk
- Authenticity and correctness of received messages
- Authentication of service providers

Server Security

- Public services are open to attacks and misuse
- Complex system build using many 3rd party components
- Protection of application server, HTTP server, parsers and encoders, database engine, data protection, …
- Identification of service consumers
- Authentication and authorization to access server resources

Threads and risks

Threads and risks

- **Social attacks** – not so important in SOA, the most dangerous attack (human factor involved)
- **Hardware attacks** – packet spoofing, sniffing ... risk on message level or communication infrastructure
- **Software attacks** – design and implementation problem, complex servers weakness, configuration weakness (availability of unnecessary services, misconfiguration), infrastructure attacks (DNS, TCP, XML; racing, DoD, buffer overflows, etc.)

Example of Attacks

Besides “classical” attacks …

- SQL injection
- Capture-replay attack
- XML external entity attack
- XML denial of service (XDoS)
- Harmful SOAP attachments
- XML Signature dereference attacks

Threads and risks

- Identification and modeling of risk and threads
- Iterative process from design and implementation to deployment and execution
- Goals is to minimize damages, but also balance the loss vs. cost of the security
- Assets – data, access control, reputation
- Security level – integrity, availability, cost of assets and it’s loss or misuse

Cryptography

Cryptography

A short excursion to …

- Cryptography basics
- Private key (symmetric) cryptography
- Public key (asymmetric) cryptography
- Digital signature
- Hash function
- Secure communication on public channels

Cryptography

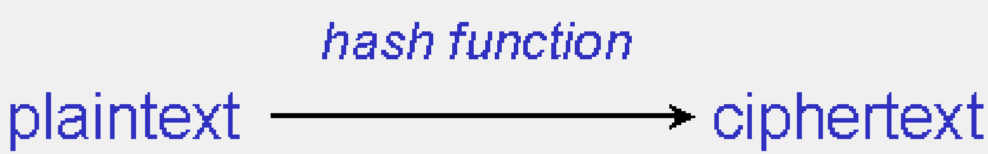
- Address the needs to communicate in secure, private, and reliable ways
- translate a message M into its encrypted form, the ***cipher-text H***, and then to decrypt it back into its original form

$$H = Encr(M) \text{ and } M = Decr(H)$$

Cryptography

A) Secret key (symmetric) cryptography. SKC uses a single key for both encryption and decryption.

B) Public key (asymmetric) cryptography. PKC uses two keys, one for encryption and the other for decryption.



C) Hash function (one-way cryptography). Hash functions have no key since the plaintext is not recoverable from the ciphertext.

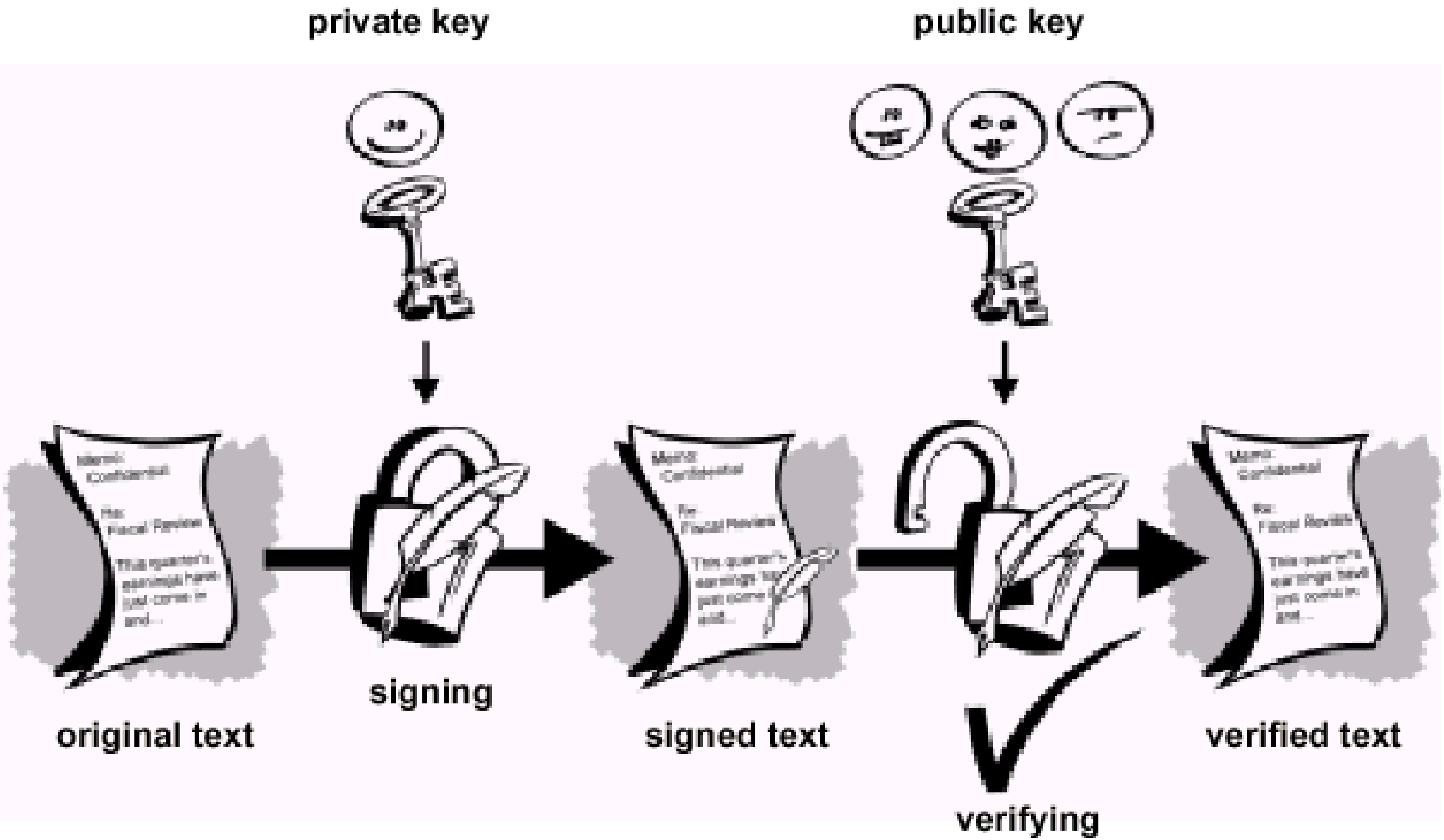
Public key cryptography

- **Encryption function *Encr*** (public)
- **Decryption function *Decr*** (private)
- Duality equation

$$Decr_A(Encr_A(M)) = M \quad \text{and} \quad Encr_A(Decr_A(M)) = M$$

- In both symmetric and asymmetric cryptography – keys distribution is the most critical issue

Digital signature



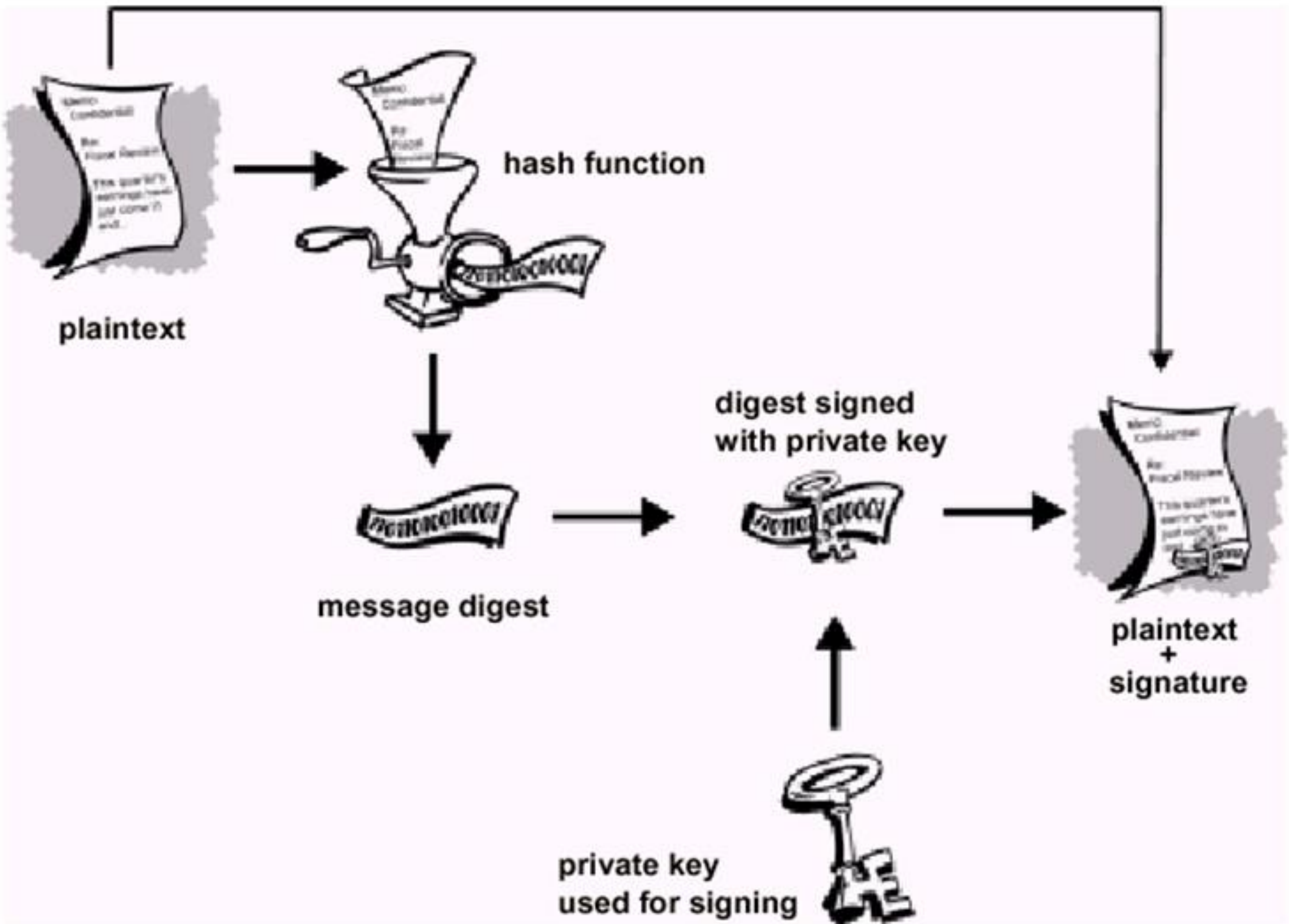
Hash function

- 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.

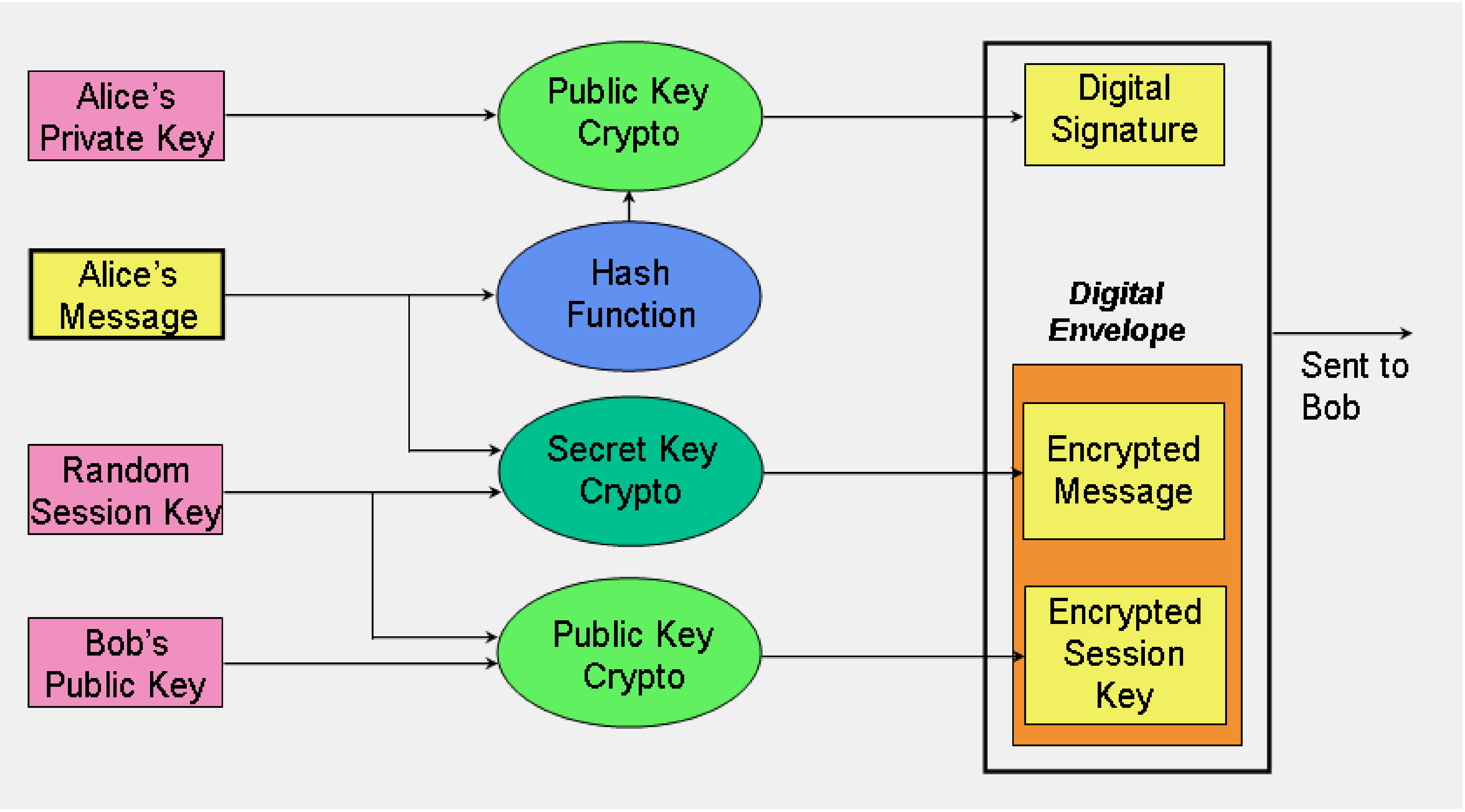
Hash function

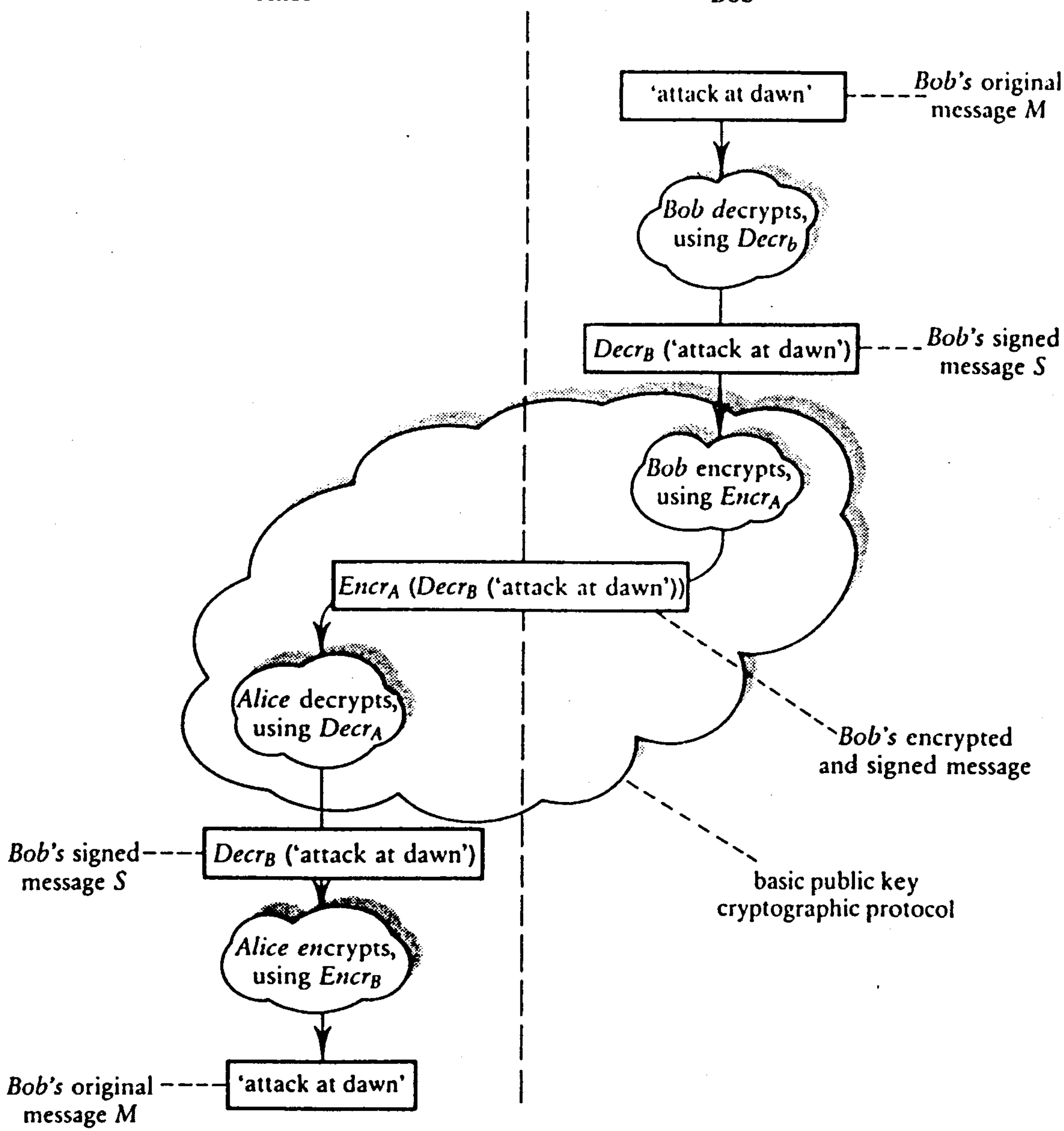
- A hash function H is **one-way** if it is hard to invert, where "hard to invert" means that given a hash value h, it is computationally infeasible to find some input x such that H(x) = h.
- If, given a message x, it is computationally infeasible to find a message y not equal to x such that H(x) = H(y) 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.

Hash function



Hybrid cryptographic scheme



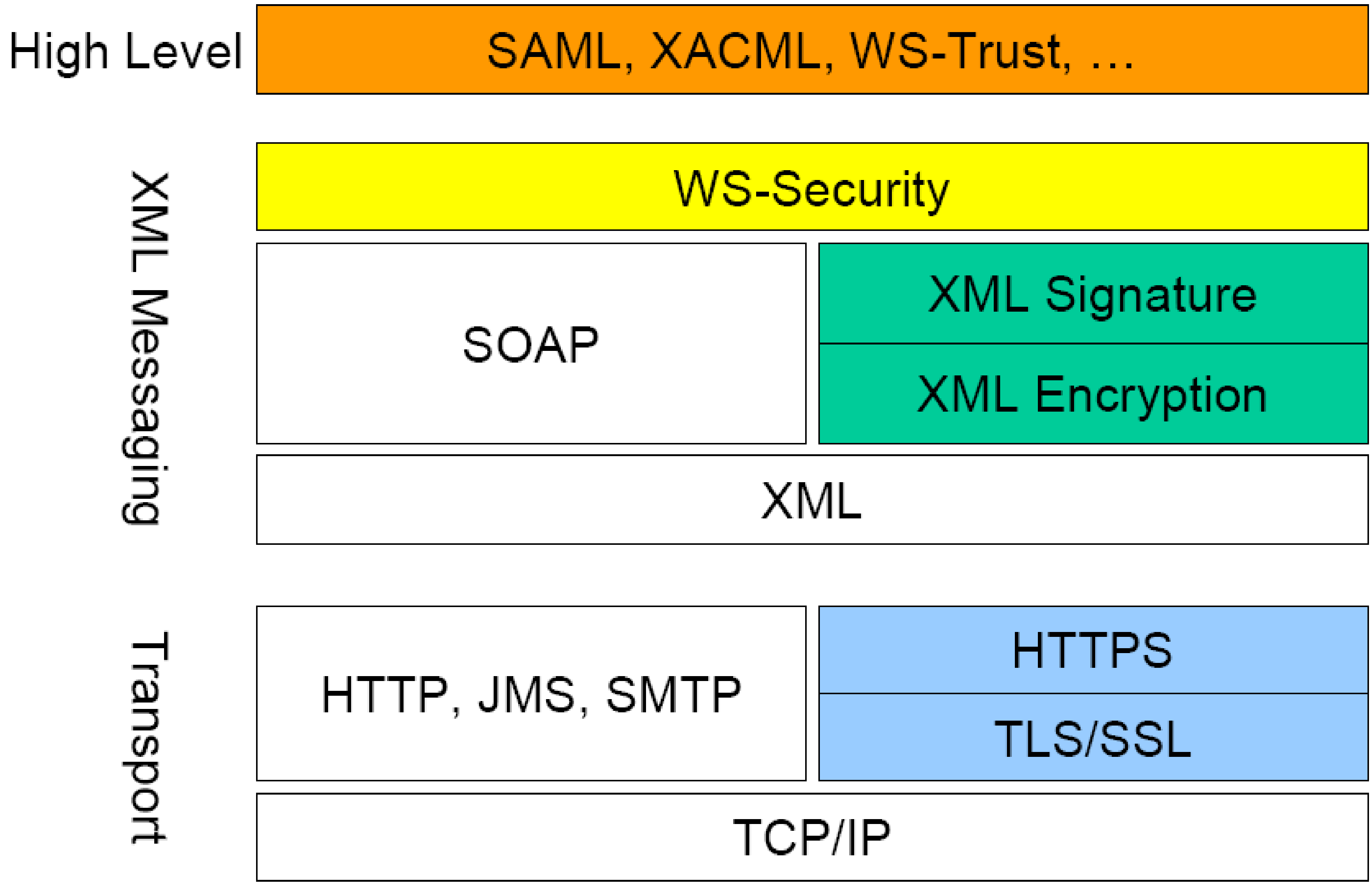

'attack at dawn'
Bob's original message M
Bob decrypts, using Decr_b
Decr_B ('attack at dawn')
Bob's signed message S
Bob encrypts, using Encr_A
Encr_A (Decr_B ('attack at dawn'))
Bob's encrypted and signed message
Alice decrypts, using Decr_A
basic public key cryptographic protocol
Bob's signed message S
Decr_B ('attack at dawn')
Alice encrypts, using Encr_B
Bob's original message M
'attack at dawn'

Cryptography in public channels

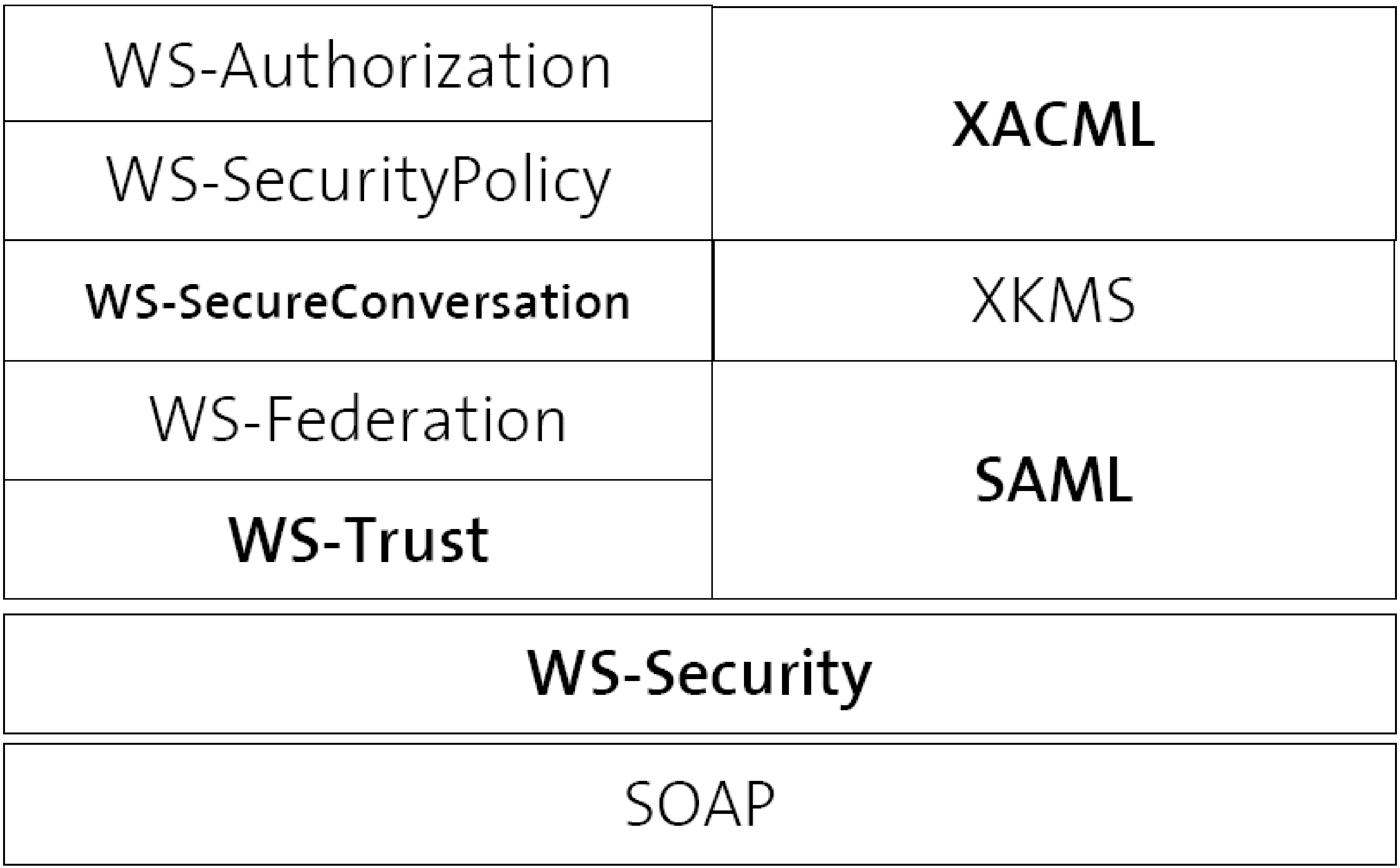
- Both communication party exchange public keys
- Exchange of random session key using public key cryptography
- Private key cryptography using session key for communication
- Public key distribution problem – **Man in the middle attack** (unavoidable on single channel)
- Private key algorithms problem (not so bad – OTP, AES, 3DES)

Web Services Security

Web Services Security



Web Services Security



Web Services Security

- XML Signature (XMLDSIG): Message Integrity and Sender/Receiver Identification
- XML Encryption (XMLENC): Message Confidentiality
- WS-Security (WSS): Securing SOAP Messages
- SAML: Interoperable security metadata exchange
- XACML: Access Control

Web Services Security

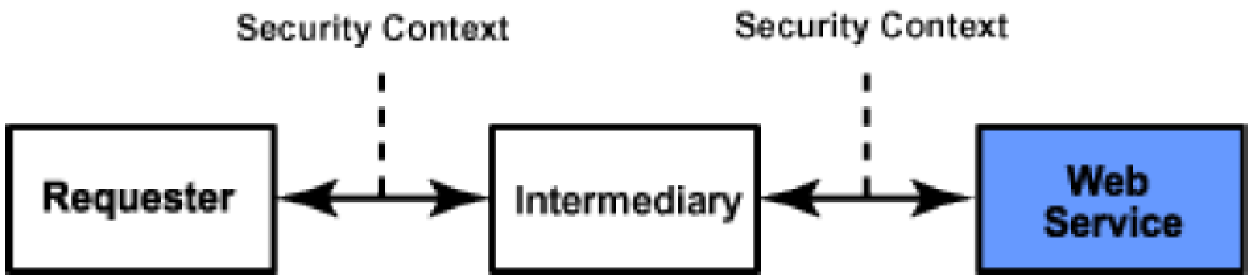
- WS-Trust and WS-Federation: Federating multiple security domains
- WS-SecureConversation: Securing multiple message exchanges
- WS-SecurityPolicy: Describing what security features are supported or needed by a Web service
- XrML: Digital Rights Management
- XKMS: Key Management and Distribution

Web Services Security

- Point-to-point
  - Communication channel level
  - Guarantied on the server basis
- End-to-end
  - Message level
  - Application oriented security

Web Services Security

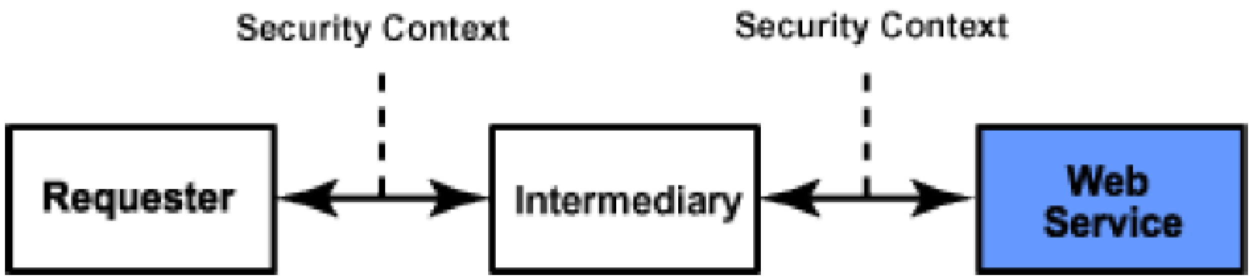
- Point-to-point



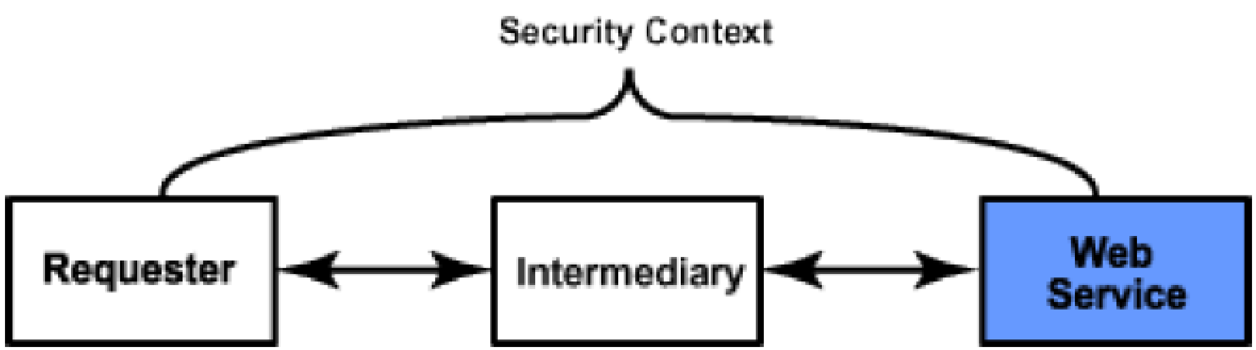
- End-to-end
  - Message level
  - Application oriented security

Web Services Security

- Point-to-point

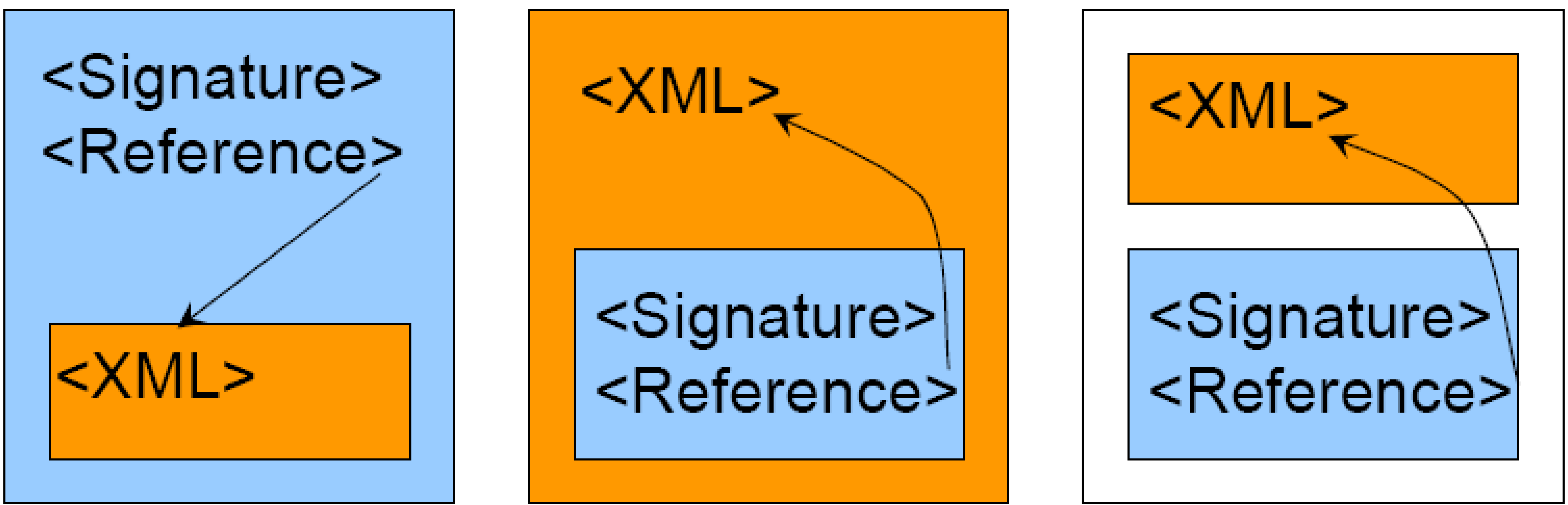


- End-to-end



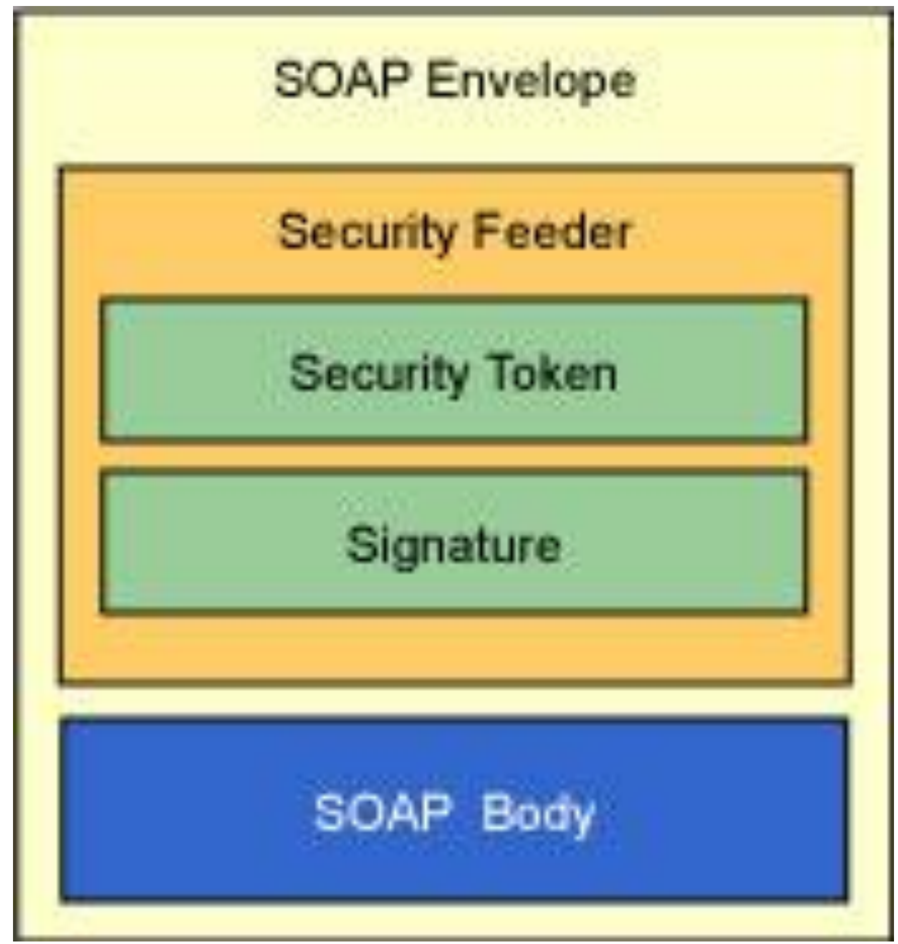
Web Services Security

- XML Signature:
  - Entire XML document
  - Parts of XML doc
  - Integrity and Identity

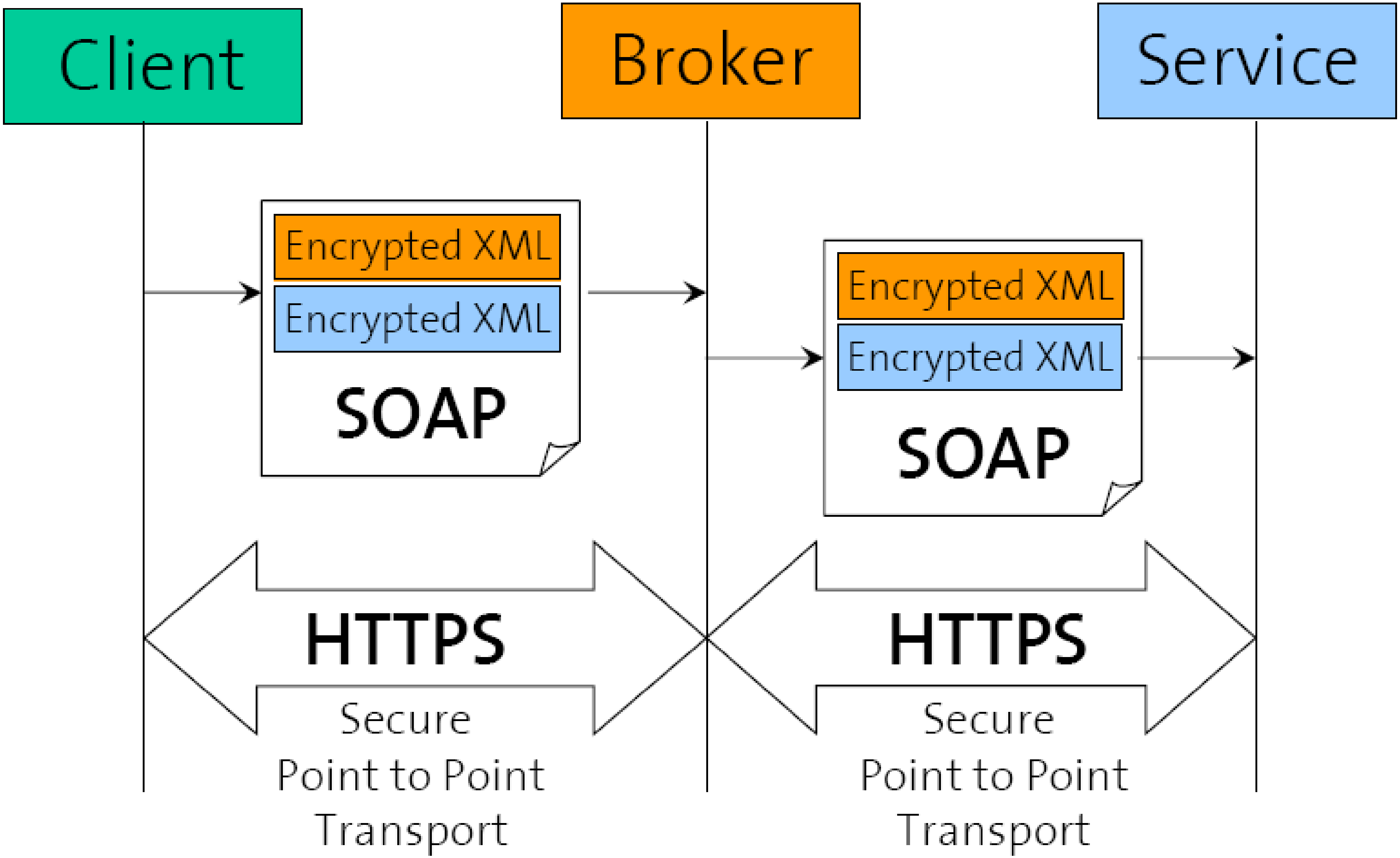


Web Services Security

- XML Encryption
  - Confidentiality of messages
  - End-to-end
  - Full or partial



Web Services Security



Web Services Security

Original XML Document

```
<Employee>
    <ID>222-654-456</ID>
    <Name>Markus Bach</Name>
    <Salary currency="CHF">100000</Salary>
</Employee>
```

Encrypted XML Document

```
<Employee>
    <ID><EncryptedData>...</EncryptedData></ID>
    <Name>Markus Bach</Name>
    <EncryptedData>...</EncryptedData>
</Employee>
```

Web Services Security

Message Security

Disadvantages

- Immature standards only partially supported by existing tools
- Securing XML is complicated

Advantages

- Different parts of a message can be secured in different ways.
- Asymmetric: different security mechanisms can be applied to request and response
- Self-protecting messages (Transport independent)

Transport Security

Advantages

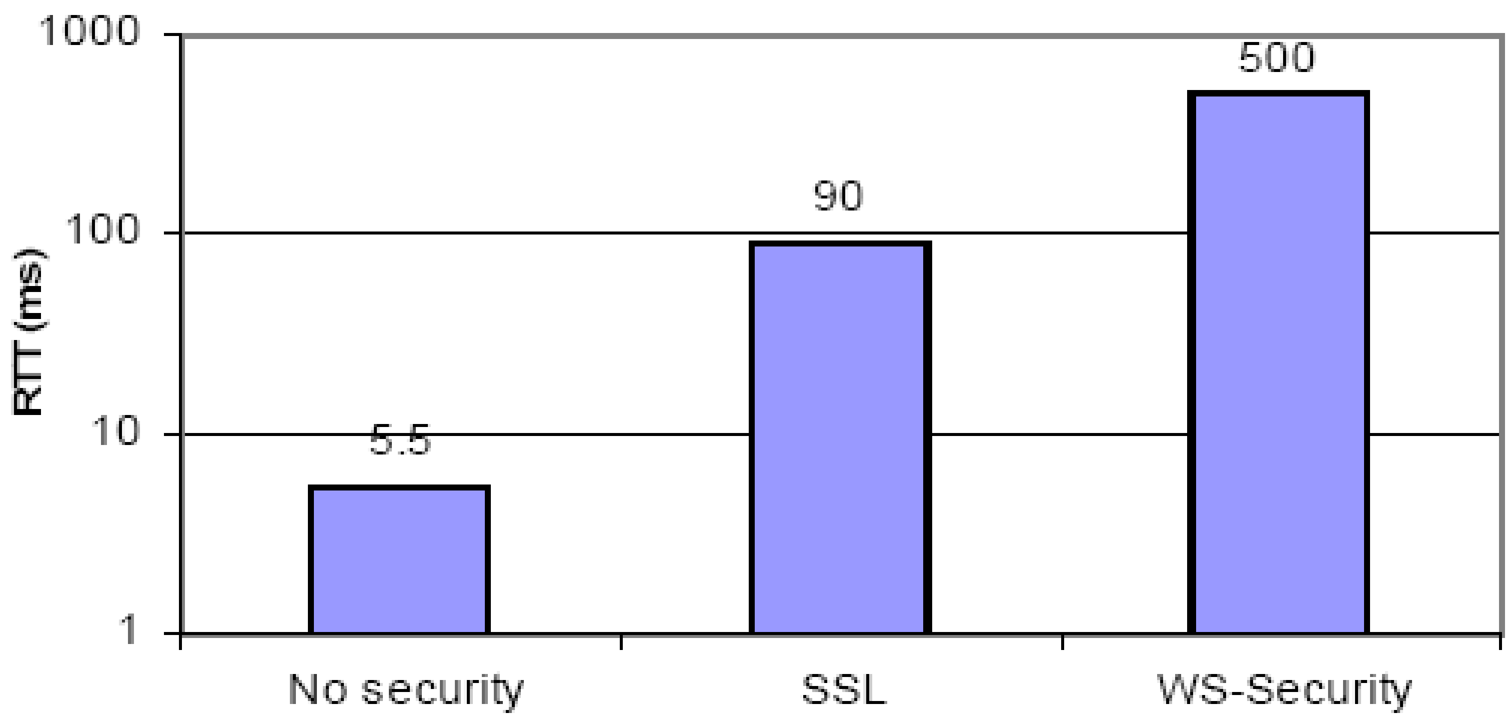
- Widely available, mature technologies (SSL, TLS, HTTPS)
- Understood by most system administrators

Disadvantages

- Point 2 Point: The complete message is in clear after each hop
- Symmetric: Request and response messages must use same security properties
- Transport specific

Performance: SSL vs. WS-Security

- 8 clients saturate a server with small messages (5 bytes payload)
- Apache XML Sec, Tomcat, Linux, Dual Xenon 2.8GHz, 2GB RAM (Shirasuna et.al., 2004)



Performance: XML overhead

- Apache, Linux, P4 2.79GHz, 768MB RAM (Liu et.al., 2005)
- It takes 10ms to sign or encrypt 100KB
- Using WS-Security takes 100-200ms to do the same

	WS-Security (enc.only)	*HTTPS*
RSA (No. operations)	6	6
DES (% of content processed)	150%	300%
XML overhead (% of content processed)	150%	0
No. SSL Negotiations	0	6

Performance: XML overhead

- Shape of the document greatly affects performance
- More structured = bigger overhead