

SWAP framework

Part I: services, usability, and trust

Kjell Jørgen Hole



Last updated June 10, 2008

Outline

1. Glossary: terms and definitions
2. Vision: the SWAP* framework
3. Main success scenarios
4. Usability
5. Trust

* Secure Wireless Application Programming

2

Glossary

(terms and definitions)

- **Client**—software that initiates a connection and sends requests
- **Server**—software that listens for connections and processes requests
- **Entity**—human, business enterprise, computer, handheld device, etc.
- **Individual**—a single human as distinct from a group
- **User**—human using client software

3

Glossary ...

- **Identifier**—pointer to an entity, e.g., (personal) name, serial number, and IP address
- **Identification**—the process of using claimed or observed attributes of an individual to infer who the individual is
- **Authorization**—the process of deciding what an entity ought to be allowed to do

4

Types of authentication

- **Individual authentication**—the process of establishing an understood level of confidence that an identifier refers to a specific individual
- **Entity authentication**—the process of establishing an understood level of confidence that an identifier refers to a specific entity
 - it may or may not be possible to link the authenticated entity to an individual

5

Vision: the SWAP framework

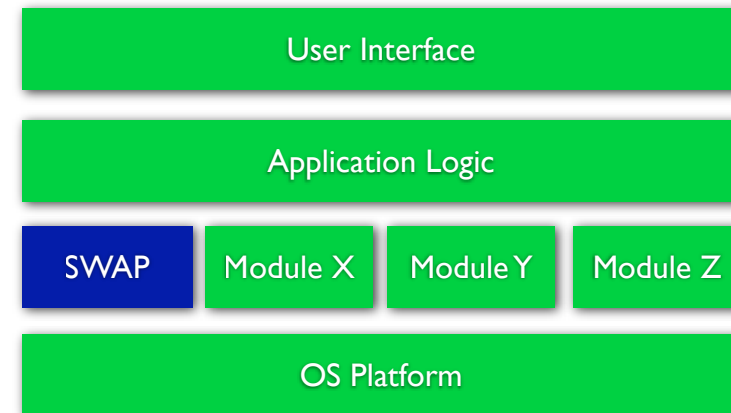
6

Framework description

- Secure **end-to-end communications** for client-server applications
- Direct client-to-client communication also possible
- Smart phones run client software and stationary PCs run server software
- Secure storage on smart phones
- Based on a Public-Key Infrastructure (PKI)

7

Middleware



(independent of implementation technologies)

8

Security services

- **Secure communication**
 - individual and entity authentication
 - data confidentiality
 - data integrity
 - data origin authentication
- **Secure storage**
 - data confidentiality
 - data integrity

9

Security services ...

- **Certificate management**
 - generation of public-private key pairs
 - issuing of public-key certificates
 - storage of valid public-key certificates*
 - storage of expired public-key certificates*
 - storage of revocation information

* Needed for logging purposes

10

Advantages of framework

1. Provides application programmers with an API that reduces the complexity of developing client-server applications with a high level of security
2. Documentation shows developers how to use API
3. Shorter time to market for new secure applications

11

Scenarios

A technique for capturing the functional requirements of a system

12

Health care services on a smart phone

1. Medical doctors update patients' medical records
2. Doctors upload/download pictures of patients
3. People download personal medical information
4. Hospitals monitor patients' health using equipment that communicate with smart phones

13

Scenarios

- In the following we consider the **main success scenarios** for the four mentioned health care services
- The scenarios illustrate why the framework should support
 - secure communication
 - secure storage
 - key management

A main success scenario is the "typical flow" in a use case

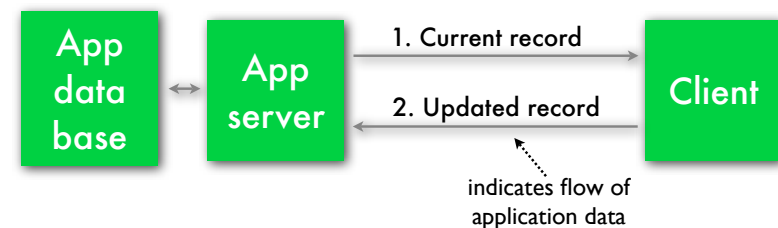
14

I. Updating medical records

1. Smart-phone client asks (application) server for medical record belonging to a particular patient
2. Server obtains record from medical data base
3. Server sends record to client
4. Client updates record
5. Client sends updated record to server
6. Server stores updated record in medical data base

15

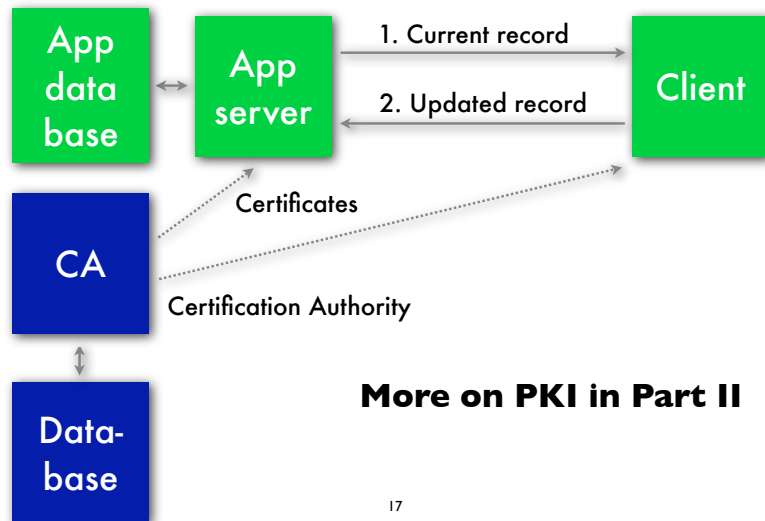
Security needs



- Needed services: *secure communication* and *certificate management*
- **Remark.** We do not consider *accountability* (who did what when)

16

Adding a PKI



2. Pictures of patients

1. Smart-phone client asks server for medical picture(s) of particular patient
 2. Server obtains picture(s) from data base
 3. Server sends picture(s) to client
 4. Client displays picture(s)
 5. Client takes new picture(s) of patient
 6. Client uploads new picture(s) to server
 7. Server stores picture(s) in data base
- 18

Security needs



- Need the same services as in the previous example
- A PKI can be used

3. Download of personal medical information

1. Smart-phone client requests medical information (belonging to a particular person) from server
 2. Server retrieves information from data base
 3. Server sends information to client
 4. Client displays information
- 20

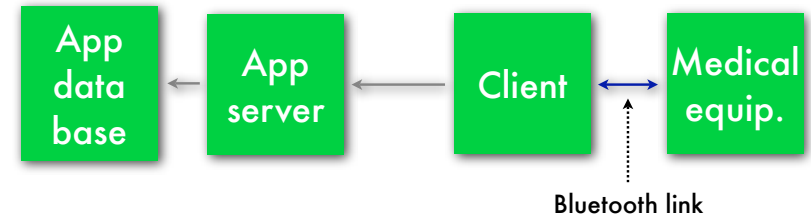
3. Download of personal medical information



- Same services as before
- In addition, the phone must provide *secure storage*
- A PKI is needed

21

4. Health monitoring

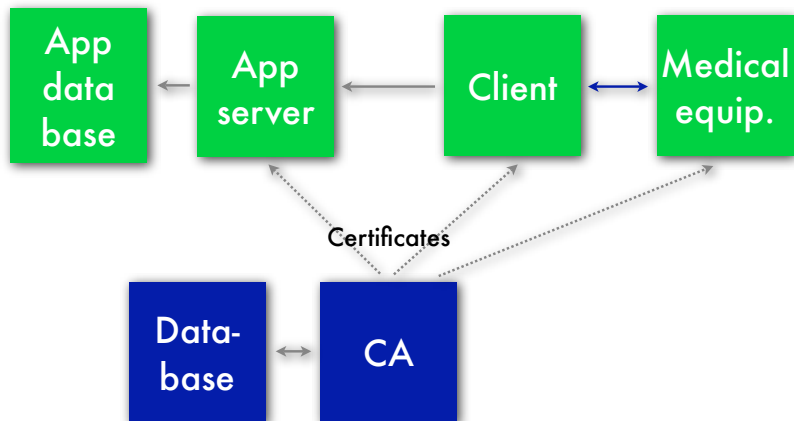


- Same services as in the previous example
- Also, there is a need for secure communication over the Bluetooth link

Client-to-client communication!

22

Adding a PKI



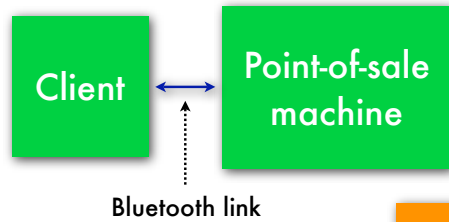
23

Smart phone m-commerce

1. Electronic wallet
2. Gaming
3. Internet banking
4. Buying audio books and music
5. Buying or updating travel insurance
6. Gambling, e.g., poker

24

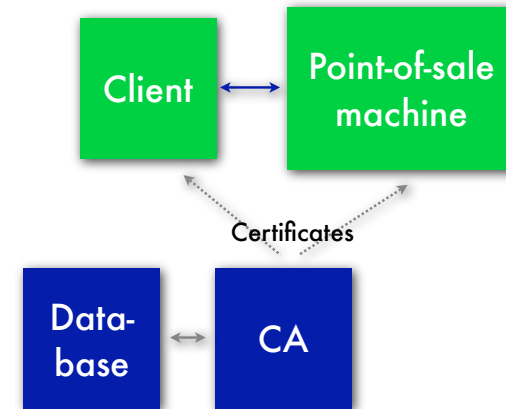
I. Electronic wallet



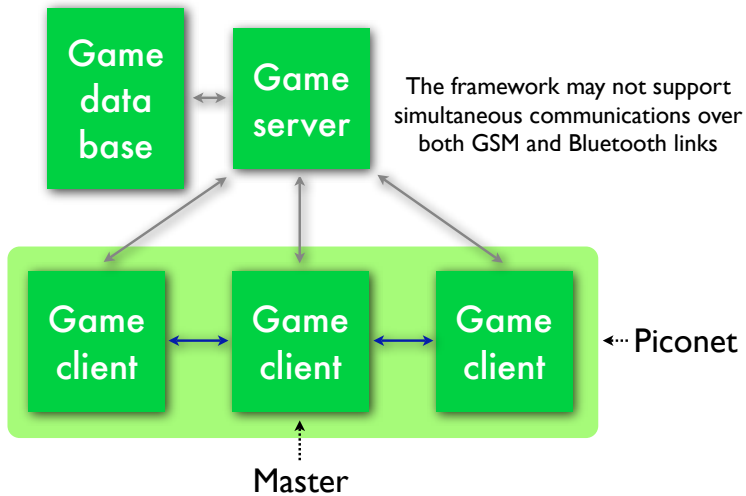
- Secure storage of e-cash
- Secure Bluetooth communication
- Note that only “local” communication takes place

Client-to-client communication!

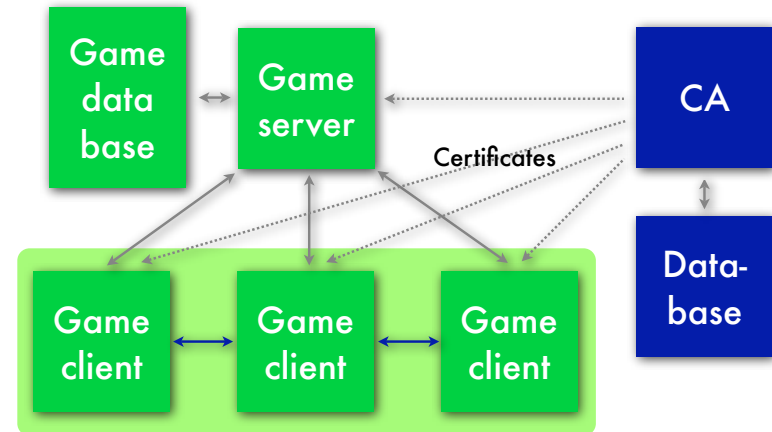
Adding a PKI



2. Gaming



Adding a PKI



Usability

If people are unable to use secure computers, they will use computers that are not secure

29

Lack of usability (1)

- Users do not understand
 - the difference between a public and a private key
 - the role of certificates
- **Unfortunately, there is no intuitive model that explains the properties of a PKI**
- important to make (most of) the PKI transparent to the user

30

Lack of usability (2)

- Users do not understand the connection between the PKI and the application goal they are trying to achieve
 - connection between secure email and key pairs
- Users find it difficult to
 - configure an application to use certificates
 - install certificates

31

Make PKI usable

- Many years of experience has shown that it is very difficult to create a large, general-purpose PKI
 - complicated interconnections of CAs
 - no global unique names to use in certificates
- **Better to develop a small-scale PKI framework that supports a *single* client-server application run by a *single* organization**
- (nearly) automatic set up is needed

32

Usability goals

1. Provide programmers with a simple security API
 - avoid functionality the average programmer will never use
2. Make it easy for users to verify certificates
3. Make the certificate generation, installation, and revocation mostly transparent to the user
4. Make it simple for the server administrator to configure security features
 - web-based server configuration

33

Trust

With trust, people will buy things, and without it, they will not

Trust is meaningless without risk

34

Initial definition of trust

An individual X **trusts** another individual Y when X assumes it knows exactly how Y will behave

- Note that trust is “directional”; X trusts Y, but Y may not trust X
- **Example:** a user trusts a CA (Certification Authority) when she assumes that the binding between the name and the public key in an issued certificate is correct

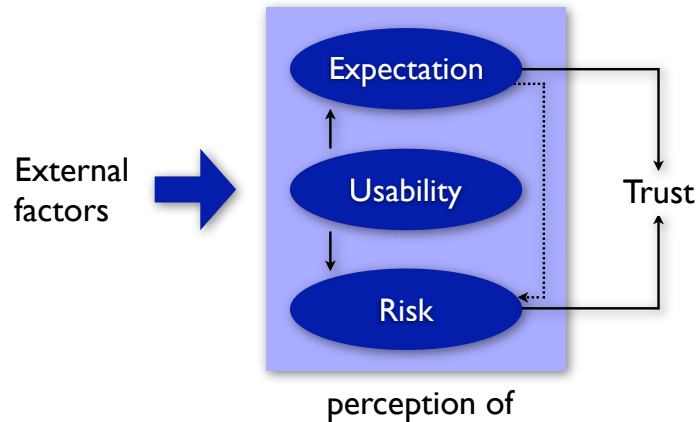
35

More general definition: levels of trust

- While a user has **complete trust** or **no trust** in a CA, we often associate different levels of trust when we consider a (professional) relationship between two individuals
 - a user doesn't necessarily have complete trust in the individual named in a certificate even though he completely trust the CA
- The level of trust depends on, among other things, the user's expectations and the usability of the system

36

Model for varying level of trust



Model discussion

- Level of trust depends on
 - context
 - behavior of entities
 - understanding of risk
 - prior experiences
 - reputation of entities
- Model is useful when analyzing PKI

38

Building trust

- Mechanisms for trust building:
 - legally binding contracts
 - openness
 - reputation (word of mouth)
 - live up to expectations
- Trust is easy to lose and difficult to regain

39

SWAP trust model (I)

- It is assumed that the central framework, the application server, and the clients are owned and operated by separate entities
- in Part III we'll also consider the case when the framework and the application server are owned (and operated) by the same entity

Central
framework
(CA, RA, database)

Application
owner
(app server,
database)

Client

40

SWAP trust model (2)

- To obtain a **strong identification**, the framework incorporates an identification procedure based on direct personal knowledge of individuals
- The application owner trusts the framework to identify all new users. The amount of trust is regulated by a **legally binding contract** between the application owner and framework owner
- the contract defines the area of trust

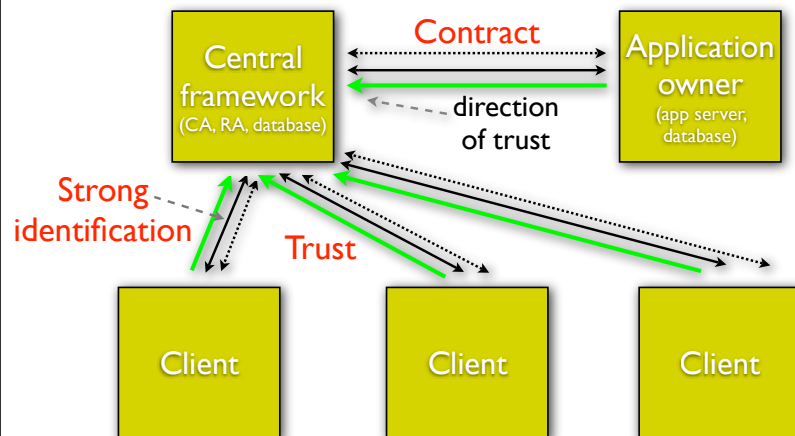
41

SWAP trust model (3)

- A user trusts the framework to identify the application server correctly
- A user trusts the **secure communication**, **secure data storage**, and **certificate management** services provided by the framework
- The amount of trust is regulated by a **legally binding contract**
- since client and server OS platforms are insecure, hardware modules must protect cryptographic keys

42

Trust relationships

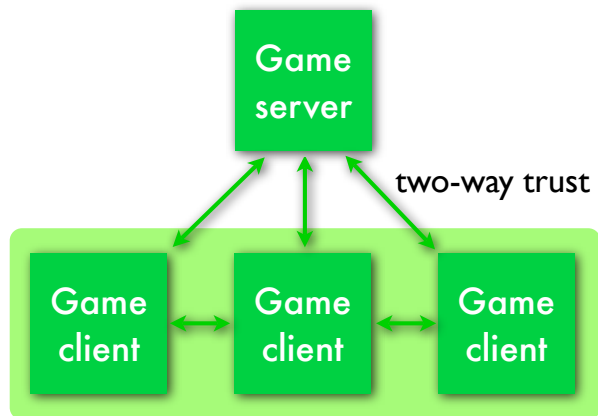


Gaming example revisited

- The application running on top of SWAP may introduce the need for additional trust
- **Example:** Game server ranking players
 - must bind identity to player
 - identity must be contained in ranking list
 - player must trust server to maintain correct ranking
 - server must trust players not to cooperate

44

Two-way trust



45

Sources

- National Research Council, *Who Goes There?* National Academies Press, 2003
- L. F. Cranor and S. Garfinkel, editors, *Security and Usability*, O'Reilly, 2005
- S. Bibb and J. Kourdi, *Trust Matters*, Palgrave Macmillan, 2004

46

