Encryption

Baseline Course

Ahsan Ayub

Ph.D. Student and Graduate Research Assistant

Tennessee Tech University

Cybersecurity Education, Research and Outreach Center (CEROC)

mayub42@students.tntech.edu

November 21, 2019

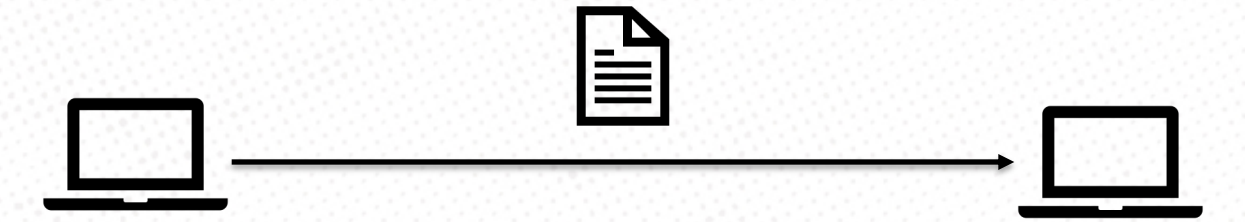
CEROC
Cybersecurity Education, Research and Outreach Center

Overview

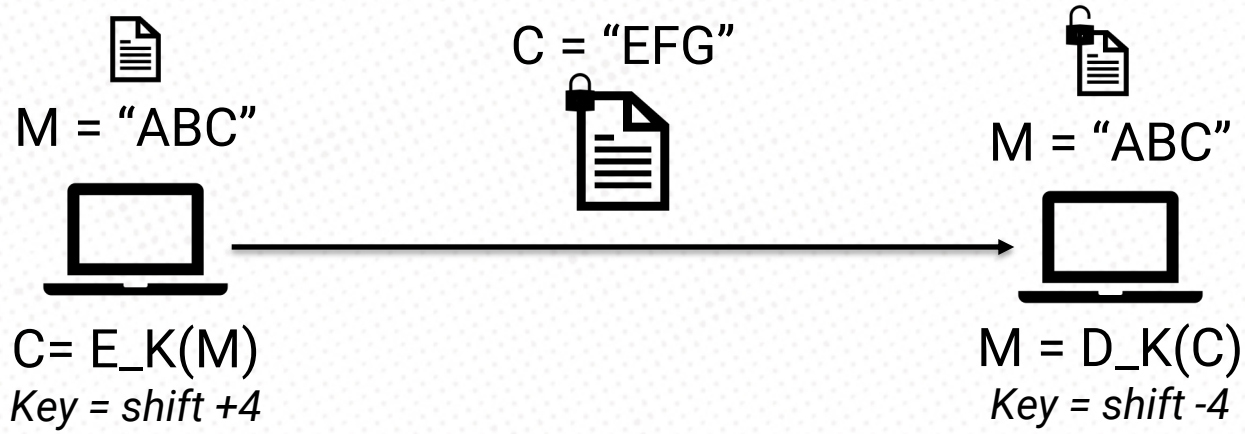
- Cryptography
  - Algorithms
  - Key Distribution
  - Attacks
  - Certifications
- Demonstration
- Adversarial Examples

Cryptography

Cryptography



Cryptography (Cont.)



Security of a Cryptosystem

- Crypto algorithms can be computationally secure
  - The cost of breaking the encryption exceeds the value of the encrypted information
  - The time required to break the cipher exceeds the useful lifetime of the information
- Cost and time are difficult to estimate
- Exhaustive key space search (brute-force)
  - If trying all possible keys requires ten billion years with the most powerful known computer, the algorithm can be considered secure

Cryptanalysis

- Attempts to discover the key or the plaintext of an encrypted message
- Types of attacks
  - Ciphertext only
    - Given: C1 = E_K(P1), C2 = E_K(P2), …. , Ci = E_K(Pi)
    - Obtain: either P1, P2, … , Pi or K
  - Known plaintext
    - Given: P1, C1 = E_K(P1); P2, C2 = E_K(P2), …. , Pi, Ci = E_K(Pi)
    - Obtain: either K or an algorithm to obtain Pi+1, from Ci+1 = E_K(Pi+1)
  - Chosen plaintext
    - Given: P1, C1 = E_K(P1); P2, C2 = E_K(P2), …. , Pi, Ci = E_K(Pi) where the attacker choose P1, P2, … , Pi
    - Obtain: either K or an algorithm to obtain Pi+1, from Ci+1 = E_K(Pi+1)

Secure Communication

- What we care about …
  - Confidentiality
  - Authentication
  - Integrity
  - Non-repudiation
- Threat Model
  - Can intercept
  - Can repeat
  - Can Modify
  - CANNOT break crypto (assumption)

Crypto Algorithms

- Symmetric algorithms (conventional, shared secret key)
  - The encryption / decryption keys are the same (K)
  - Two types of ciphers; examples –
    - DES (Block cipher and *broken*)
    - AES (Block cipher and *preferred*)
    - RC4 (Stream cipher and *broken*)
    - ChaCha (Stream cipher)

Crypto Algorithms (Cont.)

- An example of symmetric key encryption based communication –
  - Alice and Bob agree on a cryptosystem (e.g. AES)
  - Alice and Bob agree on a (secret) key K
  - Alice encrypts a message using K and she sends it to Bob

    C = E_K(M)
  - Bob decrypts the message using K M = D_K(C)

Crypto Algorithms (Cont.)

- Advantages: Symmetric Key Encryption
  - Confidentiality: Eve cannot access M without knowing K
  - Authentication (Weak): only who knows K can participate in the communication
  - Integrity (weak): if the message C is modified, it will decrypt to gibberish
- Disadvantages: Symmetric Key Encryption
  - Key distribution is critical
  - If key is compromised. Eve can impersonate both Alice and Bob
  - The number of keys increases with the number of parties
  - Cannot be used to prove that the message was sent specifically by one of the involved parties

Crypto Algorithms (Cont.)

- Key Distribution Center (KDC)
  - Alice shares a secret key Ka with the KDC
  - Bob shares a (different) secret key Kb with the same KDC
  - Alice tell the KDC that she wants to talk to Bob
  - The KDC generates a temporary secret key (session key) Kt
  - The KDC sends E_Ka(Kt) to Alice and E_Kb(Kt) to Bob
  - Alice and Bob communicate using Kt

Crypto Algorithms (Cont.)

- Advantages of communication with a KDC -
  - Limited number of keys
  - Kt validation is limited to a session
- Disadvantages of communication with a KDC -
  - SPOF
  - You have to TRUST KDC
  - Requires consonant availability of the KDC

Crypto Algorithms (Cont.)

- Public-key algorithms (Asymmetric)
  - There are two different keys (K_1, K_2)
  - It is not possible (or computationally feasible) to calculate K_1 given K_2 or vice versa
  - Encryption with one key, decryption with the other key
  - Examples –
    - Diffie-Hellman
    - RSA
    - Elliptic Curve

Crypto Algorithms (Cont.)

- An example of asymmetric key encryption based communication –
  - Alice and Bon agree on a public-key crypto system (e.g. RSA)
  - Bob sends his public key PKb to Alice
  - Alice encrypts a message with Bob's public key and sends it
  - Bob decrypts the message using his private key SKb

Crypto Algorithms (Cont.)

- Advantages: Asymmetric Key Encryption
  - Key distribution: Public keys can be sent using in-band channels
  - Confidentiality: Eve must know Bob's private key to decrypt the message
  - Integrity (Weak): Any modification of the message would be revealed when decrypting
- Disadvantages: Asymmetric Key Encryption
  - No authentication: anybody can send a message to Bob pretending to be Alice
  - Alice can deny having sent the message

Crypto Algorithms (Cont.)

- One-way Hash functions
  - Take a variable-length input and produce a fixed-length output
  - Transform the data irreversibly, and in a way that is as similar as possible to a random function
    - Unfeasible to change the message without changing the hash
    - Unfeasible to pre-compute the whole hash space (per-image resistance)
    - Unfeasible to find a (previously unknown) message that generates a specific given hash (second pre-image resistance)
    - Unfeasible to find two messages that generate the same hash (collision resistance)
  - Transformation can be parameterized using secret key (K)
  - Examples: MD5 (Broken), Sha-256 (Preferred), etc.

Crypto Algorithms (Cont.)

- Random Number Generators
  - Random number generators (use physical phenomena)
  - Pseudorandom number generators (use initial seed and an algorithm)

Case Study: Asymmetric Encryption

- Alice and Bon agree on a public-key crypto system
- Alice and Bob exchange their public keys (PKa, PKb)
- Alice encrypts a message with her private key SKa and then with Bob's public key PKb
- Alice sends the message
- Bob decrypts the message with his private key and then with Alice's public key

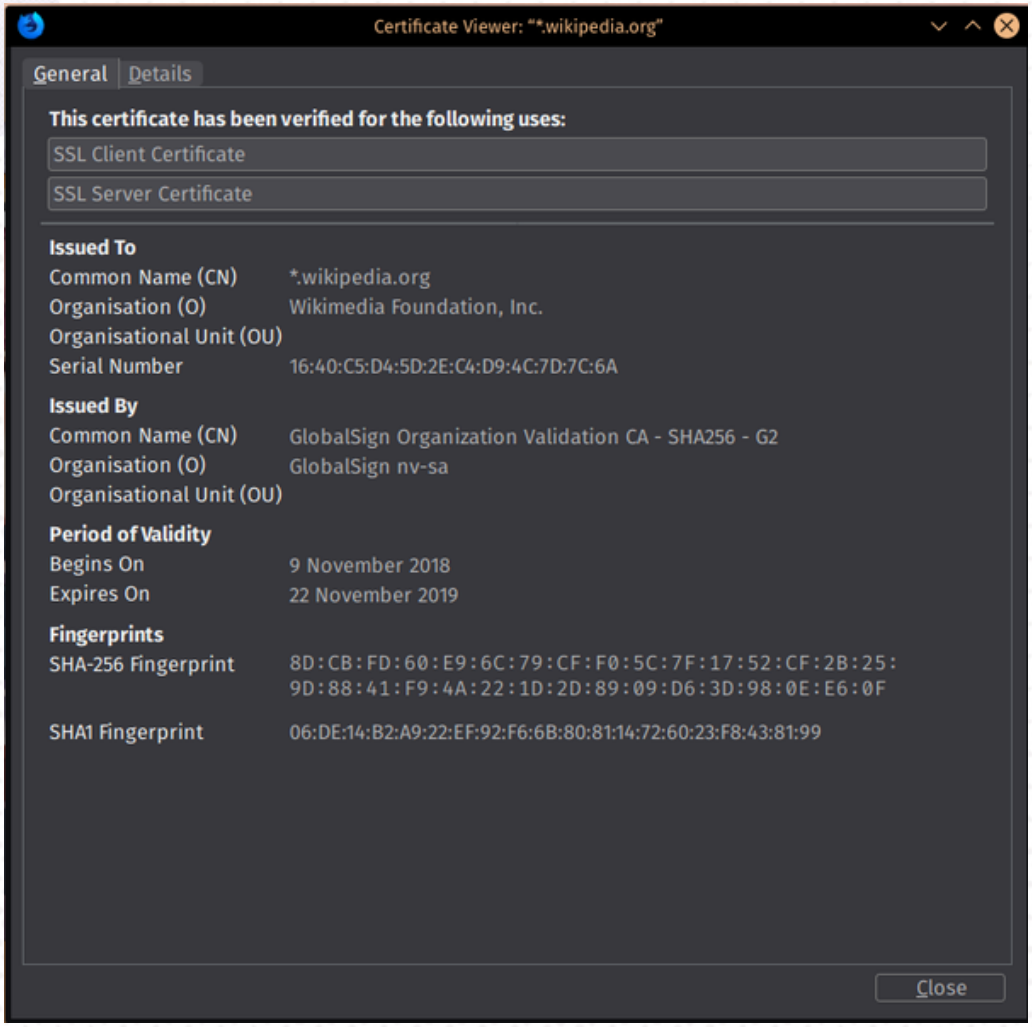
Case Study: Asymmetric Encryption (Cont.)

- Advantages –
  - Confidentiality: Eve needs Bob's private key to decrypt the message
  - Integrity: Message tampering is detected during the decryption process
  - Authentication: Alice is the only one who can encrypt with her private key (Note: this process is called "signing")
  - Non-repudiation: Bob can prove to a third party that Alice is the originator of the message
- Disadvantages –
  - MITM Attack: Eve can work as proxy between Alice and Bob communication

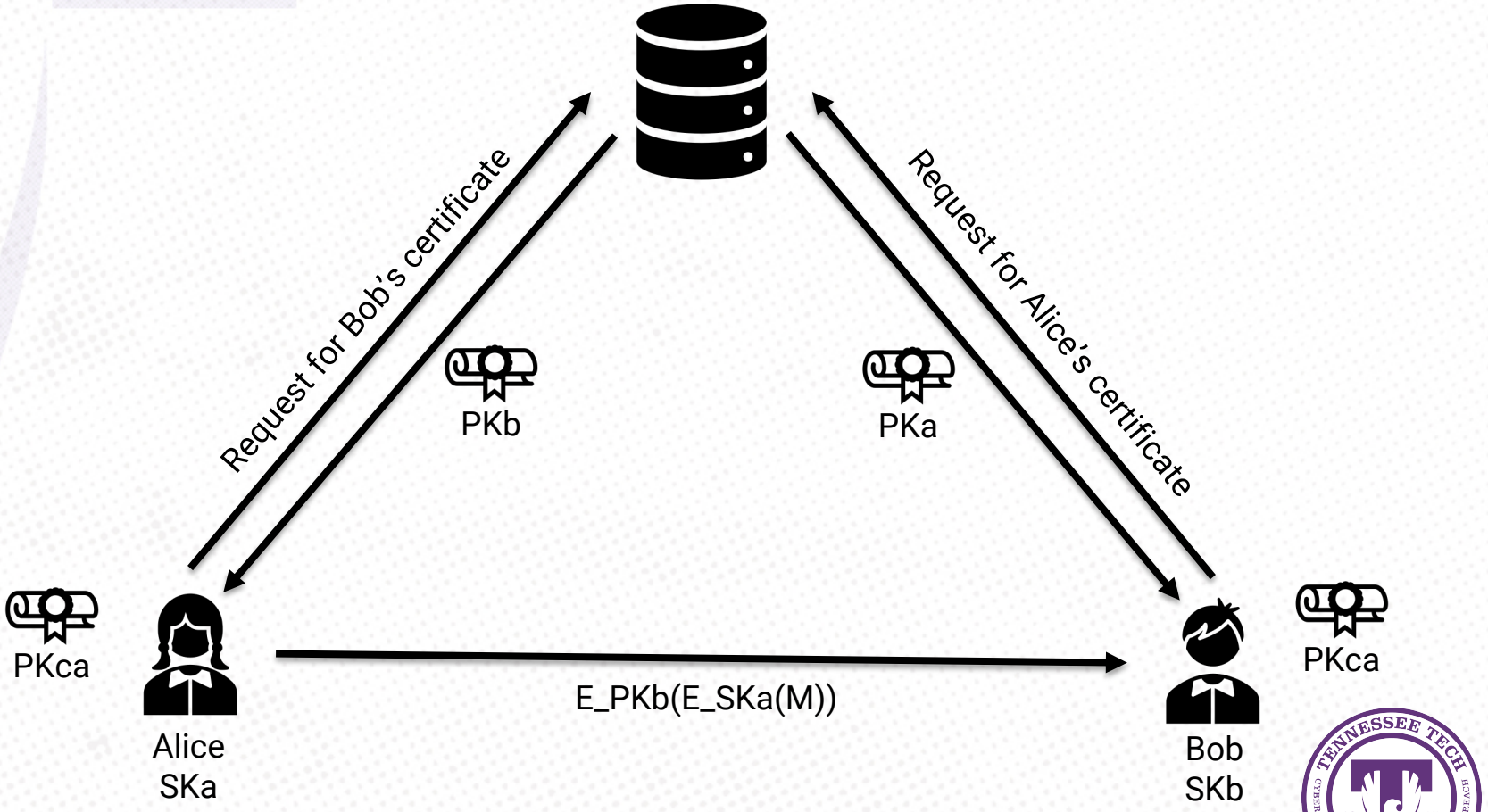
Public Key Certificates

- Used to associate an identity (e.g., Alice and Bob) with a public key
- The association is guaranteed by a trusted third party, the Certification Authority (CA)
- The CA owns a public key and a private key PKca, SKca
- The CA creates a self-signed certificate that is distributed through many channels to avoid interception
- The CA signs certificates containing an identity and the corresponding public key after having verified the identity of the requester
- Certificates are made available in public database

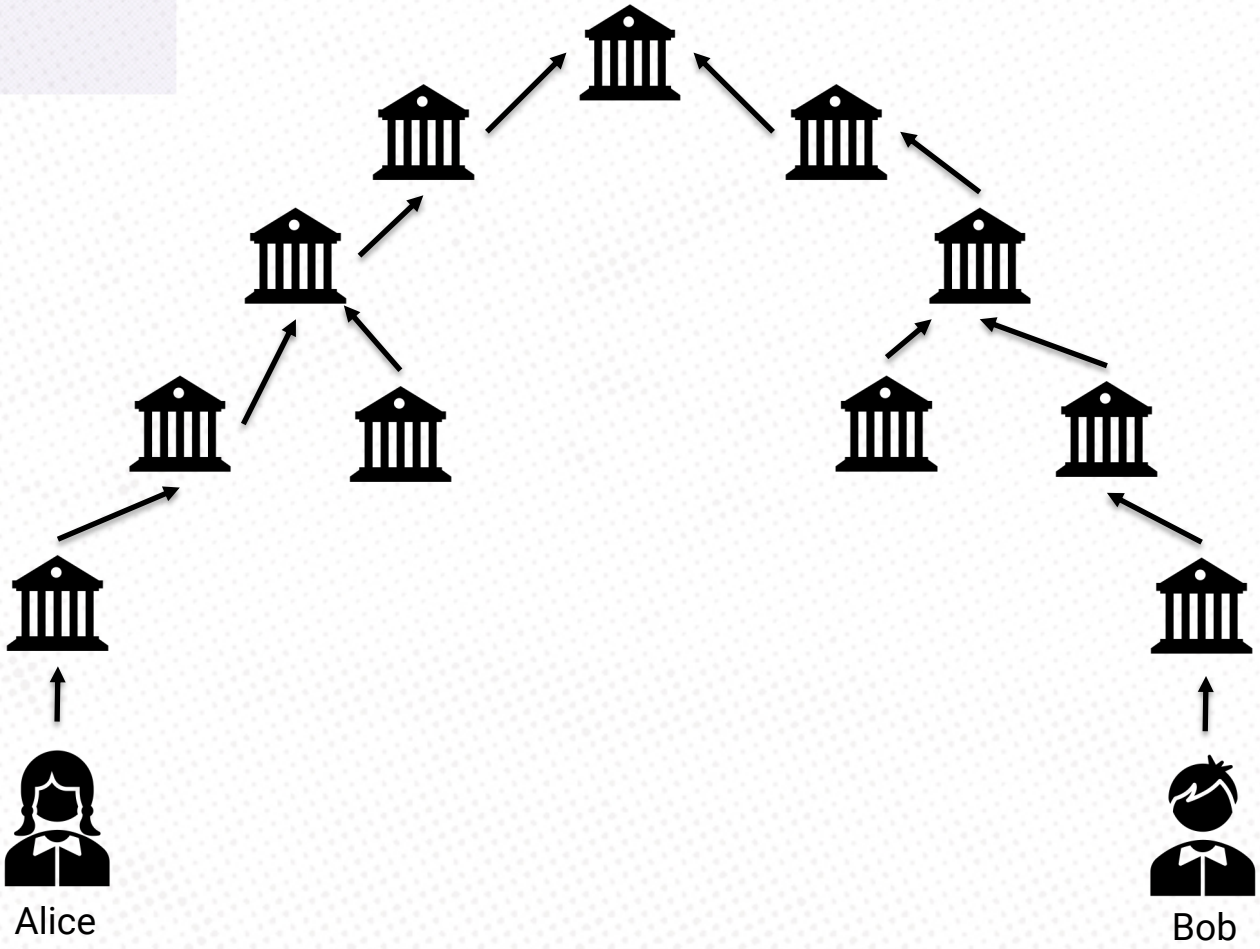
X.509 Certificate



Communication using Certificates



Public Key Infrastructure (PKI)



Demonstration

Secure Messaging and OpenPGP

- Pretty Good Privacy (PGP) is an application for secure messaging originally developed by Phillip Zimmerman.
- OpenPGP (RFC 2440) standardized the message-exchange packet formats used to provide encryption, decryption, signing, and key management functions.
- Open software applications, such as, **GnuPG**, use the standard to achieve interoperability.

Secure Messaging and OpenPGP (Cont.)

- OpenPGP provides data integrity services for messages and data files using:
  - Digital signature
  - Encryption
  - Compression and radix-64 conversion (ASCII armor)

GnuPG

```
ahsan@debian:~$ gpg --help
gpg (GnuPG) 2.2.12
libgcrypt 1.8.4
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/ahsan/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2

Syntax: gpg [options] [files]
Sign, check, encrypt or decrypt
```

User and Keys

```
ahsan@debian:~$ gpg --gen-key
gpg (GnuPG) 2.2.12; Copyright (C) 2018 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: keybox '/home/ahsan/.gnupg/pubring.kbx' created
Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: Ahsan Ayub
Email address: mayub42@students.tntech.edu
You selected this USER-ID:
    "Ahsan Ayub <mayub42@students.tntech.edu>"
```

User and Keys (Cont.)

```
gpg: /home/ahsan/.gnupg/trustdb.gpg: trustdb created
gpg: key 2E5644EC28418D3C marked as ultimately trusted
gpg: directory '/home/ahsan/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/ahsan/.gnupg/openpgp-revocs.d/CD3
CD09D17B97CE1DC035E652E5644EC28418D3C.rev'
public and secret key created and signed.

pub   rsa3072 2019-11-21 [SC] [expires: 2021-11-20]
      CD3CD09D17B97CE1DC035E652E5644EC28418D3C
uid                      Ahsan Ayub <mayub42@students.tntech.edu>
sub   rsa3072 2019-11-21 [E] [expires: 2021-11-20]
```

User and Keys (Cont.)

```
ahsan@debian:~$ gpg --output public.pgp --export --armor Ahsan Ayub
ahsan@debian:~$ gpg --output private.pgp --export-secret-key --armor Ahsan Ayub
ahsan@debian:~$ ls -l
total 48
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Desktop
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Documents
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Downloads
dr-xr-xr-x 2 nobody nogroup 4096 Nov 14 11:47 ftp
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Music
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Pictures
-rw------- 1 ahsan  ahsan   5221 Nov 21 09:58 private.pgp
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Public
-rw-r--r-- 1 ahsan  ahsan   2464 Nov 21 09:58 public.pgp
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Templates
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Videos
```

User and Keys (Cont.)

-----BEGIN PGP PUBLIC KEY BLOCK-----

Version: GnuPG v1.4.11 (GNU/Linux)

mQINBFJPCuABEACiog/sInjg0O2SqgmG1T8n9FroSTdN74uGsRMHHAOuAmGLsTse
9oxeLQpN+r75Ko39RVE88dRcW710fPY0+fjSXBKhpN+raRMUKJp4AX9BJd00YA/4
EpD+8cDK4DuLlLdn1x0q41VUsznXrnMpQedRmAL9f9bL6pbLTJhaKeorTokTvdn6
5VT3pb2o+jr6NETaUxd99ZG/osPar9tNThVLIIzG1nDabcTFbMB+w7wOJuhXyTLQ
JBU9xmavTM71PfV6Pkh4j1pfWImXc1D8dS+jcvKeXInBfm2XZsfOCesk12YnK3Nc
u1Xe1lxzSt7Cegum4S/YuxmYoh462oGZ7FA4Cr2lvAPVpO9zmgQ8JITXiqYg2wB3

. . .

User and Keys (Cont.)

```
ahsan@debian:~$ gpg --list-key
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2021-11-20
/home/ahsan/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2019-11-21 [SC] [expires: 2021-11-20]
      CD3CD09D17B97CE1DC035E652E5644EC28418D3C
uid           [ultimate] Ahsan Ayub <mayub42@students.tntech.edu>
sub   rsa3072 2019-11-21 [E] [expires: 2021-11-20]

pub   rsa3072 2019-11-21 [SC] [expires: 2021-11-20]
      83FAEE2EB953D5383A708FFCF875A2142E7A38E8
uid           [ultimate] Ayub Ahsan <mahsan42@students.tntech.edu>
sub   rsa3072 2019-11-21 [E] [expires: 2021-11-20]

ahsan@debian:~$ gpg --list-secret-key
/home/ahsan/.gnupg/pubring.kbx
------------------------------
sec   rsa3072 2019-11-21 [SC] [expires: 2021-11-20]
      CD3CD09D17B97CE1DC035E652E5644EC28418D3C
uid           [ultimate] Ahsan Ayub <mayub42@students.tntech.edu>
ssb   rsa3072 2019-11-21 [E] [expires: 2021-11-20]

sec   rsa3072 2019-11-21 [SC] [expires: 2021-11-20]
      83FAEE2EB953D5383A708FFCF875A2142E7A38E8
uid           [ultimate] Ayub Ahsan <mahsan42@students.tntech.edu>
ssb   rsa3072 2019-11-21 [E] [expires: 2021-11-20]
```

User and Keys (Cont.)

```
ahsan@debian:~$ gpg --output public.pgp --export --armor Ahsan Ayub
ahsan@debian:~$ gpg --output private.pgp --export-secret-key --armor Ahsan Ayub
ahsan@debian:~$ ls -l
total 48
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Desktop
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Documents
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Downloads
dr-xr-xr-x 2 nobody nogroup 4096 Nov 14 11:47 ftp
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Music
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Pictures
-rw------- 1 ahsan  ahsan   5221 Nov 21 09:58 private.pgp
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Public
-rw-r--r-- 1 ahsan  ahsan   2464 Nov 21 09:58 public.pgp
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Templates
drwxr-xr-x 2 ahsan  ahsan   4096 Nov 14 09:21 Videos
```

Encryption

```
ahsan@debian:~$ gpg -e -u "Ahsan Ayub" -r "Ayub Ahsan" message.txt
ahsan@debian:~$ ls
Desktop    Downloads  message.txt      Music     private2.pgp  Public       public.pgp  Videos
Documents  ftp        message.txt.gpg  Pictures  private.pgp   public2.pgp  Templates
ahsan@debian:~$ cat message.txt
This is a simple message.
ahsan@debian:~$ cat message.txt.gpg
[binary PGP output]
ahsan@debian:~$
```

Decryption

```
ahsan@debian:~$ gpg -d message.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID B8DA013A5ACA4209, created 2019-11-21
      "Ayub Ahsan <mahsan42@students.tntech.edu>"
This is a simple message.
```

Signing a Message in Clear Format

```
ahsan@debian:~$ gpg --encrypt --sign -r "Ayub Ahsan" message.txt
ahsan@debian:~$ gpg -d message.txt.gpg
gpg: encrypted with 3072-bit RSA key, ID B8DA013A5ACA4209, created 2019-11-21
      "Ayub Ahsan <mahsan42@students.tntech.edu>"
This is a simple message.
gpg: Signature made Thu 21 Nov 2019 10:37:11 AM CST
gpg:                using RSA key CD3CD09D17B97CE1DC035E652E5644EC28418D3C
gpg: Good signature from "Ahsan Ayub <mayub42@students.tntech.edu>" [ultimate]
```

Signing a Message in Clear Format (Cont.)

```
ahsan@debian:~$ gpg --clear-sign message.txt
ahsan@debian:~$ cat message.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

This is a simple message.
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEEzTzQnRe5fOHcA15lLlZE7ChBjTwFAl3WveMACgkQLlZE7ChB
jTzYTAwAn71lTwKlc4c4DrKQVQVnOmHRlIDwW3eHT27Txb7WIVUqgmZh7Z0aJPJf
nmqieqrN+uTJM5P5SI2BHMUlVtX2RevjazkdGgjZgLzJ0HGz+1Y6cSz3InnICXvG
x2772jfxedt0f3vOuvVDwgC4sg14qxVpDJofeK9/yvi7g4MP/ivGtfGQmCGKvcwk
lWqjFf92VGCuEZhsn4xkBx0shym4UEOPjQsbCn4d/75tgFxaspSKy4j8eBHjiqTO
Gg8dVe2MdhSOHQDVnIScaD69LhWMVNw3v+HrDi4kjudjhOCTXylcnqu5DWlJHneK
tMHgn8KuZeKGzLt5xdo64yIoEPT2815ZiEM3e3CpQbAsz622tF4nod15siHuXAqc
PCNANlhXNyClk/rWZrARTeGI7uBt/TEvM8tQr+ArzoRQ+GpgdDT5MrddofGgFrxi
0Re0LYzGO/gLfgx5UI8Hyccm/6LTpJ0qT/eIEHdFWTzfdNI19TxILtPHfCYhRnZa
LPZtJi0n
=jPfG
-----END PGP SIGNATURE-----
```

Adversarial Example

Bad Usage of Cryptography

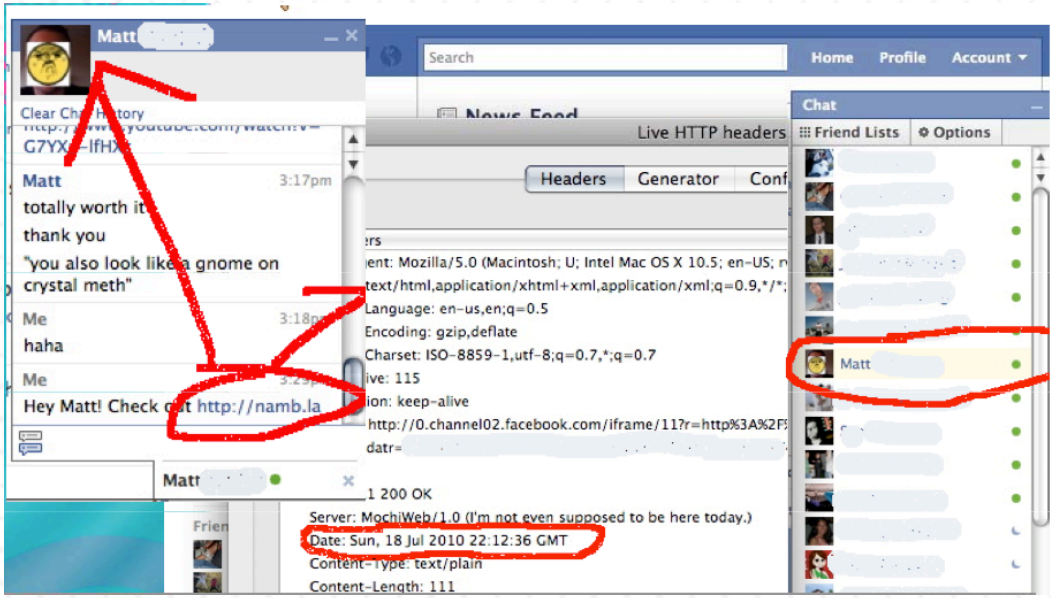
- ”How I met your girlfriend”, by Samy Kamkar (BlackHat 2010)
- Identified how PHP session cookies are created

session.c

```
PHPAPI char *php_session_create_id(PS_CREATE_SID_ARGS) /* {{{ */
{
    /* ... code ... */

    gettimeofday(&tv, NULL);

    /* ... code ... */

    spprintf(&buf, 0, "%.15s%ld%ld%0.8F",
        remote_addr ? remote_addr : "",   /* IP Address:     32 bits */
        tv.tv_sec,                        /* epoch:          32 bits */
         (long int)tv.tv_usec,            /* microseconds:   32 bits */
        php_combined_lcg(TSRMLS_C) * 10); /* rand lcg_value: 64 bits */
    /* ... code ...                      ** TOTAL:         160 bits */

    PHP_SHA1Update(&sha1_context, (unsigned char *) buf, strlen(buf));

    /* ... code ...                      ** SHA1 string:   160 bits */
```

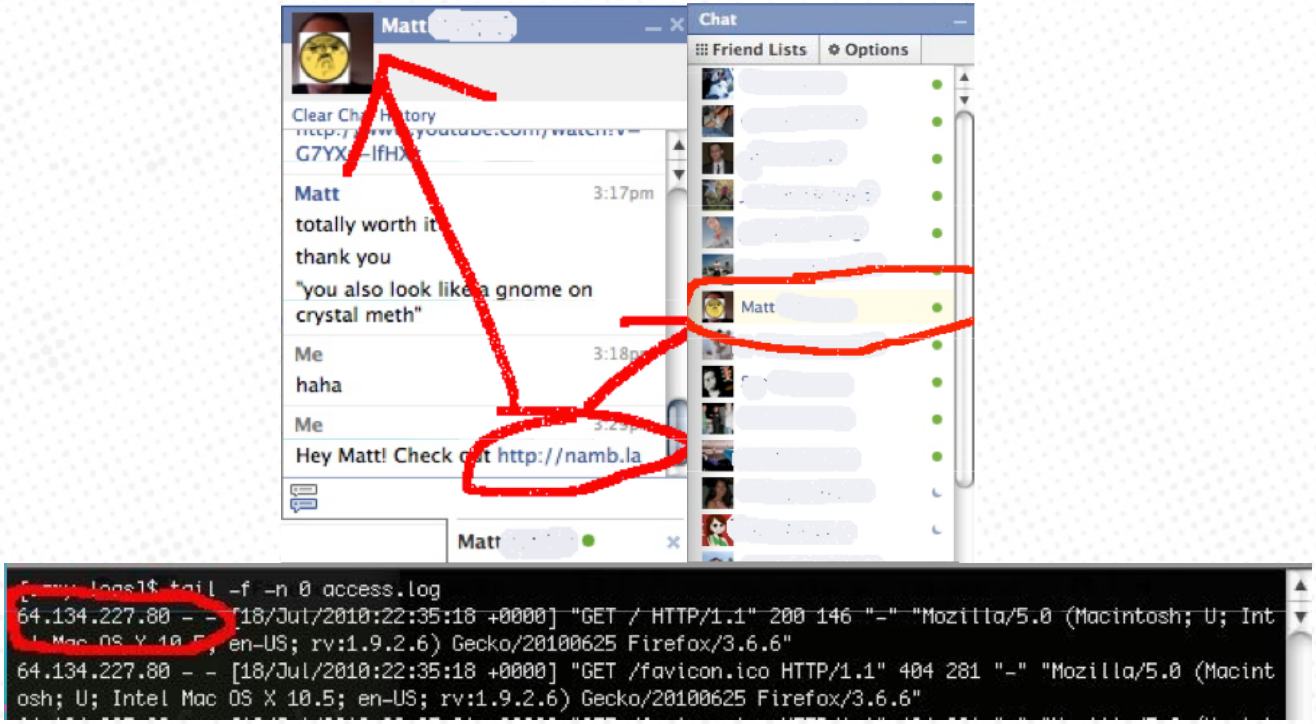
Bad Usage of Cryptography (Cont.)

- IP address (32 bit of entropy)
- Microseconds can only be 0 – 999,999 (~ **20** bits of entropy)
- Random value (64 bits of entropy)
- Epoch can be guessed by monitoring when somebody comes online (**0** bits of entropy)
- Total: **116** bits



Bad Usage of Cryptography (Cont.)

- The IP address of the user can be stolen by luring the user to connect a website under the attacker's control
- IP address entropy (**0** bit entropy)
- Total: **84** bits



Bad Usage of Cryptography (Cont.)

- PHP's seeding procedures are broken –
  - It uses a composition of time and the PID of the process

lcg.c

```
PHPAPI double php_combined_lcg(TSRMLS_D)
{
    php_int32 q;
    php_int32 z;

    if (!LCG(seeded)) {
        lcg_seed(TSRMLS_C);
    }
    /* ... code ... */
}

static void lcg_seed(TSRMLS_D)
{
    struct timeval tv;

    if (gettimeofday(&tv, NULL) == 0)
        LCG(s1) = tv.tv_sec ^ (~tv.tv_usec); /* s1 = 32 bits */
    LCG(s2) = (long) getpid();               /* s2 = 32 bits */

    LCG(seeded) = 1;
}
```

Line: 72 Column: 46 C Tab Size: 4 lcg_seed

Bad Usage of Cryptography (Cont.)

- Only the lower 20 bits matter (for some long period of time)
- Total seed (s1 + s2) is 52 bits; total left is 72 bits

```
LCG(s1) = tv.tv_sec ^ (~tv.tv_usec)
LCG(s1) = epoch ^ (~ [20 random bits])

~0          = 11111111111111111111111111111111
~1,000,000 = 11111111111100001011110110111111 (same 12 bits)

epoch = 1279493871
epoch = 01001100010000111000011011101111 (static / unknown)
epoch^= 01001100010000000000000000000000
epochv= 01001100010011111111111111111111

epoch^= Thu Jul 15 23:45:20 2010
epochv= Wed Jul 28 03:01:35 2010
epoch diff = 12+ days
```

Bad Usage of Cryptography (Cont.)

- The process ID can only be up to 32,768 and therefore only **15** of the 32 bits matter
- Result: the PRNG seed is only **35** bits with a decrease from 52 bits.
  - Total bits left: **55** bits
- It one learns the PID (e.g., by executing PHP code on the server) it goes down to **20** bits with a decrease from 35 bits
- Total bits of entropy left: **40**

Bad Usage of Cryptography (Cont.)

- If the attacker can execute lcg_value() on the server, it is possible to determine the seed (*pre-computation* the sequence)
- Total bits of entropy left: **20**
- Twenty bits correspond to 1,048,576 cookies
  - Can easily be brute-forced!

Bad Usage of Cryptography (Cont.)

- Takeaway
  - Understand the source of entropy
  - Understand the threat model

Acknowledgement

- Prof. Dr. Giovanni Vigna
  - Fall 2017: CS279 Advanced Topics in Security class
  - University of California in Santa Barbara (UCSB)
  - Available on YouTube
- Mentor-in-Training (DCIG) at Tennessee Tech University

THANK YOU!


TENNESSEE TECH
19
15
CYBERSECURITY EDUCATION, RESEARCH & OUTREACH