Public Key Infrastructure: Part 1

Gaurav S. Kasbekar
Dept. of Electrical Engineering
IIT Bombay

References

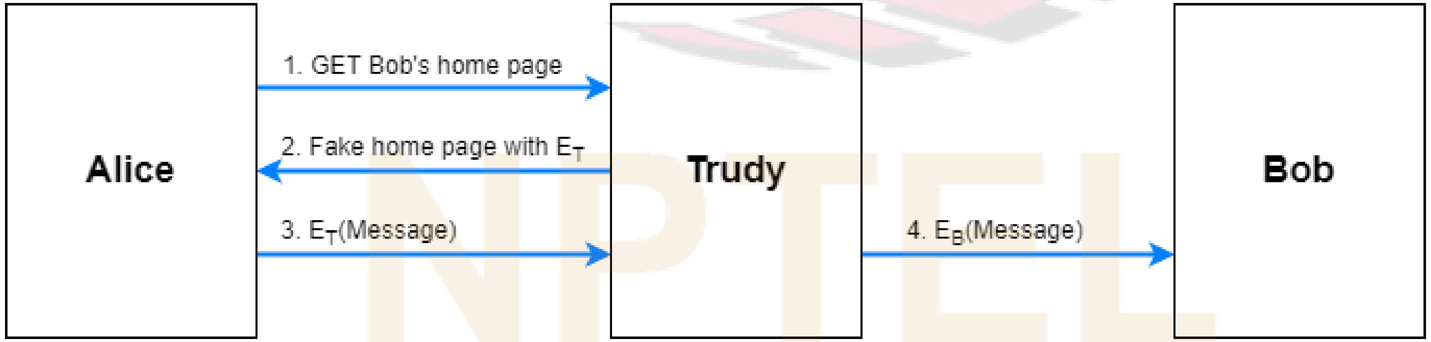
- J. Kurose, K. Ross, "*Computer Networking: A Top Down Approach*", Sixth Edition, Pearson Education, 2013
- A. Tanenbaum, D. Wetherall, "*Computer Networks*", Fifth Edition, Pearson Education, 2012.
- C. Kaufman, R. Perlman, M. Speciner, "*Network Security: Private Communication in a Public World*", Pearson Education, 2nd edition, 2002

Checking Whether a Public Key Belongs to a Specific User

- Suppose Alice and Bob do not know each other; Alice wants to send an encrypted message to Bob
- How does Alice get Bob's public key?
- *Obvious solution*:
  - ❑ Bob puts his public key on his website
  - ❑ Alice downloads public key from Bob's website

<u>*Shortcoming*</u>:

- Vulnerable to an attack such as the following:
  - ❑ Alice's request (GET message) to download Bob's webpage intercepted by Trudy
  - ❑ Trudy sends a fake webpage (*e.g.*, copy of Bob's webpage with Bob's public key replaced with Trudy's public key) to Alice
  - ❑ When Alice encrypts her message with this public key, Trudy decrypts and reads it, and sends modified version encrypted with Bob's true public key to Bob
- Want a mechanism using which Alice can check whether a public key that is supposed to be Bob's is indeed that of Bob
- **Public key infrastructure (PKI)**: components (e.g., organizations, policies, procedures) used to securely distribute public keys



Key Distribution Center (KDC)

- Suppose a KDC is available online for 24 hours everyday, for securely distributing public keys on demand
- Public key of KDC is known to everyone (e.g., preloaded in browsers); hence, everyone can securely communicate with KDC
- If Alice wants to know current public key of Bob, she connects to KDC and requests for it
- Disadvantages of this approach:
  - ❑ Not scalable: every user needs to connect to KDC
  - ❑ KDC can become a performance bottleneck, since all users connect to it whenever they want to obtain a public key
  - ❑ If KDC fails (e.g., crashes), users are not able to communicate securely
- Due to above reasons, people have developed a *different solution, which does not require any organization (e.g., KDC) to be online*
- Instead, there are organizations that issue "*certificates*" that public keys belong to specific persons, organizations, etc.

Certification Authority

- An organization that issues a certificate that a public key belongs to a specific person, organization, etc.
- *E.g.*: Suppose Bob wants a certificate for his public key
  - ❑ goes to a CA with his public key along with his passport, driver's license, etc.
  - ❑ CA verifies that Bob is who he claims to be, issues a certificate, say m, and adds a digital signature, $K_{CA}^{-}(H(m))$, where K_{CA}^{-} is the CA's private key
  - ❑ CA's public key is well-known
  - ❑ Bob may put $(m, K_{CA}^{-}(H(m)))$ on his website

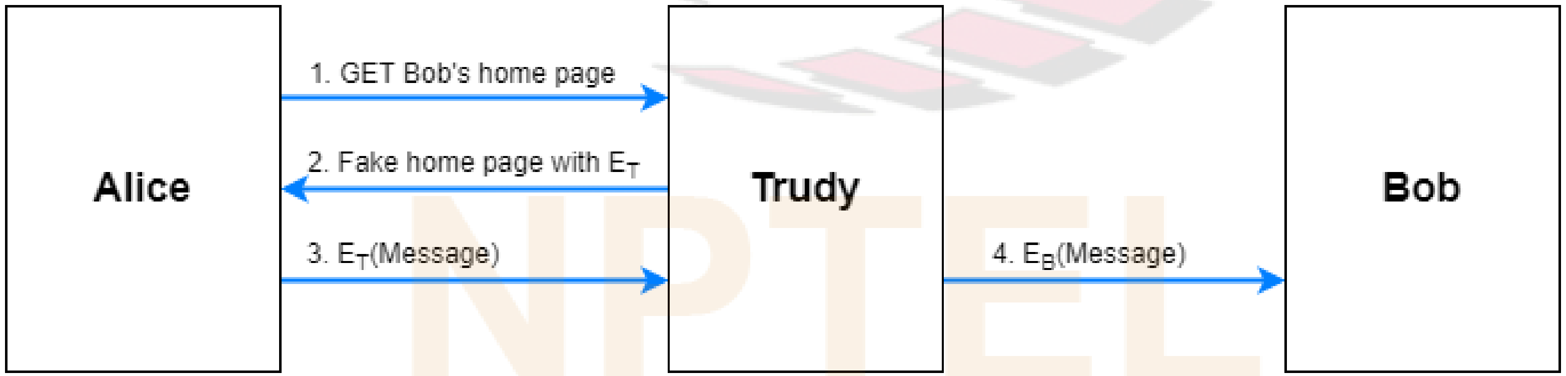
I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupernet.com

SHA-1 hash of the above certificate signed with the CA's private key

Checking Whether a Public Key Belongs to a Specific User (contd.)

- Recall above attack:
  - Alice's request (GET message) to download Bob's webpage intercepted by Trudy
  - Trudy sends a fake webpage (*e.g.*, copy of Bob's webpage with Bob's public key replaced with Trudy's public key) to Alice
  - When Alice encrypts her message with this public key, Trudy decrypts and reads it, and sends modified version encrypted with Bob's true public key to Bob
- Assuming that Alice knows CA's public key, does above mechanism prevent above attack?
- Yes, *e.g.*:

1) Suppose when Trudy intercepts Alice's GET request, she sends fake home page with her own certificate signed by CA
   - attack fails since Alice can read certificate and see that Trudy's (not Bob's) name is in it
2) Suppose Trudy modifies Bob's certificate by replacing his public key with her own to get m'; sends $(m', K_{CA}^{-}(H(m)))$ in place of $(m, K_{CA}^{-}(H(m)))$
   - attack fails since $K_{CA}^{+}(K_{CA}^{-}(H(m))) \neq H(m')$



Certification Authority (contd.)

- Recall: signed certificate is $(m, K_{CA}^{-}(H(m)))$
- User needs to trust CA's public key for above scheme to work
- Software such as browsers (*e.g.,* Firefox, Chrome) are preloaded with a set of trusted CA certificates:
  - CA certificate contains CA's name and public key
- CA certificates can be added/ removed by user
- *E.g. CAs*:
  - Symantec, Comodo, GoDaddy, GlobalSign

Certification Authority (contd.)

- Problematic if different certificates were in different formats, *e.g.*, browsers would need to understand all the formats
- ITU X.509 is a widely used standard that specifies format of certificates
- Table below shows some of the fields in a certificate

Field Name	Description
Version	Version number of X.509 specification
Serial number	CA-issued unique identifier for a certificate
Signature	Specifies the algorithm used by CA to sign this certificate
Issuer name	Identity of CA issuing this certificate, in distinguished name (DN)[RFC 4514] format
Validity period	Start and end of period of validity for certificate
Subject name	Identity of entity whose public key is associated with this certificate, in DN format
Subject public key	The subject's public key as well as indication of the public key algorithm (and algorithm parameters) to be used with this key

Example 1

- Certificate issued to www.freesoft.org by the CA Thawte Consulting
- To verify this certificate, we need certificate of the CA Thawte Consulting

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 7829 (0x1e95)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/Email=server-certs@thawte.com
        Validity
            Not Before: Jul  9 16:04:02 1998 GMT
            Not After : Jul  9 16:04:02 1999 GMT
        Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                 OU=FreeSoft, CN=www.freesoft.org/Email=baccala@freesoft.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                    33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                    66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                    70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                    16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                    c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                    8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                    d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                    e8:35:1c:9e:27:52:7e:41:8f
                Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
        92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
        ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
        d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
        0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
        5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
        8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
        68:9f
```

Example 2

- Certificate of the CA Thawte Consulting
- Note that this certificate is self-signed, *i.e.*, signed by issuer itself
- Such certificates are preloaded in web browsers

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/Email=server-certs@thawte.com
        Validity
            Not Before: Aug  1 00:00:00 1996 GMT
            Not After : Dec 31 23:59:59 2020 GMT
        Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                 OU=Certification Services Division,
                 CN=Thawte Server CA/Email=server-certs@thawte.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                    68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
                    85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
                    6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
                    6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
                    29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
                    6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
                    5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
                    3a:c2:b5:66:22:12:d6:87:0d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: md5WithRSAEncryption
        07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
        a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
        3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
        4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
        8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
        e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
        b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
        70:47
```