Asymmetric Cryptography

Lukasz Pater
CMMS Administrator and Developer

EDMS 1373428

Agenda

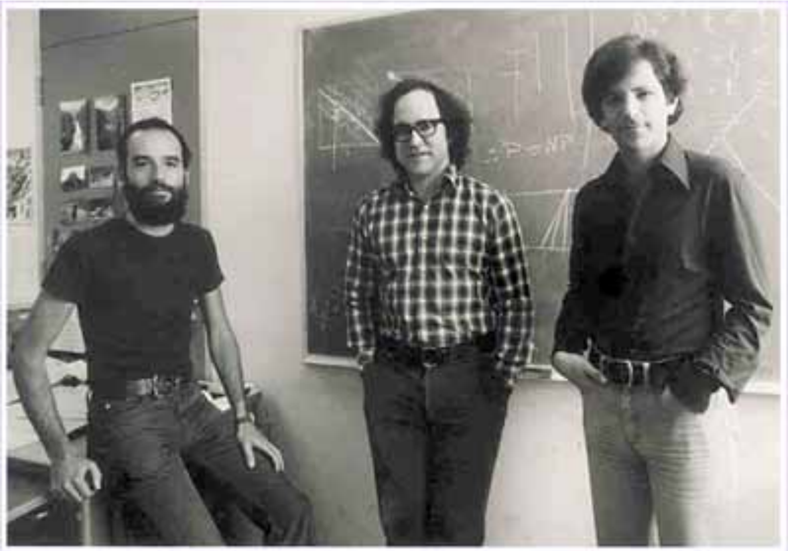
- **Introduction**
  - Why do we need asymmetric ciphers?
  - One-way functions
- **RSA Cipher**
- **Message Integrity**
- **Examples**
  - Secure Socket Layer
  - Single Sign On
- **Conclusion**

Introduction

Challenge: How perfect strangers can send each other encrypted messages?

Powerful Idea: Public-key encryption!

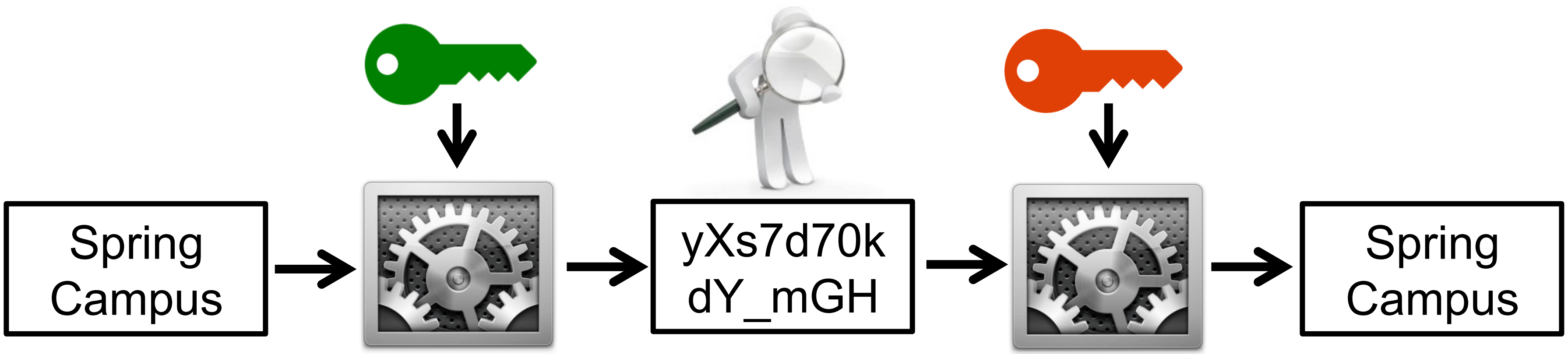
Diffie, Hellman, Merkle [1976]



Rivest, Shamir, Adleman [1977]

Introduction

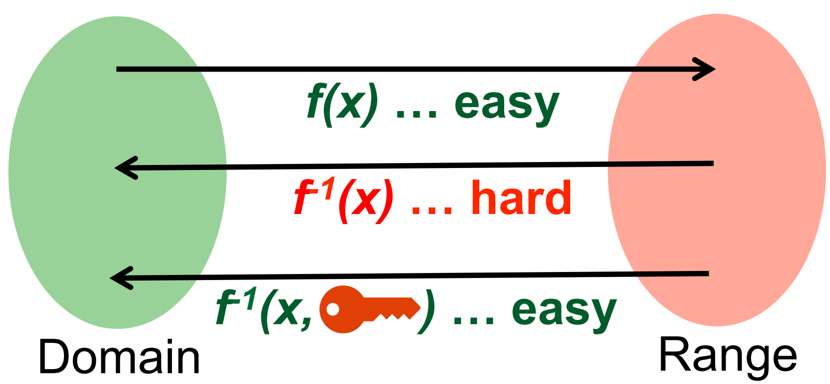
- **Symmetric Cryptography**
  - Requires that sender and receiver know shared secret key
- **Asymmetric Cryptography – radically different approach**



  - Sender and receiver do not share the same key
  - **Public** encryption key known to all
  - **Private** decryption key know only to the receiver

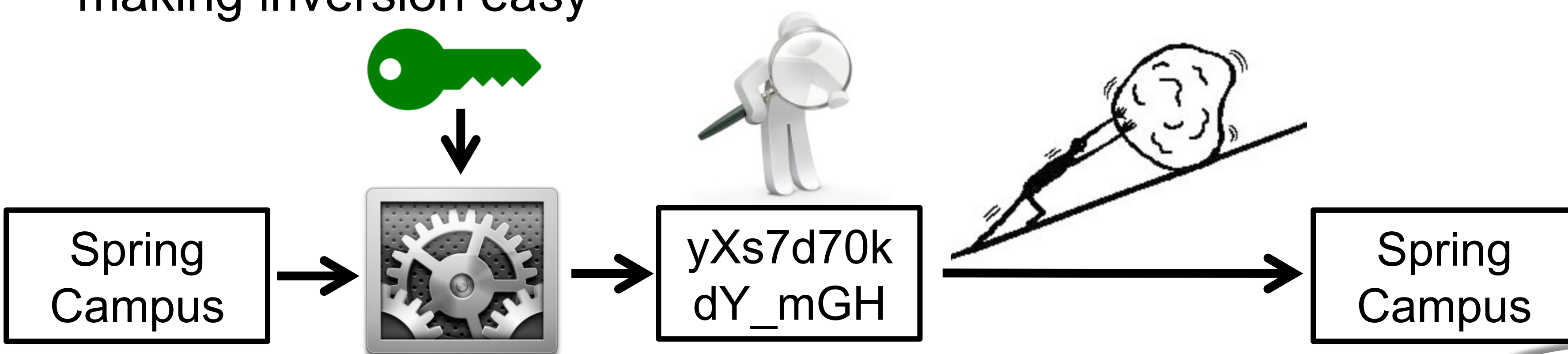
One-way functions

- **Most functions are invertible; for any *f(x)* = *y*, there is an *f^{-1}(y)* = *x* (e.g. division, multiplication, DES)**
  - *$f(x) = 2 \bullet x$, $f^{-1}(x) = x / 2$ …*
- **A function which is easy to compute in one direction, but difficult to compute in the other, is called *one-way function***
  - Polynomial factorization, hashing, modular arithmetic, …
  - One-way function that can be easily inverted with additional knowledge ( ) is known as ***trapdoor one-way function***



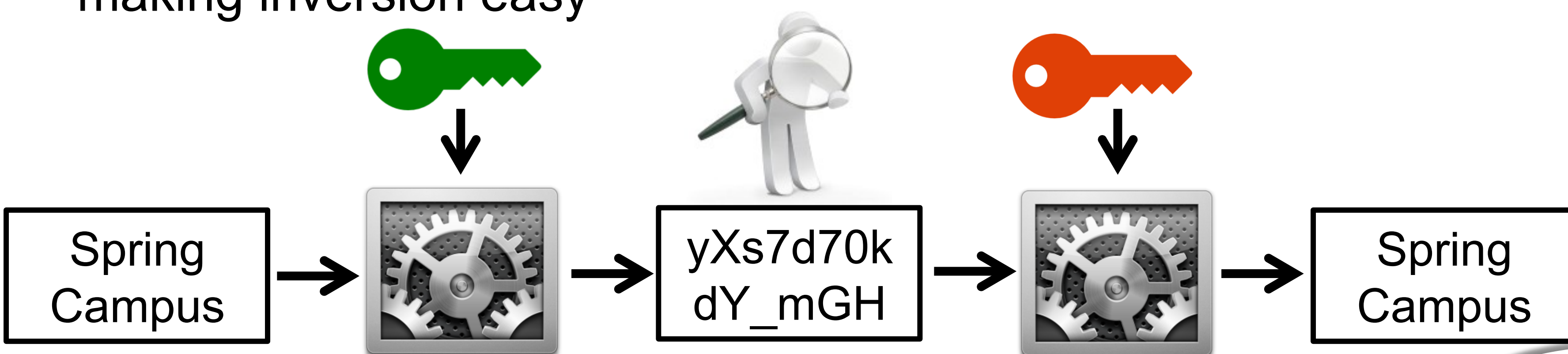
One-way functions and cryptography

- **Public key encryption is based on the existence of trapdoor one-way functions**
  - Encryption, done with the public key ( ), is very easy
  - Decryption is computationally hard (factorization of large numbers, discrete logarithm problem)
  - Knowledge of the private key ( ) "opens the trapdoor", making inversion easy



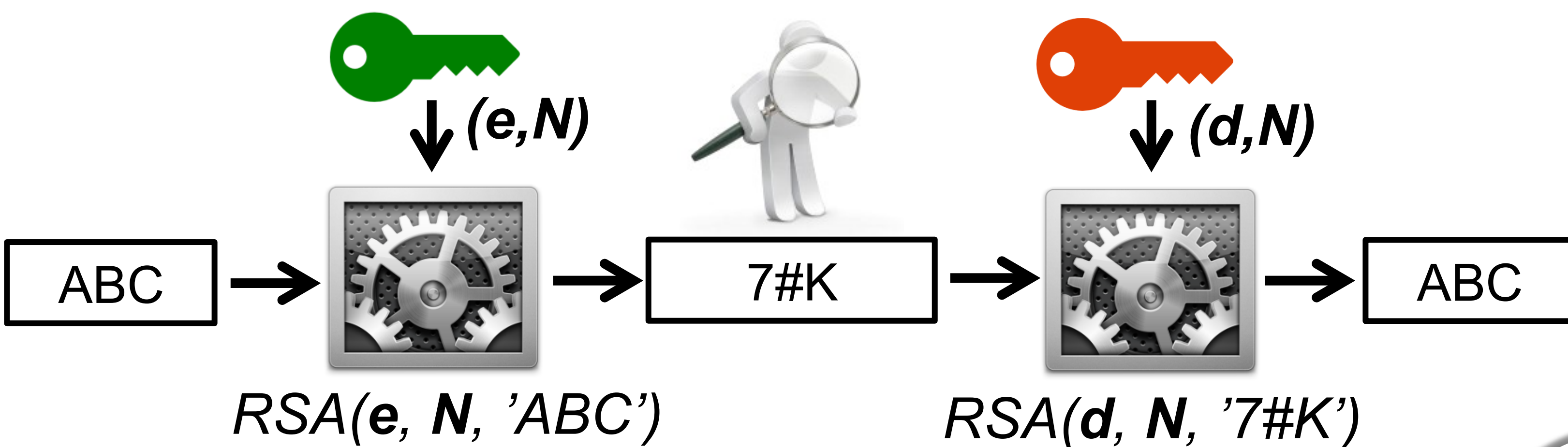
One-way functions and cryptography

- **Public key encryption is based on the existence of trapdoor one-way functions**
  - Encryption, done with the public key ( ), is very easy
  - Decryption is computationally hard (factorization of large numbers, discrete logarithm problem)
  - Knowledge of the private key ( ) "opens the trapdoor", making inversion easy



RSA (**R**ivest, **S**hamir, **A**dleman)

- **RSA is the most well-known and common public key cryptosystem**
- **Basic notation: a key pair *(e,d,N)* contains two keys:**
  - ***(e,N)*** is the public key, ***(d,N)*** is the private key
  - Let RSA be the encryption function and ‘ABC’ plaintext message



RSA Algorithm

- **RSA's security is based on modular arithmetic**
  - *x mod n*; remainder of *x* when divided by *n (5 mod 2 = 1)*
  - *p* and *q* are relatively prime if their greatest common divisor is 1
  - *p* and *q* are multiplicative inverse when (*p • q = 1 mod n*)
- **RSA algorithm**
  - Pick two large prime numbers *p* and *q*, let *N = p • q*
  - Select small integer *e* that is relatively prime to *(p-1)(q-1)*
  - Find *d*, the multiplicative inverse of *e mod (p-1)(q-1)*
  - ***(e,N)*** is the public key, to encrypt compute $C = M^e \bmod N$
  - ***(d,N)*** is the private key, to decrypt compute $M = C^d \bmod N$

RSA Algorithm / Example

- **Pick two large prime numbers *p* and *q*, let *N = p • q***
  - *p = 3, q = 11, N = p • q = 33*
- **Select small integer e that is relatively prime to *(p-1)(q-1)***
  - *(p-1)(q-1) = 20,* **e = 3** *(prime to 20)*
- ***Find d*, the multiplicative inverse of *e mod (p-1)(q-1)***
  - *Find d, that d • 3 mod 20 = 1, d = 7*
- ***(e,N)* is the public key, to encrypt compute M^e *mod N***
  - *(3,33)* is the public key, 2^3 *mod 33 =* ***8***
- ***(d,N)* is the private key, to decrypt compute M^d *mod N***
  - *(7,33)* is the private key, 8^7 *mod 33 =* **2**

Strengths of RSA

- **Why is RSA secure?**
  - Suppose you know the public key *(e,N)*, $N = p \bullet q$
  - The security of RSA is dependent on the assumption that it's difficult to determine the private key *d* from *(e,N)*
  - Essentially need to find factors of *N* without knowing *p* and *q*
  - Factoring is **thought** to be computationally hard **(no proof!)**
- **Some statistics for the fastest known factoring algorithm**

Key Size	MIPS-years required to factor
512 bits	30,000
1024 bits	300,000,000,000
2048 bits	300,000,000,000,000,000,000

Message Integrity

- **Allows communicating parties to verify that received messages are authentic**
  - Content of the message has not been altered
  - Source of the message is who/what you think it is
- **Message Digests**
  - A Hash function *H()* that takes as input an arbitrary length message and outputs a fixed-length string – **message digest**

Music — bash — 68×5

```
lpatermac:Music lukaszpater$ shasum Sak_Noel_-_Loca_People.mp3
f62d2ca0425717ede8d85bb3a6650fe02b2c3fac  Sak_Noel_-_Loca_People.mp3
lpatermac:Music lukaszpater$ 
```

Message Integrity

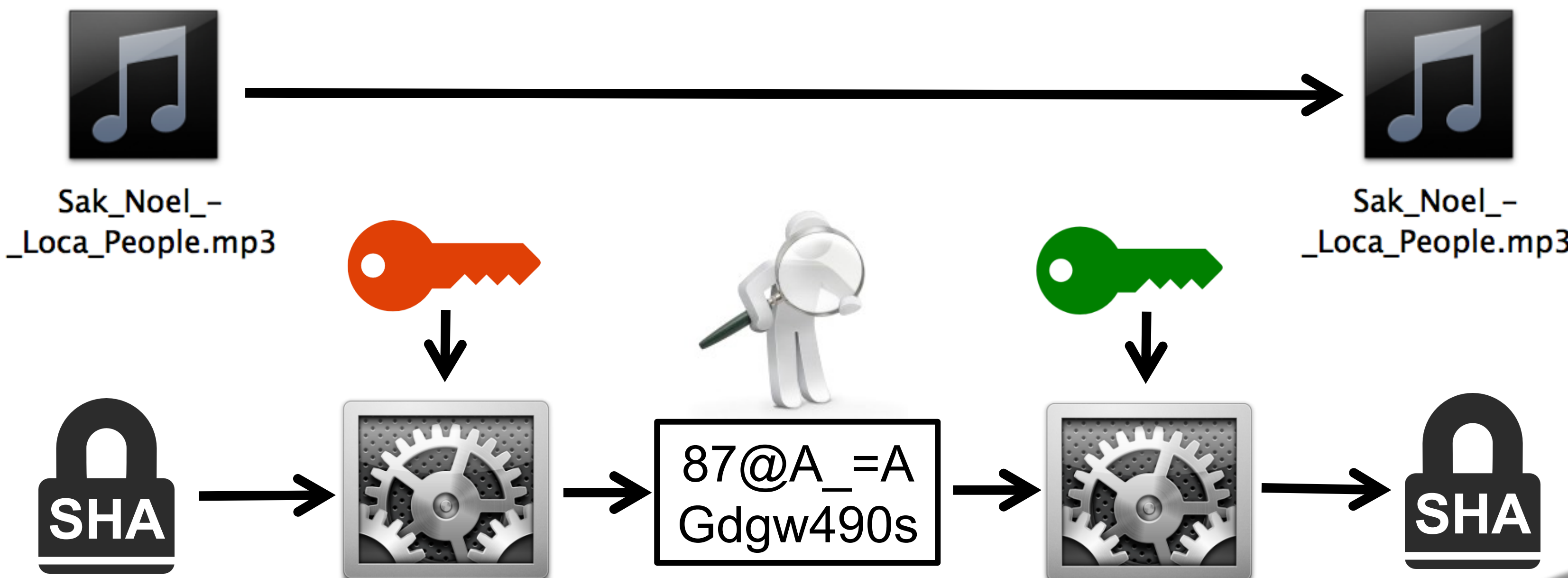
- **Allows communicating parties to verify that received messages are authentic**
  - Content of the message has not been altered
  - Source of the message is who/what you think it is
- **Message Digests**
  - A Hash function *H()* that takes as input an arbitrary length message and outputs a fixed-length string – **message digest**

Music — bash — 68×5

```
lpatermac:Music lukaszpater$ shasum Sak_Noel_-_Party.mp3
5fb5403ff5d430bd0ffe6879ab565eb612f3265f  Sak_Noel_-_Party.mp3
lpatermac:Music lukaszpater$
```

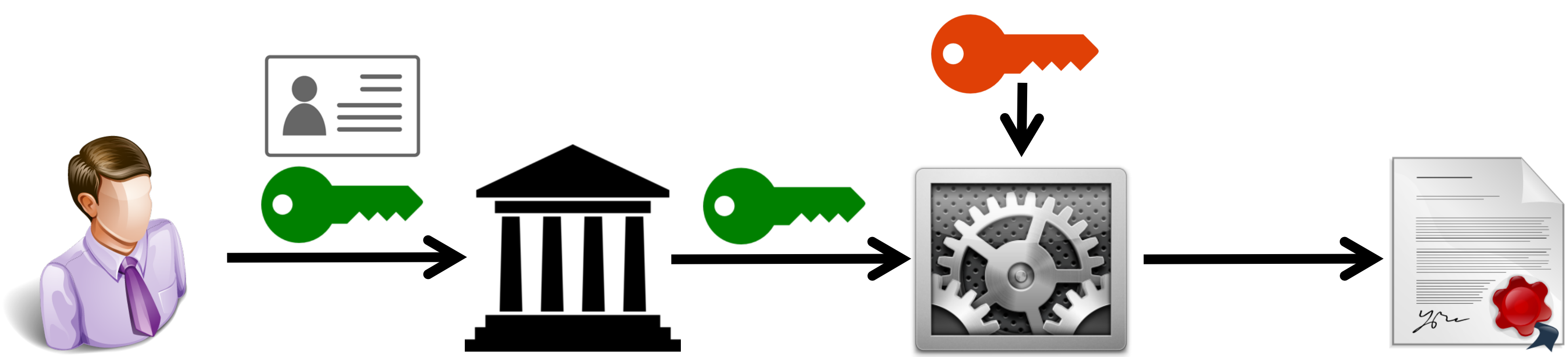
Digital Signatures

- **Cryptographic technique analogous to hand-written signatures**
  - Sender digitally signs the digest of the document (SHA) with his/hers private key ( ), establishing that he is the owner



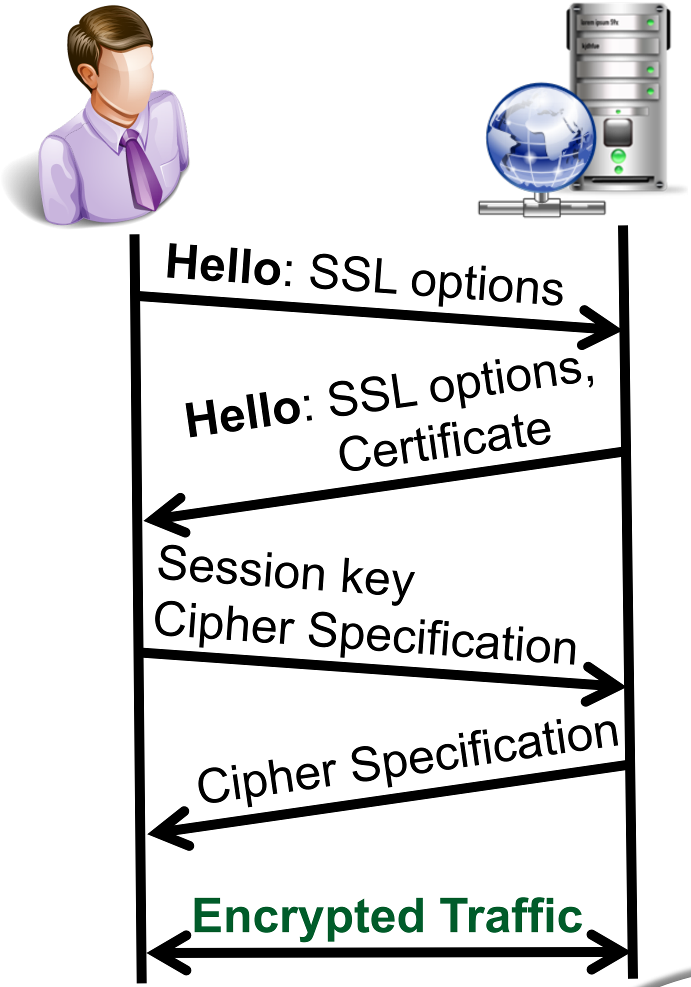
Certification Authorities

- **Certification Authority (CA) binds public key to particular entity, for example: to Bob**
  - Bob provides “proof of identity” to the CA
  - CA creates certificate binding Bob to his public key
  - Certificate ( ) containing Bob’s public key ( ) digitally signed by CA’s private key ( ) says: “This is Bob’s public key”

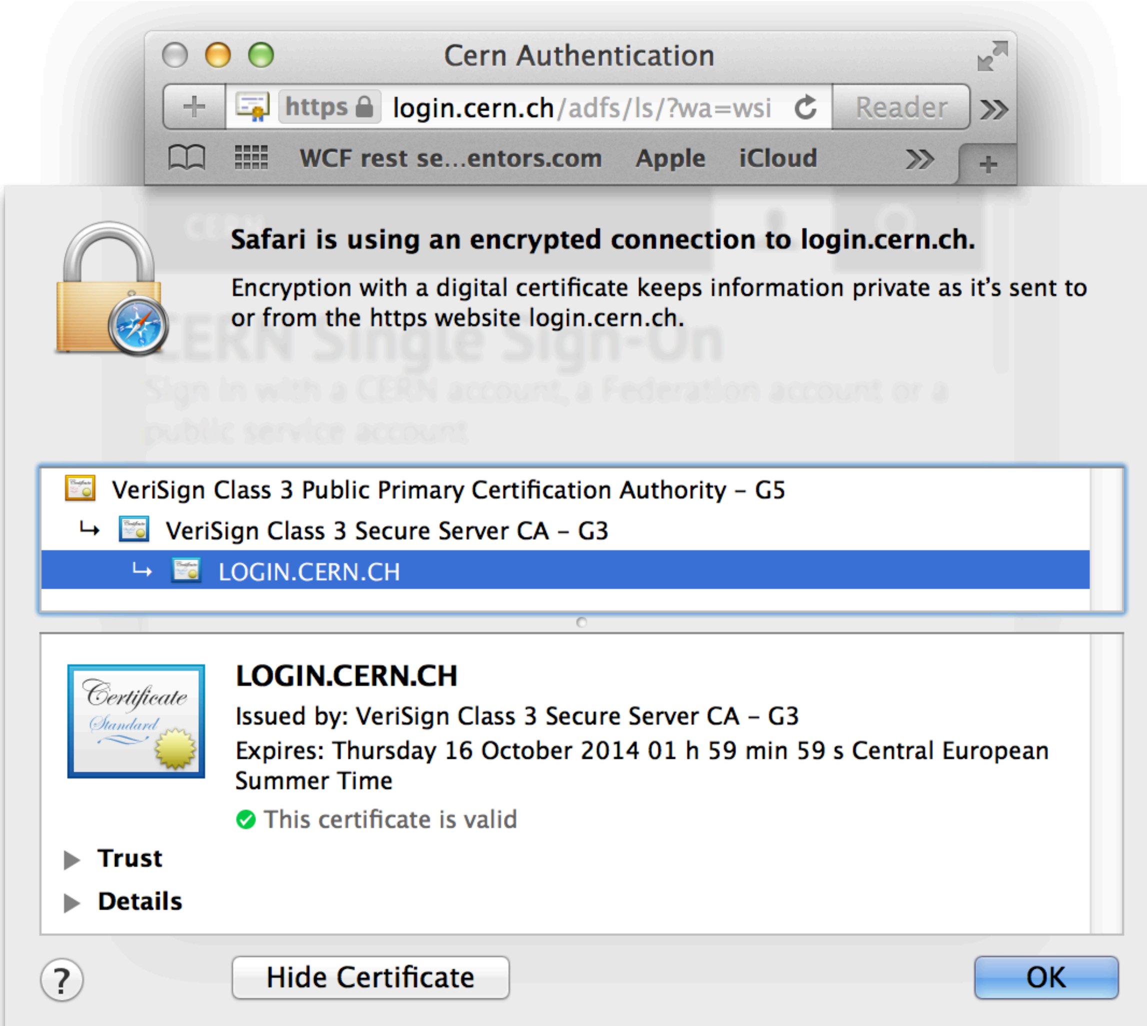


Secure Socket Layer (SSL)

- **TCP provides reliable end-to-end service**
- **TCP with SSL provides reliable and secure end-to-end service**
  - HTTPS: HTTP over SSL
  - Subsequently became Internet standard know as TLS
  - Provides **message integrity** (hash functions) and **message confidentiality** (symmetric encryption with shared secret key)

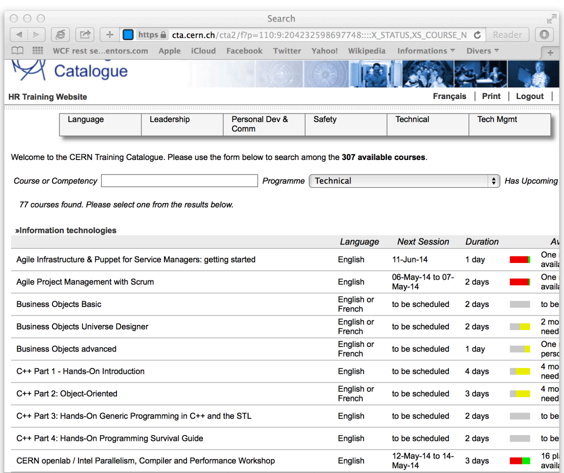


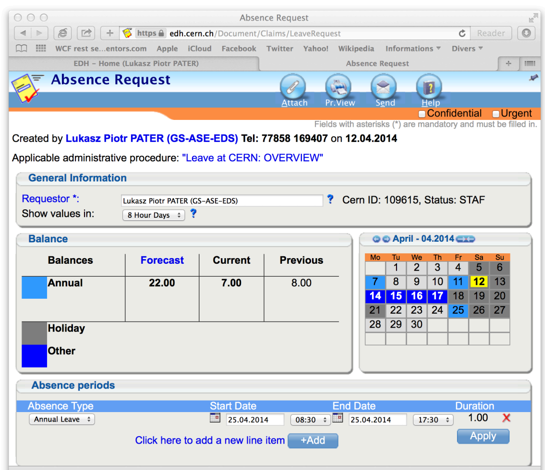
Secure Socket Layer (SSL)

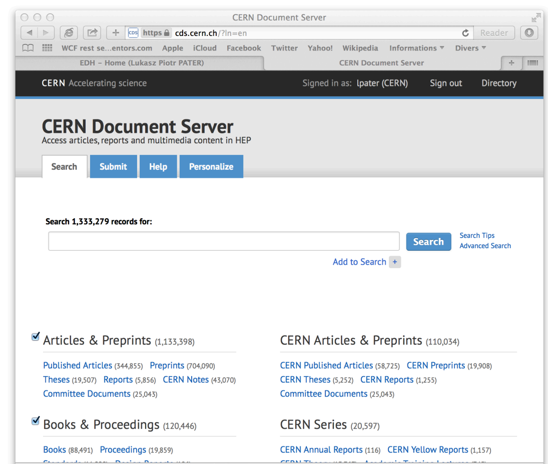


Single Sign On (SSO)

- **Scenario during typical work day (at CERN)**
  - Sign in for an absence request
  - Sign in to subscribe to one of the technical trainings
  - Sign in to order a book from the library
  - Sign in …

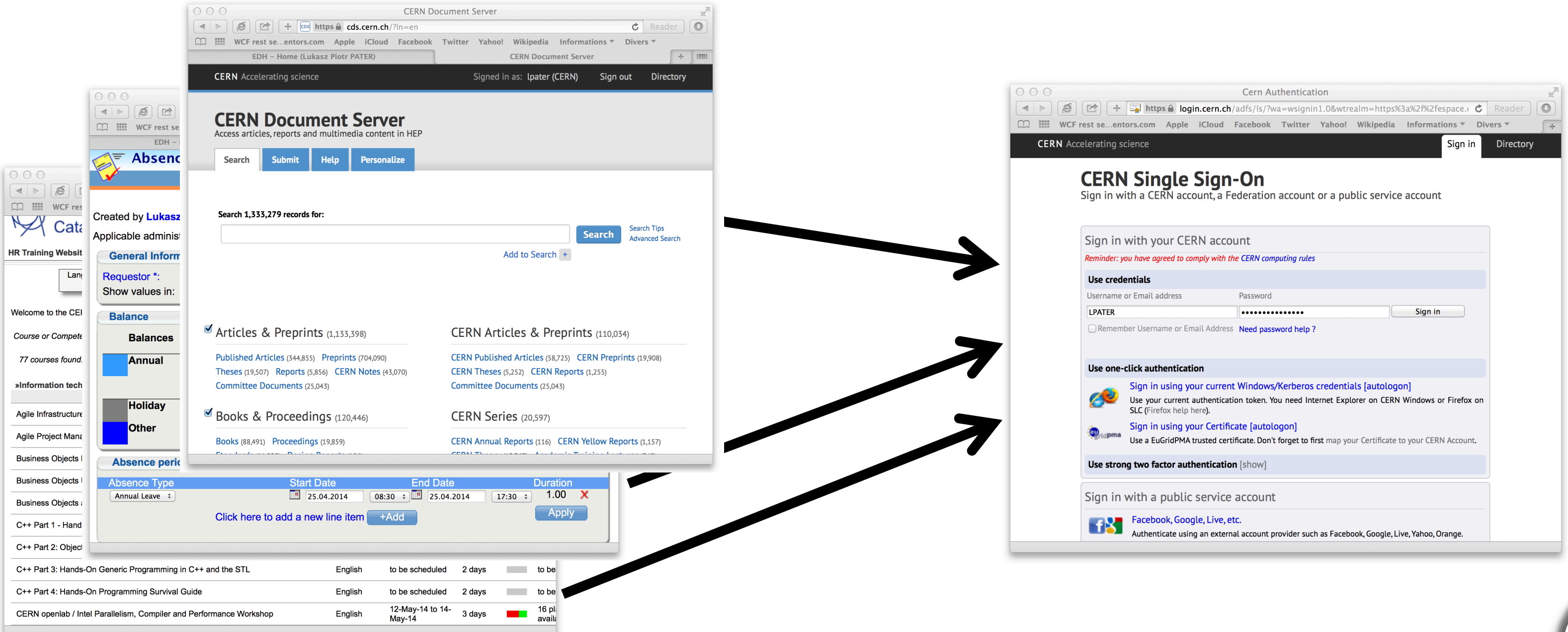






Single Sign On (SSO)

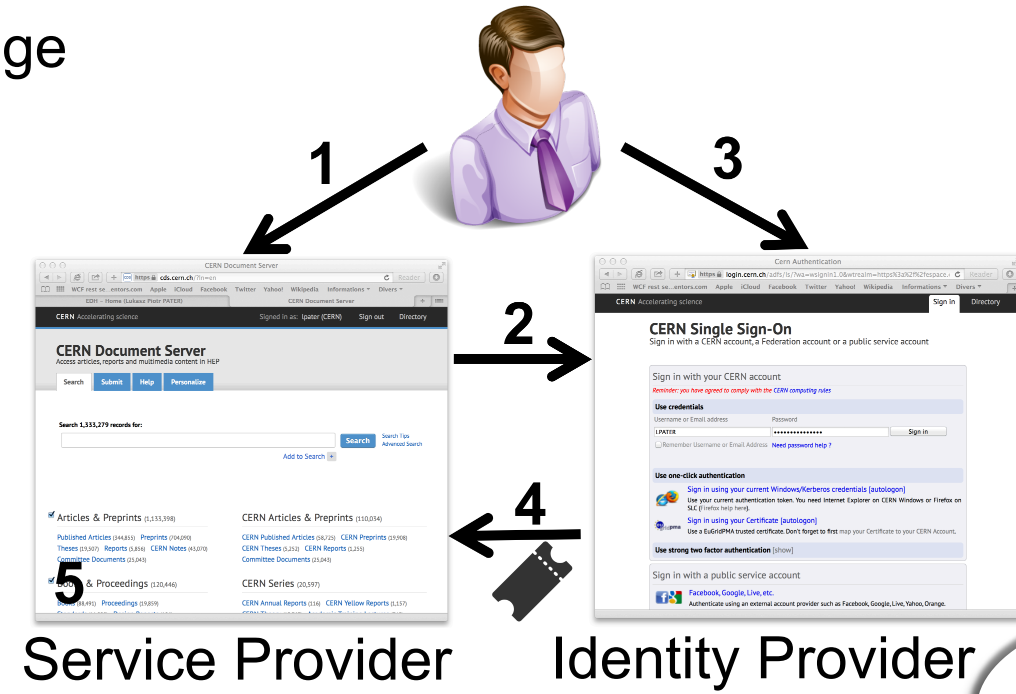
- **Multi sign on is troublesome**
  - Is it possible to just sign-on once to perform all the actions?
  - Single sign-on can be used to answer that question
- **SSO delegates the authentication to centralized service**



Single Sign On / How does it work?

- **Identity Provider (IP):** the system that asserts information about a subject (by issuing a ticket)
- **Service Provider (SP):** the system that relies on the information supplied to it by the Identity Provider (in the issued ticket)

1) User opens a web page
2) SP redirects to IP
3) User logs in
4) IP issues a ticket and redirects to SP
5) SP validates the ticket



Single Sign On (SAML) Ticket


```xml
<t:RequestSecurityTokenResponse>
    <t:Lifetime>
        <wsu:Created>2014-04-08T13:11</wsu:Created>
        <wsu:Expires>2014-04-08T23:11</wsu:Expires>
    </t:Lifetime>

    <saml:Attribute AttributeName="UPN">
        lukasz.piotr.pater@cern.ch
    </saml:Attribute>
    <saml:Attribute AttributeName="CommonName">
        lpater
    </saml:Attribute>

    <ds:Signature>
        <ds:SignatureValue>
            nChTR2/k8ltpJTAuAmnsBkt04....
        </ds:SignatureValue>
        <X509Certificate>
            MIIIEjCCBfqgAwIBAgIKLYgjv....
        </X509Certificate>
    </ds:Signature>
</t:RequestSecurityTokenResponse>
```


lukaszpater — bash — 65×29

```
lpatermac:~ lukaszpater$ openssl x509 -in sso.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            2d:88:23:bd:00:00:00:00:00:30
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: DC=ch, DC=cern, CN=CERN Certification Authority
        Validity
            Not Before: Nov  8 08:38:55 2013 GMT
            Not After : Jul 29 09:19:38 2023 GMT
        Subject: DC=ch, DC=cern, OU=computers, CN=login.cern.ch
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:a7:ab:75:0b:44:86:94:b7:5d:2f:63:3e:96:d7:
                    df:1a:2a:13:9c:34:f7:7b:3b:4e:c6:50:3d:6c:01:
                    ae:bc:1f:3f:46:3c:a6:f1:e9:fa:f8:44:fd:6e:83:
                    34:d8:2c:a8:47:95:69:9f:c8:84:95:75:56:71:c0:
                    04:c4:f2:10:12:ae:f4:0f:a2:cf:7b:ce:af:6f:eb:
                    58:86:e2:8f:78:b7:ee:20:23:7d:67:44:8c:43:87:
                    fe:e7:bb:e4:eb:50:a8:d7:fb:b0:ff:c9:0f:bd:79:
                    72:77:e0:22:f0:3b:27:fc:6d:21:a4:7e:bb:6c:12:
                    3d:54:e4:0a:7e:92:6a:72:57:12:26:e4:a5:50:1e:
                    4f:25:f7:ef:31:4a:6b:db:d0:74:79:1c:7c:2c:a1:
                    67:20:76:c3:20:b4:ae:ea:75:5b:14:b7:1a:76:d5:
                    88:8d:b6:2f:a0:aa:2b:5e:0d:c9:ea:ea:ed:93:3f:
                    b3:10:37:98:31:3b:c0:e9:91:8e:15:ff:d7:cb:97:
```

SSO / Advantages and Disadvantages

- **Advantages**
  - Reduces the chance of forgetting your password. By having your one-set master password
  - Very reach API
  - Less code to maintain for the web master
  - Single Sign Out
- **Disadvantages**
  - The SSO Identity Provider is highly-critical
  - Bad for a multi-user computer, especially if the user stays logged in all the time

Conclusions

- **Asymmetric encryption was introduced to complement the inherent problem of the need to share the same key in symmetric ciphers**
  - Symmetric encryption algorithms can be extremely fast
  - Public key algorithms are orders of magnitude less efficient than symmetric key encryption
  - Use public key cryptography just to exchange the key!
- **Public key certificates allow the secure communication between services and client applications**
- **Single Sign On enables to access multiple related, but independent software systems**

Questions & Answers