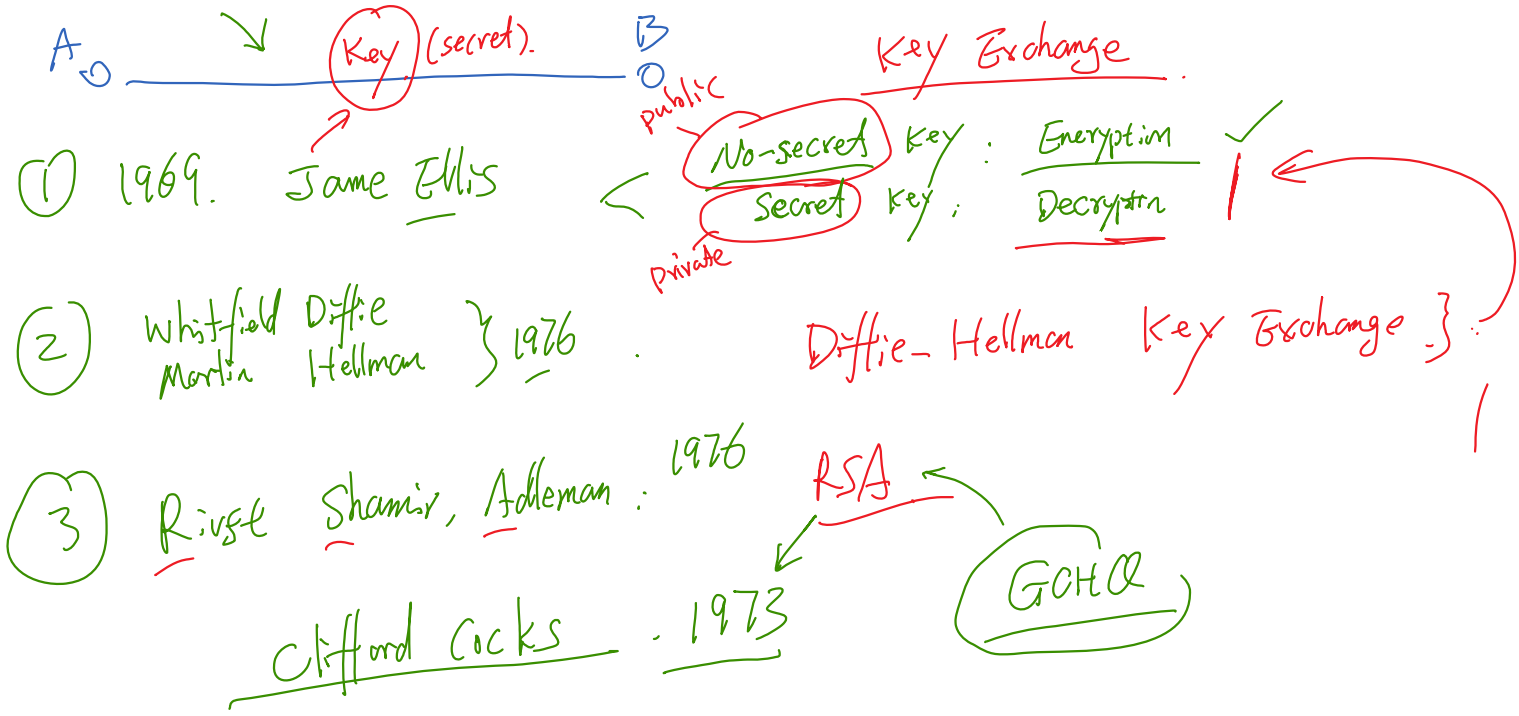
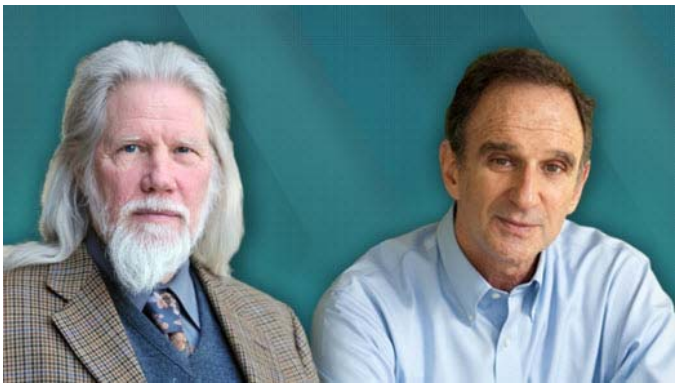
Public-Key Encryption

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Public-Key Cryptography: History and Concept



Inventors of Public-Key Encryption



Whitfield Diffie Martin Hellman

2015 Turing Award Winner

Clifford Cocks

Leonard Adleman Ron Rivest Adi Shamir

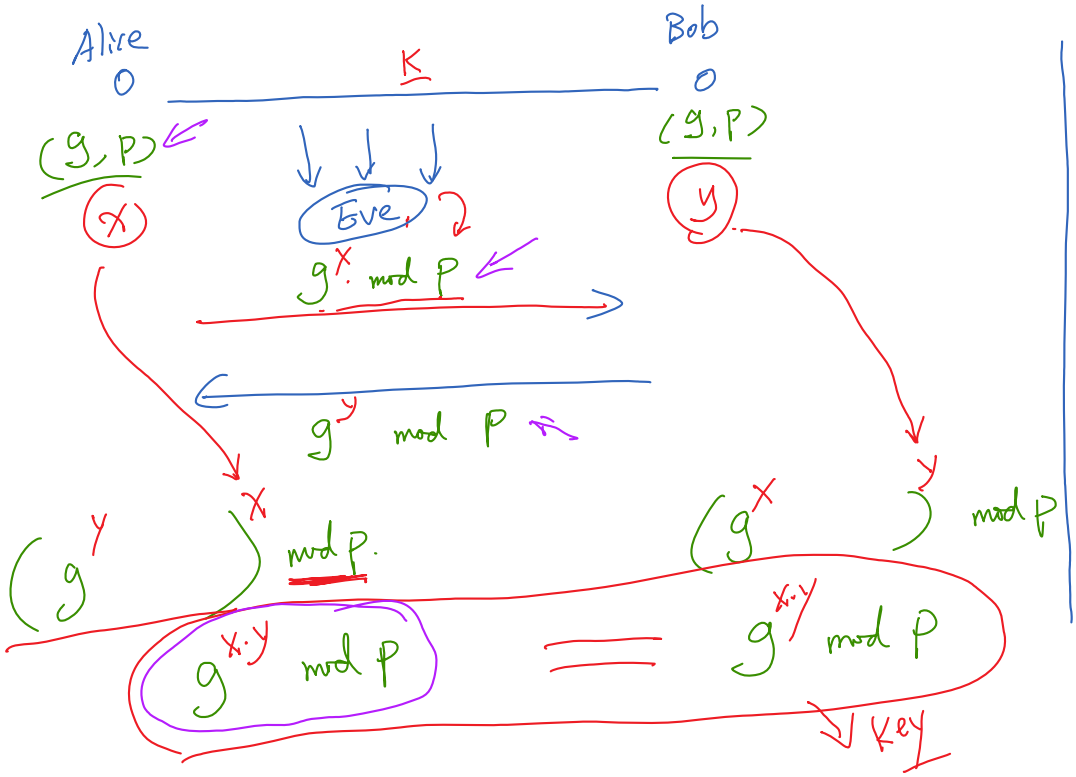
2002 Turing Award Winner

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Diffie-Hellman Key Exchange

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Diffie-Hellman Key Exchange



Discrete Logarithm

$g^x \bmod p = b$

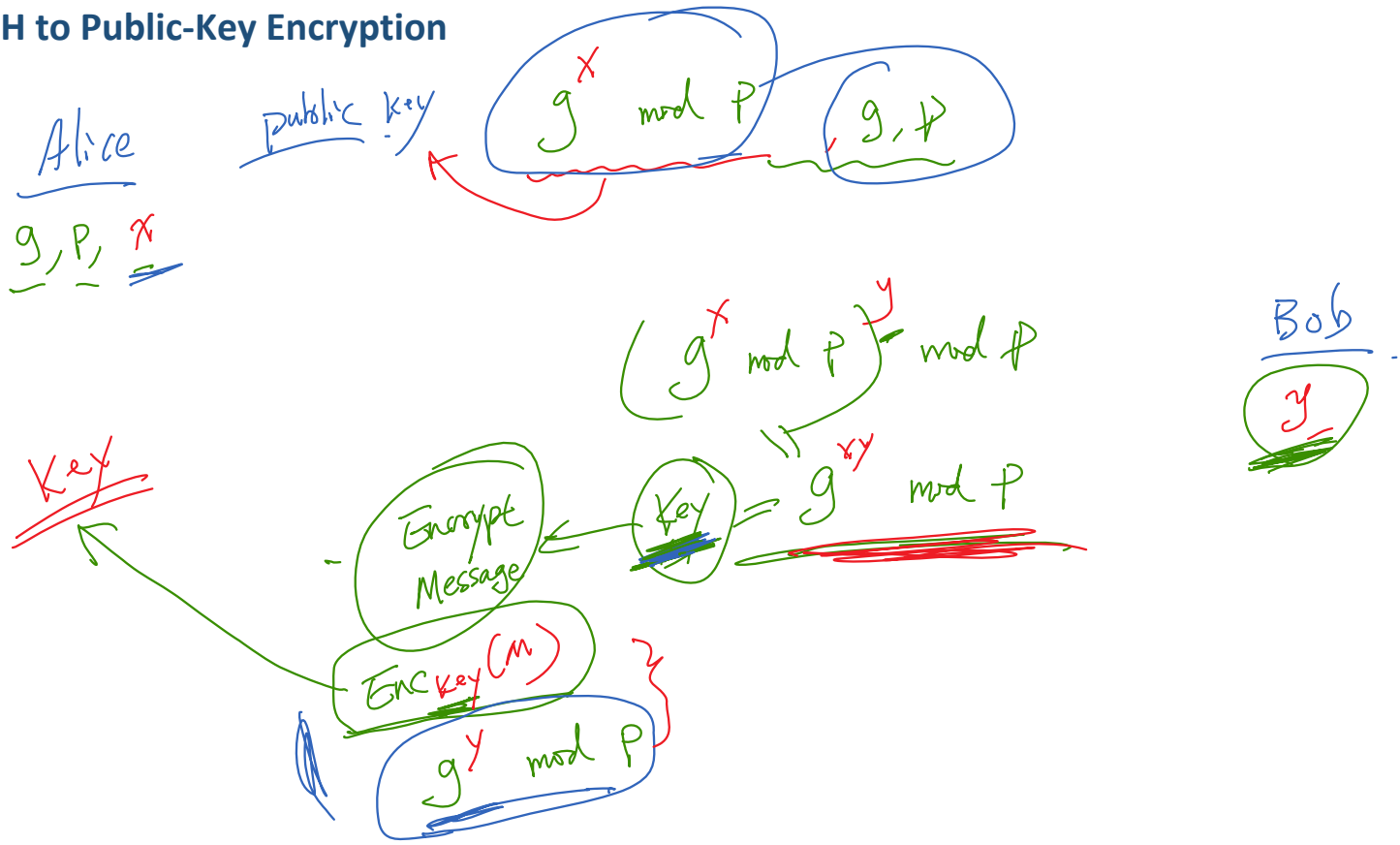
find x is hard

$a^x = b$

$\log_a a^x = \log_a b$

$x = \log_a b$

Turn DH to Public-Key Encryption

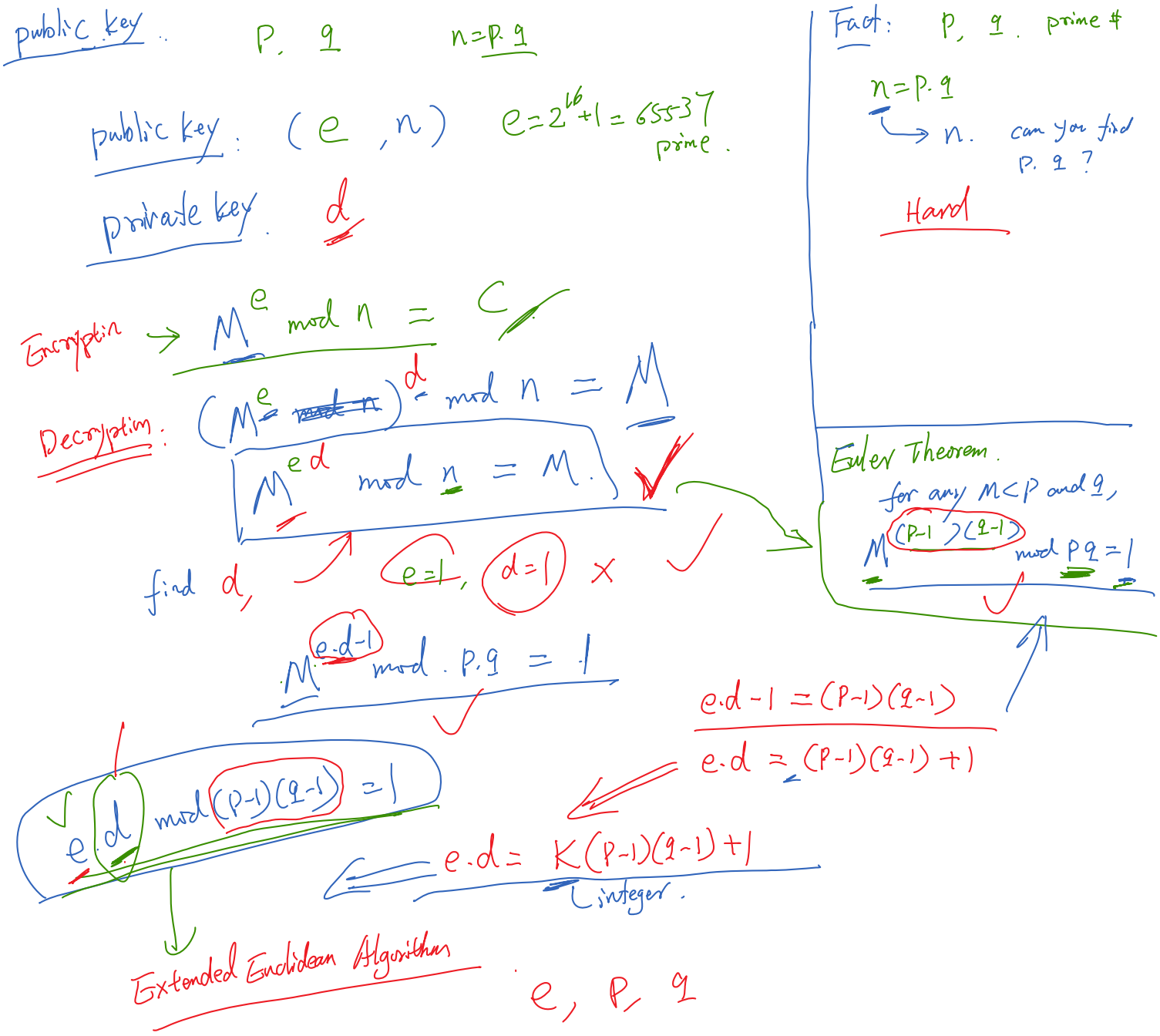


SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

RSA Public-Key Encryption Algorithm

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

RSA Algorithm



SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Exercise Related to RSA

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

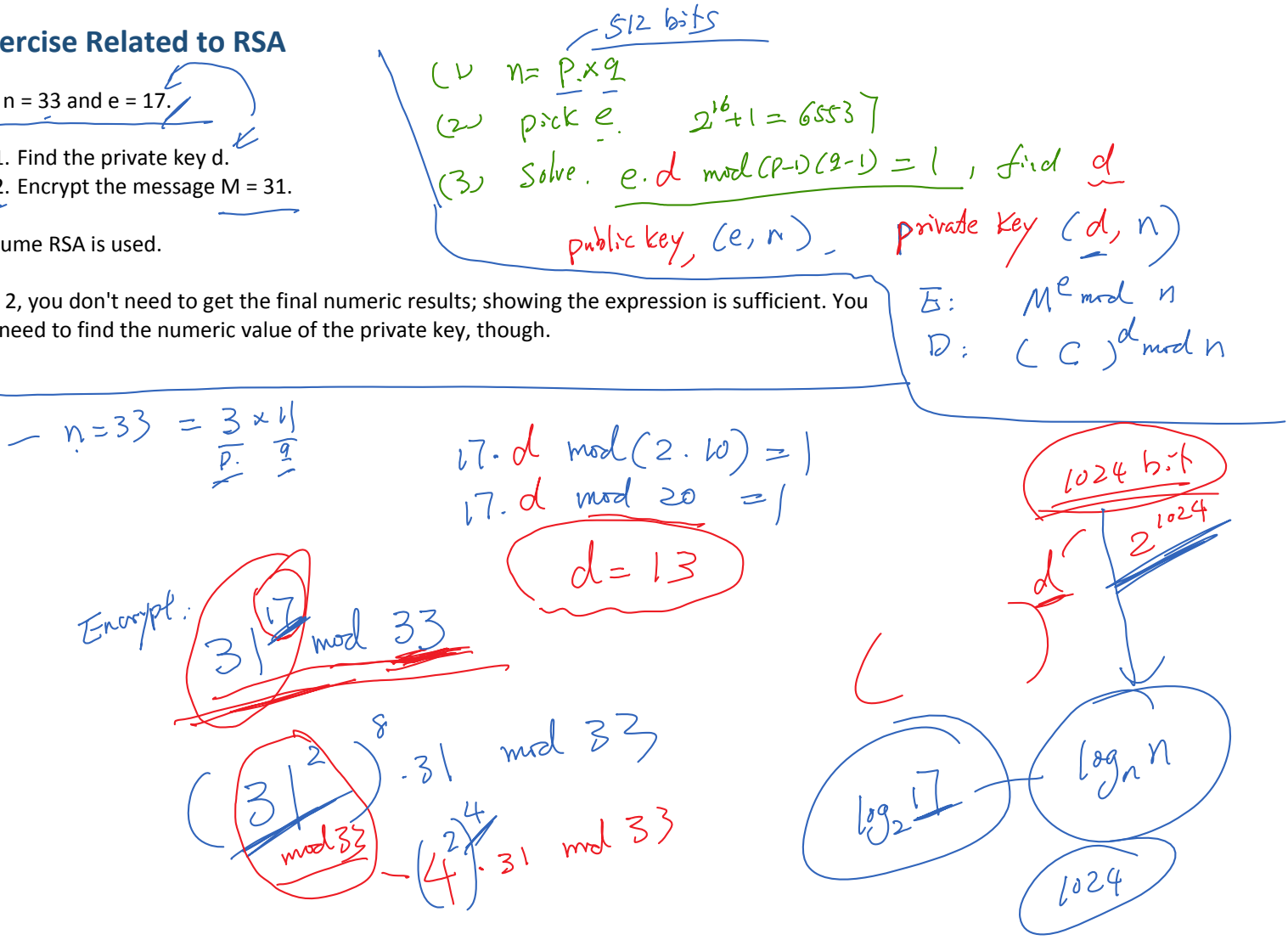
Exercise Related to RSA

Let n = 33 and e = 17.

1. Find the private key d.
2. Encrypt the message M = 31.

Assume RSA is used.

For 2, you don't need to get the final numeric results; showing the expression is sufficient. You do need to find the numeric value of the private key, though.

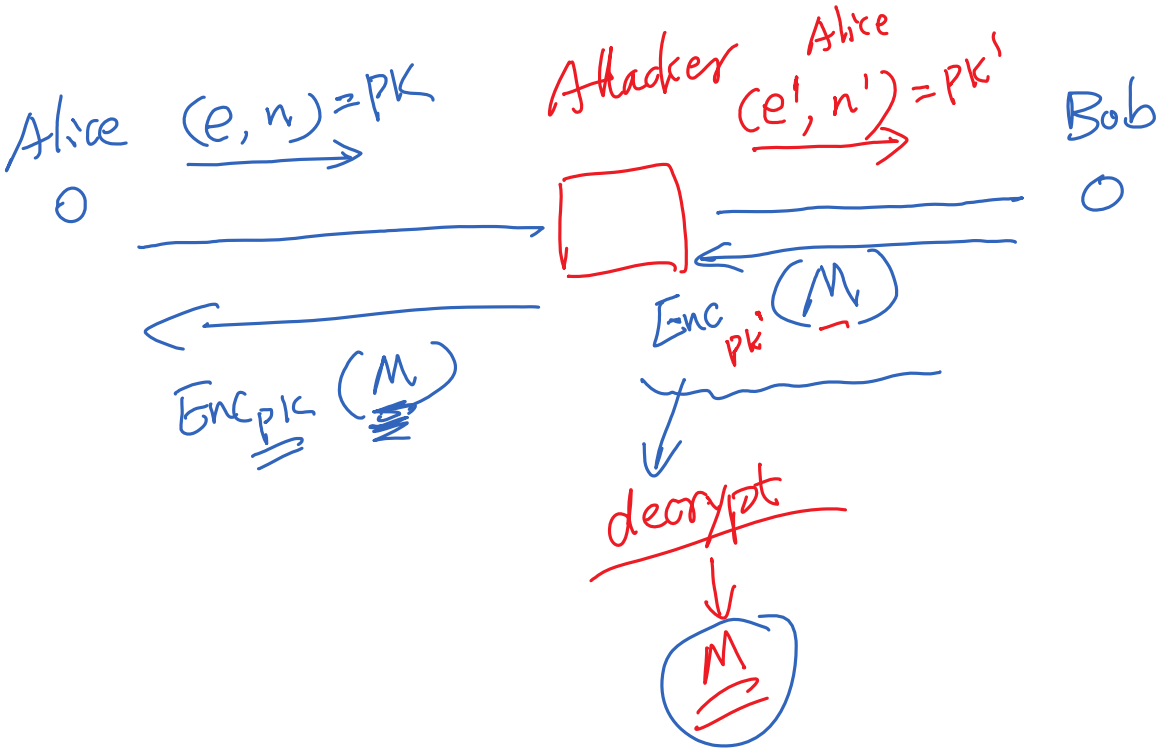


SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Man-in-the-Middle Attack

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

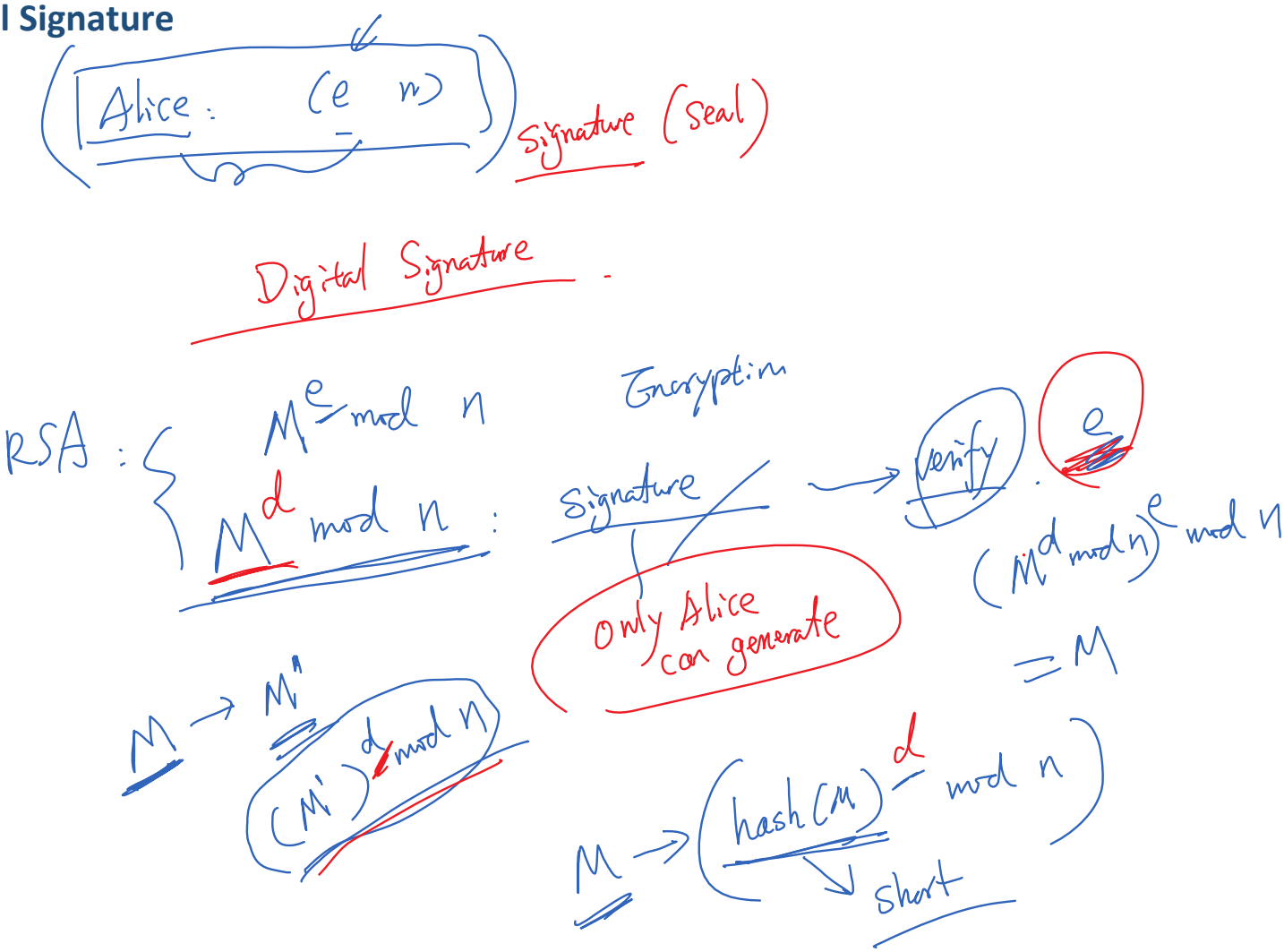
Man-in-the-Middle Attack



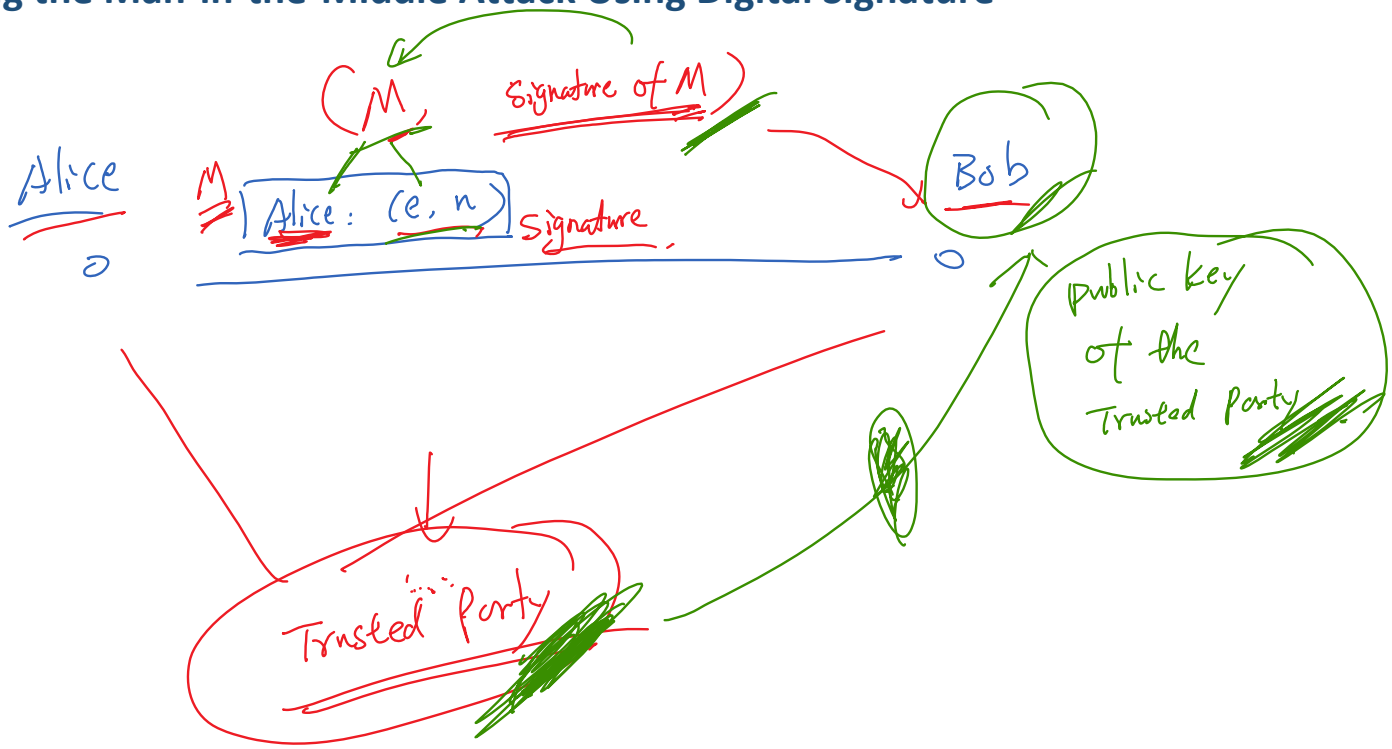
SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Digital Signature

Digital Signature



Defeating the Man-in-the-Middle Attack Using Digital Signature

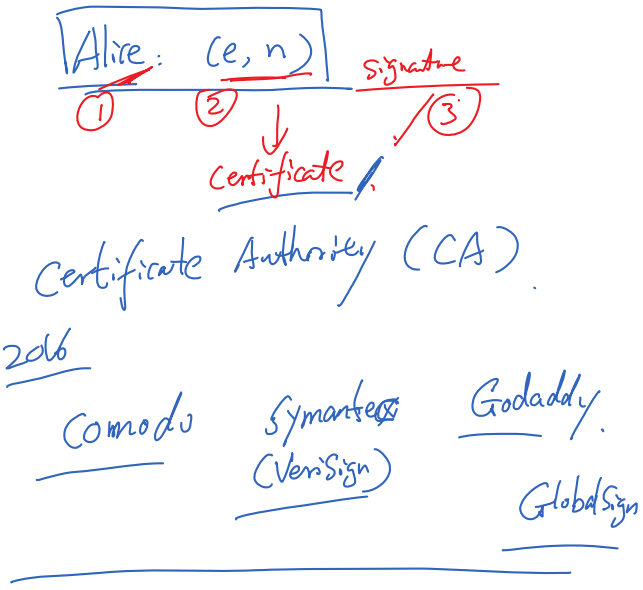


SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

X.509 Certificate

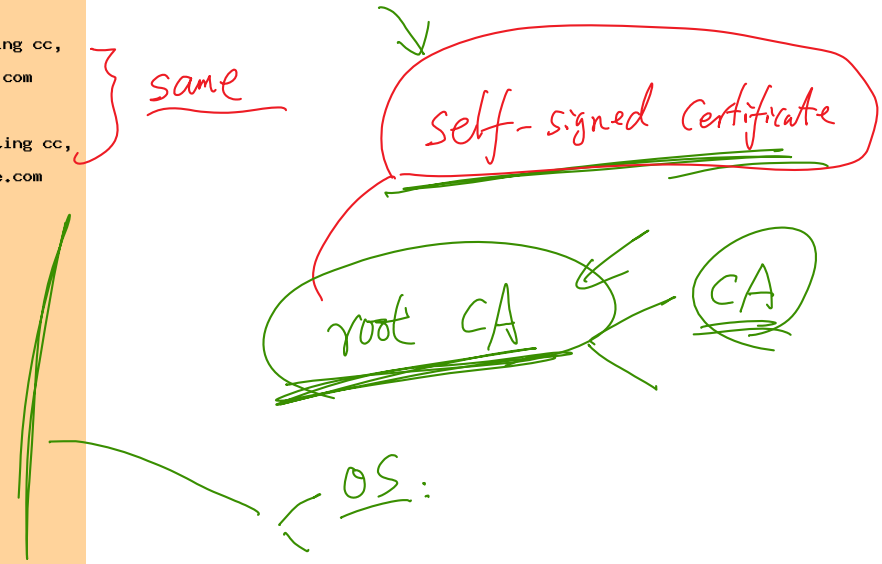
X.509 Certificate

```
A Sample X.509 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/emailAddress=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/emailAddress=...
        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
```



CA's X.509 Certificate

```
-----------------------------------------------------------------------
CA's 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/emailAddress=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/emailAddress=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
-----------------------------------------------------------------------
```

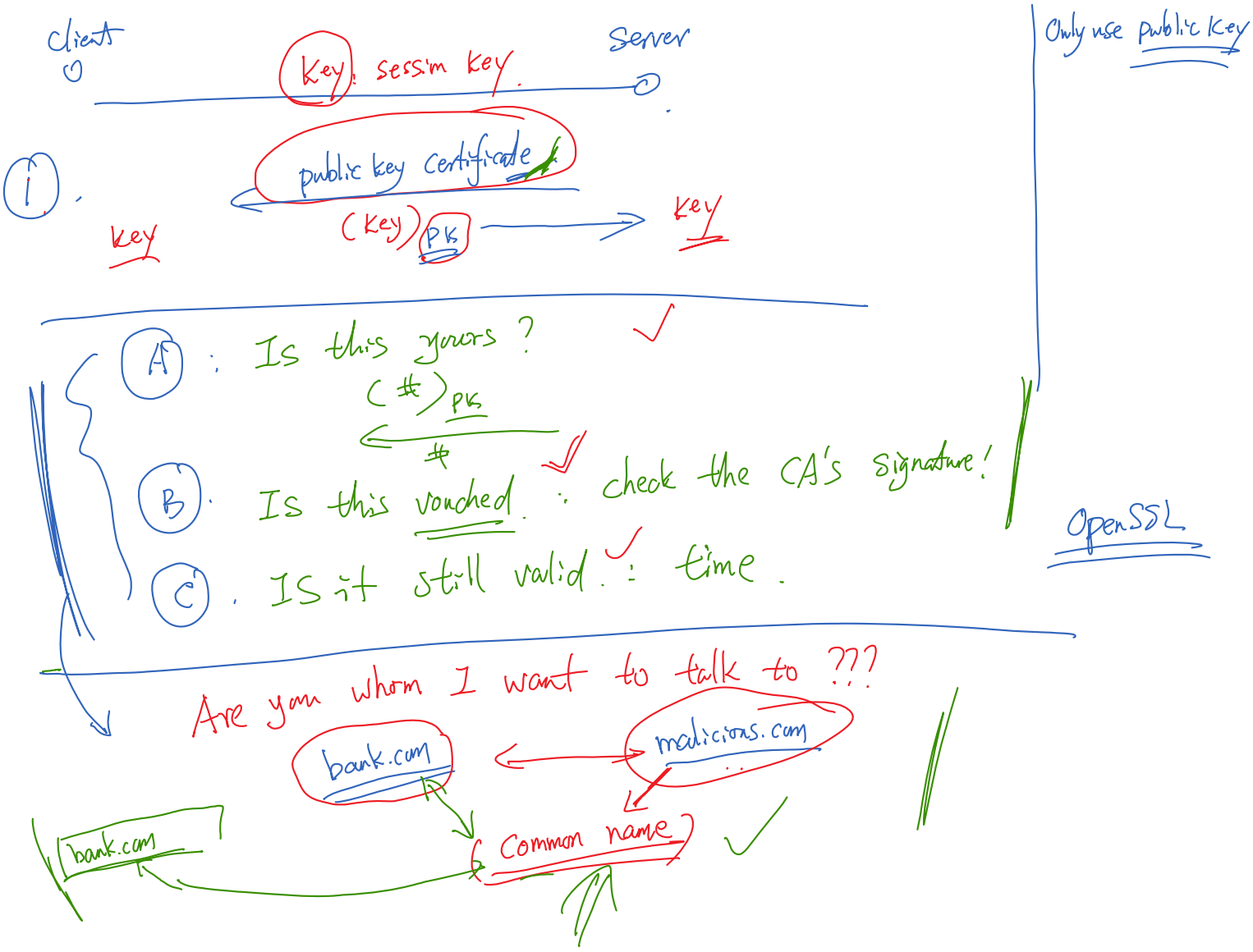


SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

The TLS/SSL Protocol

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

The TLS/SSL Protocol



Setup of SSL

❖ The client side

```
SSL_load_error_strings(); // readable error messages
SSL_library_init(); // initialize library

// Specify this is a client
meth = SSLv23_client_method();
ctx = SSL_CTX_new (meth);
if (!ctx) {
  ERR_print_errors_fp(stderr);
  exit(2);
}

// Will verify the server
SSL_CTX_set_verify(ctx,SSL_VERIFY_PEER,NULL);

// Set the location of the CA certificate
SSL_CTX_load_verify_locations(ctx,CACERT,NULL);
```

❖ The server side

```
SSL_load_error_strings(); // readable error messages
SSL_library_init(); // initialize library

// Specify this is a server
meth = SSLv23_server_method();
ctx = SSL_CTX_new (meth);
if (!ctx) {
  ERR_print_errors_fp(stderr);
  exit(2);
}

// Will not verify the client
SSL_CTX_set_verify(ctx,SSL_VERIFY_NONE,NULL);

// Set the location of the CA certificate
SSL_CTX_load_verify_locations(ctx,CACERT,NULL);

// Prepare the certificate (the client will request it)
if (SSL_CTX_use_certificate_file(ctx, CERTF, SSL_FILETYPE_PEM) <= 0) {
  ERR_print_errors_fp(stderr);
  exit(3);
}
if (SSL_CTX_use_PrivateKey_file(ctx, KEYF, SSL_FILETYPE_PEM) <= 0) {
  ERR_print_errors_fp(stderr);
  exit(4);
}
if (!SSL_CTX_check_private_key(ctx)) {
  fprintf(stderr,"Private key does not match the certificate public key\n");
  exit(5);
}
```

Establish SSL Connection

❖ The client side

```
/* ---------------------------------------------- */
/* Create a socket and connect to server using normal socket calls. */

sd = socket (AF_INET, SOCK_STREAM, 0);       CHK_ERR(sd, "socket");

memset (&sa, '\0', sizeof(sa));
sa.sin_family      = AF_INET;
sa.sin_addr.s_addr = inet_addr ("127.0.0.1");   /* Server IP */
sa.sin_port        = htons     (1111);          /* Server Port number */

err = connect(sd, (struct sockaddr*) &sa, sizeof(sa));
CHK_ERR(err, "connect");

/* ---------------------------------------------- */
/* Now we have TCP conncetion. Start SSL negotiation. */

ssl = SSL_new (ctx);                         CHK_NULL(ssl);
SSL_set_fd (ssl, sd);
err = SSL_connect (ssl);                     CHK_SSL(err);
```

❖ The server side

```
/* ---------------------------------------------- */
/* Prepare TCP socket for receiving connections */

listen_sd = socket (AF_INET, SOCK_STREAM, 0);   CHK_ERR(listen_sd, "socket");

memset (&sa_serv, '\0', sizeof(sa_serv));
sa_serv.sin_family      = AF_INET;
sa_serv.sin_addr.s_addr = INADDR_ANY;
sa_serv.sin_port        = htons (1111);          /* Server Port number */

err = bind(listen_sd, (struct sockaddr*) &sa_serv,
           sizeof (sa_serv));                   CHK_ERR(err, "bind");

/* Receive a TCP connection. */

err = listen (listen_sd, 5);                    CHK_ERR(err, "listen");

client_len = sizeof(sa_cli);
sd = accept (listen_sd, (struct sockaddr*) &sa_cli, &client_len);
CHK_ERR(sd, "accept");
close (listen_sd);

printf ("Connection from %lx, port %x\n",
        sa_cli.sin_addr.s_addr, sa_cli.sin_port);

/* ---------------------------------------------- */
/* TCP connection is ready. Do server side SSL. */

ssl = SSL_new (ctx);                           CHK_NULL(ssl);
SSL_set_fd (ssl, sd);
err = SSL_accept (ssl);                        CHK_SSL(err);
```

Verify Common Name

```c
/* Get server's certificate (note: beware of dynamic allocation) - opt */
server_cert = SSL_get_peer_certificate (ssl);       CHK_NULL(server_cert);
printf ("Server certificate:\n");

/* Get the subject from the certificate */
X509_NAME *subject =  X509_get_subject_name (server_cert);  CHK_NULL(subject);
str = X509_NAME_oneline(subject,0,0);   CHK_NULL(str);
printf ("\t subject: %s\n", str);
OPENSSL_free (str);

/* Get the Common Name field from the subject */
int nid_cn = OBJ_txt2nid("CN");
char common_name[256];
X509_NAME_get_text_by_NID(subject, nid_cn, common_name, 256);
printf ("\t CN: %s\n", common_name);
```

The following is a better way to verify common names:

```c
#include <openssl/x509.h>

int X509_check_host(X509 *, const char *name, size_t namelen,
                    unsigned int flags, char **peername);
int X509_check_email(X509 *, const char *address, size_t addresslen,
                     unsigned int flags);
int X509_check_ip(X509 *, const unsigned char *address, size_t addresslen,
                  unsigned int flags);
int X509_check_ip_asc(X509 *, const char *address, unsigned int flags);
```

DESCRIPTION

The certificate matching functions are used to check whether a certificate matches a given host name, email address, or IP address. The validity of the certificate and its trust level has to be checked by other means.

However,

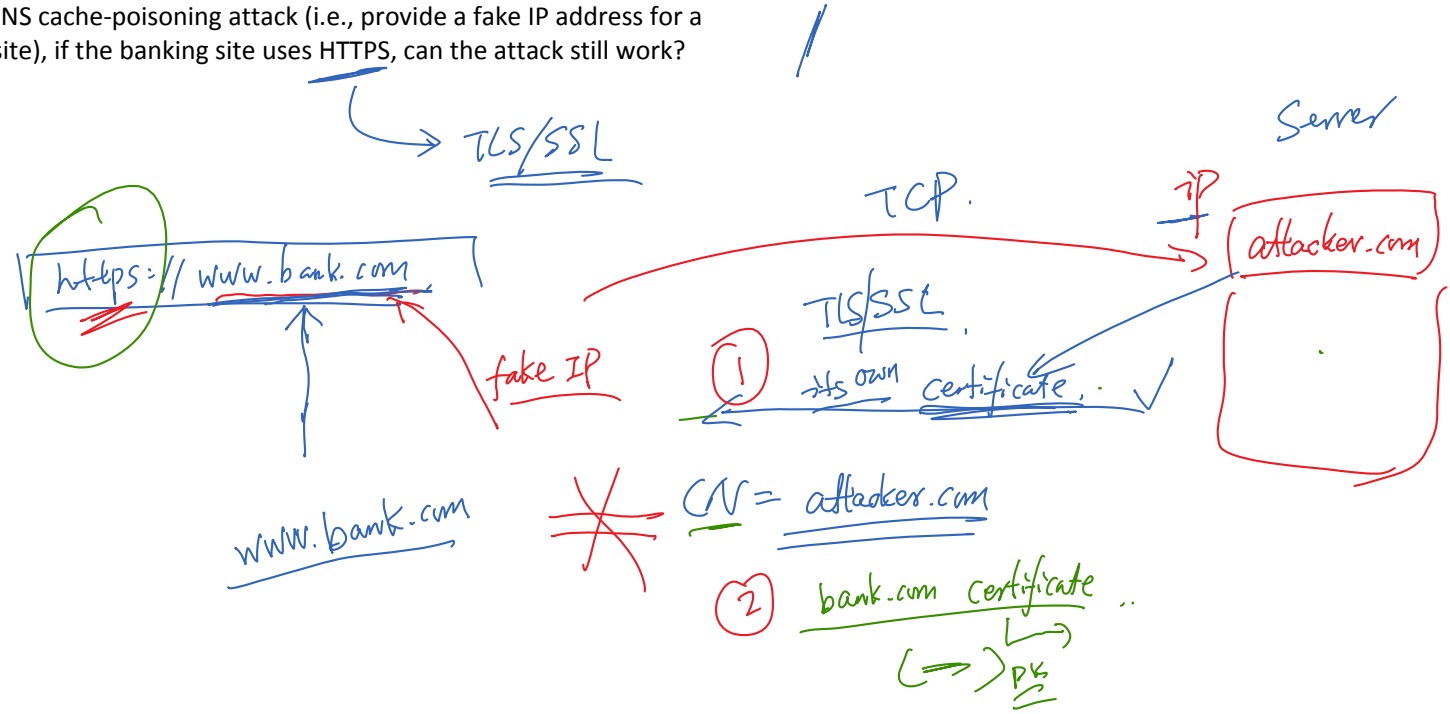
X509_VERIFY_PARAM_set1_host() sets the expected DNS hostname to **name** clearing any previously specified host name or names. If **name** is NULL, or empty the list of hostnames is cleared, and name checks are not performed on the peer certificate. If **name** is NUL-terminated, **namelen** may be zero, otherwise **namelen** must be set to the length of **name**. When a hostname is specified, certificate verification automatically invokes X509_check_host(3) with flags equal to the **flags** argument given to **X509_VERIFY_PARAM_set_hostflags()** (default zero). Applications are strongly advised to use this interface in preference to explicitly calling X509_check_host(3), hostname checks are out of scope with the DANE-EE(3) certificate usage, and the internal check will be suppressed as appropriate when DANE support is added to OpenSSL.

Data Exchange and Clean Up

```
/* ---------------------------------------------------- */
/* DATA EXCHANGE - Send a message and receive a reply. */

err = SSL_write (ssl, "Hello World!", strlen("Hello World!"));  CHK_SSL(err);

err = SSL_read (ssl, buf, sizeof(buf) - 1);                     CHK_SSL(err);
buf[err] = '\0';
printf ("Got %d chars:'%s'\n", err, buf);
SSL_shutdown (ssl);  /* send SSL/TLS close_notify */

/* Clean up. */

close (sd);
SSL_free (ssl);
SSL_CTX_free (ctx);
```

Question: DNS

For the DNS cache-poisoning attack (i.e., provide a fake IP address for a banking site), if the banking site uses HTTPS, can the attack still work?



SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

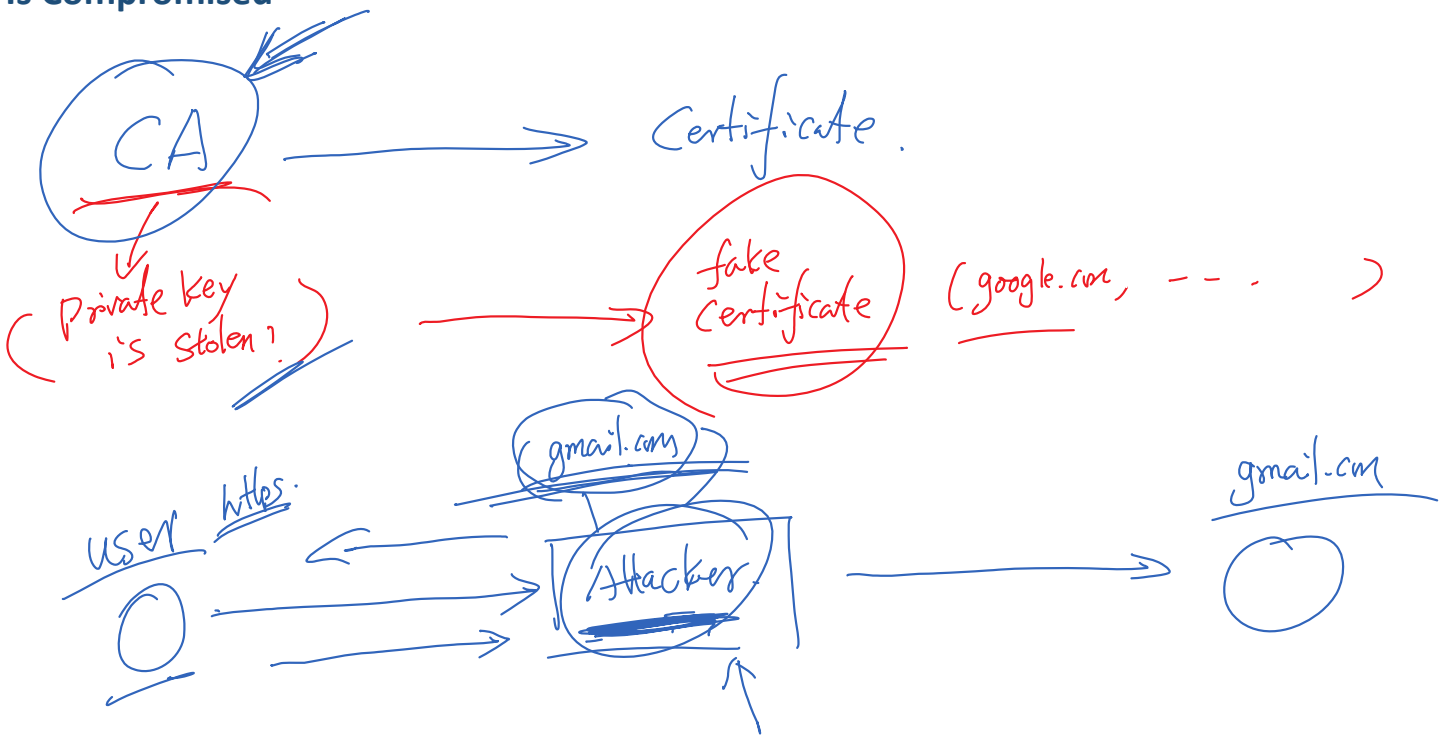
The Trust on CA

Root Certificate Authority (CA)

Survey result on April 2016:

- Comodo Group: 40.6%
- Symantec: 26.0% market share
- GoDaddy: 11.8%
- GlobalSign: 9.7%

If a CA Is Compromised



Case Studies

DigiNotar B.V. [3] was a Certificate Authority that provided digital certificate services. The digital certificates were used to secure Internet traffic, to issue (qualified) electronic signatures and to provide data encryption. DigiNotar also issued government accredited PKIoverheid certificates. During the months of June and July of 2011, the security of DigiNotar was breached and rogue certificates were issued. One of these certificates, a rogue Google certificate, was abused on a large scale in August of 2011 targeting primarily Iranian Internet users. At the end of August the intrusion became public knowledge and set into motion a chain of events that eventually led to the removal of all the Certificate Authorities that were hosted by DigiNotar from trust lists and ultimately the bankruptcy of the company.

"Using this [Gmail authentication] cookie, the hacker is able to log in directly to the Gmail mailbox of the victim and also read the stored emails," said Fox-IT. The hackers could also use the same credentials to log onto other Google services, including Google Docs and Google Latitude -- in the latter case, to identify the exact location of the victim -- and hijack Facebook and Twitter accounts.

Fox-IT said that approximately 300,000 IP addresses, each representing at least one computer and so at least one user, had accessed sites displaying a fake certificate for *google.com* between July 27 and Aug. 29. Nearly all -- Fox-IT said 99% -- of those IP addresses originated in Iran.

Investigators assumed that the *google.com* certificate was used primarily to spy on Iranians' Gmail accounts.

CNNIC Case Study

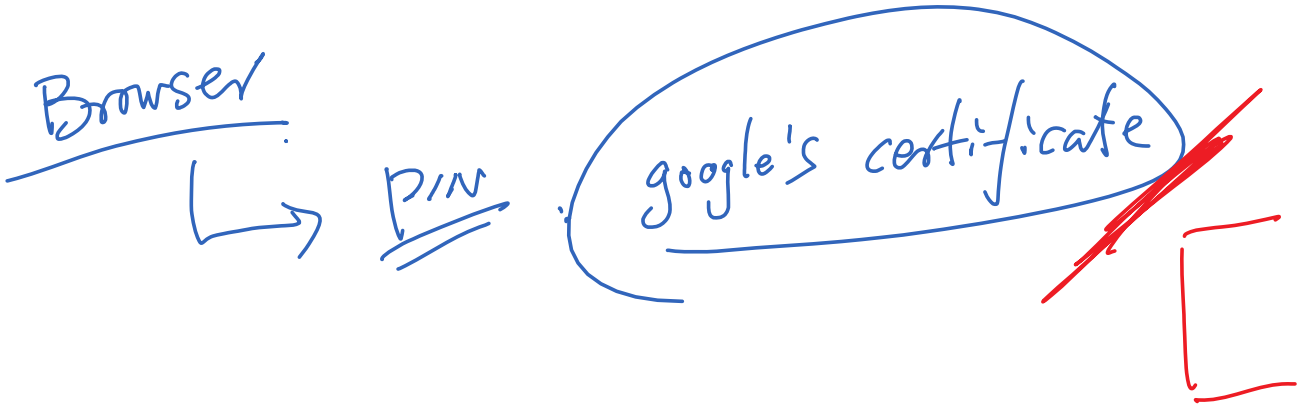
Google to drop China's CNNIC Root Certificate Authority after trust breach

OWEN WILLIAMS
12 days ago
Follow

Last month, a Chinese certificate authority issued valid security certificates for a number of domains, including Google's, without their permission, which resulted in a major trust breach in the crypto chain.

CNNIC had delegated its authority to Egyptian intermediary MCS Holdings to issue the certificates in question and the company installed it in a man-in-the-middle proxy internally.

Certificate Pinning



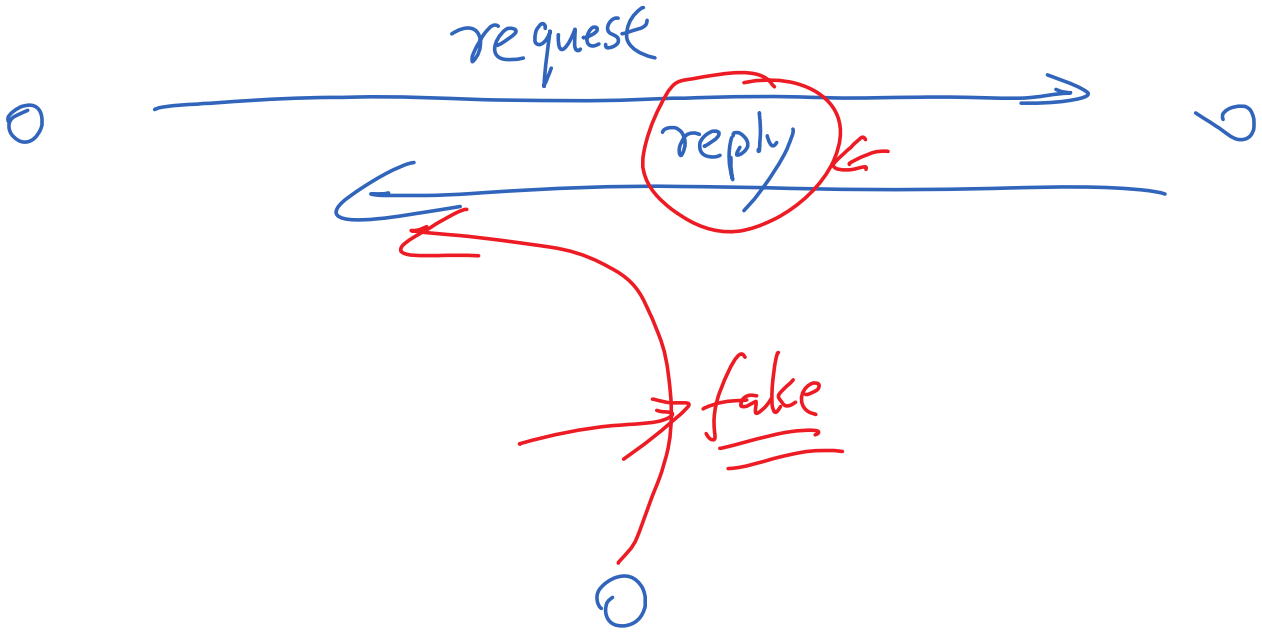
Certificate Revocation List (CRL)

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

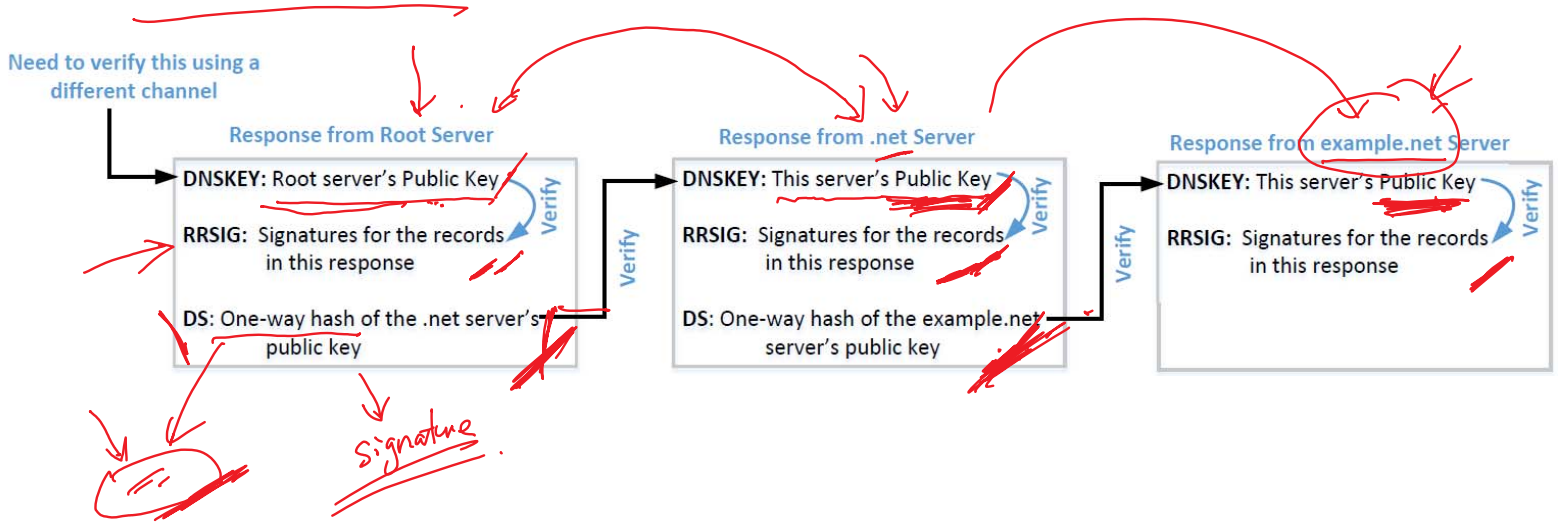
Application: DNSSEC

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

DNSSEC



DNSSEC Chain of Trust



DNSSEC Case Study

1. Query the root server.

```
seed@ubuntu:/var/www/CSRF$ dig +dnssec @a.root-servers.net gov
```

```
;; AUTHORITY SECTION:
gov.                    172800  IN      NS      a.gov-servers.net.
gov.                    172800  IN      NS      b.gov-servers.net.
gov.                    86400   IN      DS      7698 8 1 6F109B46A80CEA9613DC86D5A3E065520505A
AFE
gov.                    86400   IN      DS      7698 8 2 6BC949E638442EAD0BDAF0935763C8D003760
384FF15EBBD5CE86BB5 559561F0
gov.                    86400   IN      RRSIG   DS 8 1 86400 20150425170000 20150415160000 486
13 . LVifMvxuI531jBxMpWb4rTopQCB9yRZz4koI4W5CgtOuM4eQXy8qkWH5 +sGy04f8JZSY1da5Q3KrF4YHCNEAPzD1
rJm3htfZL9ei0lSzfO7PRCDv ipV9WY5X9vZI36bzmAfcHn1nh0bUed0xJ7VxzhzET4RFA8rrN4FII94w 3mw=
```

2. Query the .gov server.

```
seed@ubuntu:/var/www/CSRF$ dig +dnssec @b.gov-servers.net nsf.gov
```

```
;; AUTHORITY SECTION:
nsf.gov.                86400   IN      NS      swirl.nsf.gov.
nsf.gov.                86400   IN      NS      whirl.nsf.gov.
nsf.gov.                86400   IN      NS      cyclone.nsf.gov.
nsf.gov.                86400   IN      NS      twister.nsf.gov.
nsf.gov.                3600    IN      DS      5779 7 1 B74C98333849E241C2C57282C9FD24A7001AC
504
nsf.gov.                3600    IN      DS      5779 7 2 1F8F090FDFB13FD17AAF609F5358F5D218F2D
59F8264A34C185AEF51 2B1E5ED3
nsf.gov.                3600    IN      RRSIG   DS 8 2 3600 20150422221015 20150415221015 2956
7 gov. ax9BMy2zoSY3ASgZB4fq4F3Y1o2XY0FBK9p++HC/H0xUYCxukjb7opx/ pVRbmREErMx/f/kK+r1ezubfjZGadF
sS3OqjowzAnfG5e/rRt7m6HOkI w2C9SiEwZ1IAF0lqtfFu67gnjPnh3GIzDpLwjURvyhaa/eGvtFpsEEaV DS8=
```

3. Get the .gov server's public key.

```
seed@ubuntu:/var/www/CSRF$ dig DNSKEY @b.gov-servers.net gov
```

```
;; QUESTION SECTION:
;gov.                           IN      DNSKEY

;; ANSWER SECTION:
gov.                    86400   IN      DNSKEY  256 3 8 AQPGnrIGJp8OInQgK4MxDaVik9qhFDf2wLgdt2
bLvBQsE/rioqANibWv P45+XQ8gccgHciJN1WHmvCvX6j8YYZKH3vTKLbLsi0XyWqrFwzpbBtCB K6CM7KsswzFtgF98b+
dcNIoyGd22NfcJUTLoe/OmwUkWGiz6nu25WcNC NlliIQ==
gov.                    86400   IN      DNSKEY  256 3 8 AQPjAbHdR58IqX5S82ArSjCglRvWAiq1CBvDGh
ng5ph6dQRz4dj5AX3u qy1YFdfbTb8Jgzkpn6ld5vKozyKT9cskDFUqTxQ2AmN87o/KDYrEH3Mm HdGLwsDWVGVBes+8yP
eqNumqIcuu++UC9YK14UnLnHJk5sWN5LxiclcA dSRFHw==
gov.                    86400   IN      DNSKEY  257 3 8 AQO8daaz7B+yshOfL60rytKd9aOSujgponEw3f
wBMEC3/+e9XzHw2k+V KnbJTZ+QaVtpfUd1q9HKZIv/ck83Gl5TjYKE5jtUZ2kpEDZfVNGv6yx0 smtWAXv1nCJS9ohnyO
Td397eMojGDHqkEC+uojEScZheEkMxzgCZwDAs +/CSU7mSuHtCRZn19xlZUd5Gv7yDQ3mbOUwuy30oSk0z1Q5UUPpoihO
u gIZHFX6Jk7NLiW2wlqfq9qhV4zj7TiBiJY0mCc4zHN8/aq2VKDHp2Na7 mWzvKyTy+SYQkBQ/08LbPwj9YMc+uCzKL6s
U/ObHv17EFhD8aPDftTHZ vV9L+OZr
```

4. Query the nsf.gov server.

```
seed@ubuntu:/var/www/CSRF$ dig +dnssec @twister.nsf.gov nsf.gov
```

```
;; QUESTION SECTION:
;nsf.gov.                       IN      A

;; ANSWER SECTION:
nsf.gov.                600     IN      A       128.150.4.107
nsf.gov.                600     IN      RRSIG   A 7 2 600 20150422090044 20150415080044 23165
nsf.gov. A3p4/SkL/ecN/UAaVimmRHIPsfoA+IOVGmoPIaQqKkVNNGIRDZG49+LP 8mgeMZCyGfjjzqPPyuxvZH9zmRMp
aTtDIhX8AVtsf/4DVEzXXtSLDii6 Q6JS0RSQ60Q/GDQWxevvdVbSIYiqWf4Qg29ZFwps8iEMosVmDidI+6U5 rJM=
```

```
;; ADDITIONAL SECTION:
swirl.nsf.gov.          3600    IN      A       198.181.231.15
swirl.nsf.gov.          300     IN      AAAA    2620:10f:6012:1::15
whirl.nsf.gov.          3600    IN      A       198.181.231.16
whirl.nsf.gov.          300     IN      AAAA    2620:10f:6012:1::16
cyclone.nsf.gov.        3600    IN      A       204.14.134.227
cyclone.nsf.gov.        3600    IN      AAAA    2607:f478:80:1::53
twister.nsf.gov.        3600    IN      A       198.181.231.17
twister.nsf.gov.        300     IN      AAAA    2620:10f:6012:1::17
swirl.nsf.gov.          3600    IN      RRSIG   A 7 3 3600 20150422090044 20150415080044 23165
 nsf.gov. KyWv/Pj4o1kijCFeLBzPS9gF1zciCJ7cqhzRBN+ZlBl/e3OUrYv12dkv cGq2cz3DO1vGQhZyjyK3o2c9JYV
/cnuKjfM0fpvh/S9I0WZc4YSBQj6h BrTrPNVtZo2qiAwxIt7+gatpJ69EwpbxbED6scPlaOvnle37UiUpZFGY fsA=
swirl.nsf.gov.          300     IN      RRSIG   AAAA 7 3 300 20150422090044 20150415080044 231
65 nsf.gov. VoIP+8qHbSrKRX5ZCdSk/HBw4BXimudbFiOWr2rmvSuRMf3WKGsgZ0PV VGcrlHjwJmq1jpgdLkIJICRr0
r1e43p2dxkfrS5c55gX0zJ/Pi/X3Hdk tz5FQYO8CCULWC7cDOG6xC5VHLGqMQbSkEtq9B1fEp1Mk/3z0QuGYz9d QBY=
whirl.nsf.gov.          3600    IN      RRSIG   A 7 3 3600 20150422090044 20150415080044 23165
 nsf.gov. MMbTS0MEho8HKokD7QrVyzlExF+NoJq3aiwrsnvNgzls3LDcH/isgrGw fD1CMuYSLpDT16TwMdl1G+iewHu
BHVvdpSDV3mjbu8bX083p6Zw7DUlb R/SVCn6BVHPEKZZZHenmX9uRx0wK5qD5PfN7ROArnz45KxCUQqud8XX8 +Ys=
whirl.nsf.gov.          300     IN      RRSIG   AAAA 7 3 300 20150422090044 20150415080044 231
65 nsf.gov. PdF52xAPZLZUgdbpTVcYLfGWvpvrBqdhatj0uwDVmWj2Wlh6Tenly1Vd KEEZOp89Jpo6yGmhrN9vgg3VQ
ngud9o/2VhIPnsXou+ZsNaiwf5Q1ddf wrjqaQS73qtdf3AqFon5G7j8J7UvNFMinVQqpZs6mRC4C20BX+fJ59V2 7nI=
```

5. Get nsf.gov's public key.

```
seed@ubuntu:/var/www/CSRF$ dig DNSKEY @twister.nsf.gov nsf.gov
```

```
;; QUESTION SECTION:
;nsf.gov.                       IN      DNSKEY

;; ANSWER SECTION:
nsf.gov.                3600    IN      DNSKEY  256 3 7 AwEAAXBoA4fmTw+3vY2CMsVgOFmiP8mYb80m+i
Y5A3vcAxdGRQY68VUT lrKyyi6GC/4JI2TOwuTvmFesUhNbBMja/qonJ1yyxiocDqYhUCJgmcx3 9oLBgGQrhGoSBvPNA/
i+Y8+6xlv6XzK5HC+H1NUlc600CIzNo4sSZG4c aDjAFsg9
nsf.gov.                3600    IN      DNSKEY  256 3 7 AwEAAXV2Ejokqi8BFsMQYEW/4D5r7srevzdBB+
nzNVvW2ViYGBjyF8Ol dLezEL7GSlsmHhh4BsrYzR60YeQw6sn1w+bPm+FAxUKWTy5rnMy/Dogc OCcqI4w6y7j7PxtI4F
S1DpvlzbQK/Dfr4MBMvfTCPSnXQarhIeIz7Efk OluwI62f
nsf.gov.                3600    IN      DNSKEY  257 3 7 AwEAAW5iDWXVILLyVfOgjl5GndsbYhk4S6Ojpr
KDQIY8XeW0hFuh0Cxd I6R8FnddYiNkz6qCrgGu6Vmx+vAbiLwL1nQLbhWHb/g14BmoF3eTauSG WKequMSgX+MNZ1vhpP
4LeodTMTBFVKXAOlJ3hwrEwb51R2vAeogmc5n7 wxf7PJOGnXBzdOjHfqqek651e89iaQ8CA1pZxZtVsN19voULEXGzSlH
e 1eb2kSl+nqVk6dDC0h9OzjX2FRs7BAiu8Ezih5lrl/lyd0ZO17jZRYEQ nlVdVqkjmFwm6wt7wRhVBWFKUaVJxPlqu4n
FeE92oa2BtHMXlKE/Djyc r2VF4o+JUQ8=
```

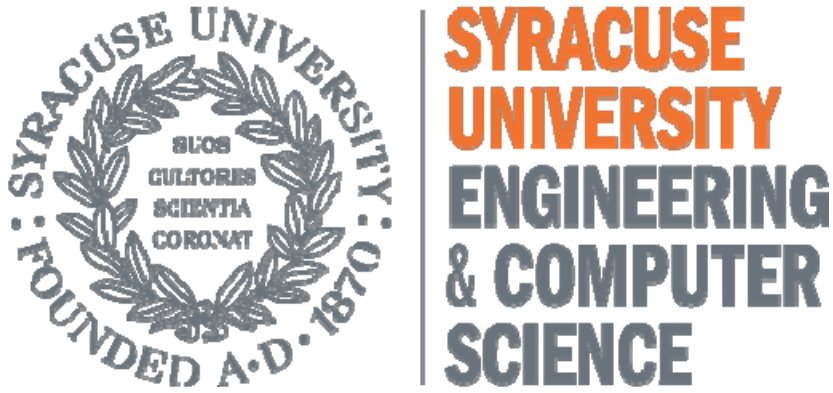
SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Summary


SYRACUSE UNIVERSITY
SUOS
CULTORES
SCIENTIA
CORONAT
FOUNDED A·D·1870
SYRACUSE
UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Summary

- Public key encryption concept
- Diffie-Hellman key exchange protocol
- RSA algorithm
- Man-in-the-middle attack
- Digital signature, X.509 certificate, and CA
- TLS/SSL protocol
- Case studies on CAs
- DNSSEC


SYRACUSE UNIVERSITY
SUOS
CULTORES
SCIENTIA
CORONAT
FOUNDED A·D· 1870
SYRACUSE
UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE