Encryption methods

- Symmetric cryptography
  - Sender and receiver know the secret key (apriori)
    - Fast encryption, but key exchange should happen outside the system
- Asymmetric cryptography
  - Each person maintains two keys, public and private
    - M ≡ PrivateKey(PublicKey(M))
    - M ≡ PublicKey (PrivateKey(M))
  - Public part is available to anyone, private part is only known to the sender
  - E.g. Pretty Good Privacy (PGP), RSA

My Public Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

mQGiBDqtLPwRBADnG0+9IkDvI8t/3wdL3CSO4DytEH0NjrNwAYYIaewp3MklsxkP
p6iVblwiiCH4T4Nqkaru+kaEQlhSTa7E/F9yQCWN5J0u1U7mtgTKFyt7VG0txAVx
tV7TuyxNogJkpm2BqoKqqUdCdbm+GurX/G2ynbINjEOvhcy0ilttxgyDrwCg/8HZ
tM0i06VVNcR/QCmA+JdHGwMEAIjXLVV97huEtpuWDiq4J53ecV3HXQm6XoUZq4Sc
n+nsvXe4UD+6ldub/riOqBy22fBBAKhUsM3lGFgr7h19X3RGdw/yBVox+BLajpW+
F+ddjJAVSFeTvNanhnXL9a3nwCThb4aEUTdD61kgoUWJl2BnsK1DUSo2X6AsZYo+
GknOA/92dUNYUzspPLkXvPjOo+uJErZA4aN+UYsJwD3AlYugVLkc3nQBQySO4bAR
XitjnN0DA6Kz/j6e+cqReCyEuBnPtaY/Nn/dAn1lgUlJ/EtKQ9J4krI3+RxRmlpY
UtWyTaakV/QCXkB/yB9i6iAfsCprlcRSpmZAGuNXr+pHTHB0ILQmU3VyZW5kYXIg
Q2hhbmRyYSA8c3VyZW5kYXJAY3MudWdhLmVkdT6JAFgEEBECABgFAjqtLPwICwMJ
CAcCAQoCGQEFGwMAAAAACgkQlU7dFVWfeisqTACfXxU9almbouW2nbWdx6MHatQ6
TOgAoM9W1PBRW8Iz3BIgcnSsZ2UPNJHDuQINBDqtLPwQCAD2Qle3CH8IF3Kiutap
QvMF6PlTETlPtvFuuUs4INoBplajFOmPQFXz0AfGy0Op1K33TGSGSfgMg711GRfU
odNQ+PVZX9x2Uk89PY3bzpnhV5JZzf24rnRPxfx2vIPFRzBhznzJZv8V+bv9kV7H
AarTW56NoKVyOtQa8L9GAFgr5fSI/VhOSdvNILSd5JEHNmszbDgNRR0PfIizHHxb
LY7288kjwEPwpVsYjY67VYy4XTjTNP18FldDox0YbN4zISylKv884bEpQBgRjXyE
pwpylobEAxnIByl6ypUM2Zafq9AKUJsCRtMIPWakXUGfnHy9iUsiGSa6q6Jew1Xp
Mgs7AAICCACLxNC3Vth553Y90JCVyM9mPWzvrkjfEGfBiCFDZ0HONW81ywUyV6jT
O/lsUsgR7jGB26XBsnIY96a9WTpUoI+20YstFLRj8sXOVXuaP/YTmgSLv82O6SWd
Bze1S0YJcU31/zdCftsz67UWT8vg39yeGyQ5KQP83p9DKpi4Z5K4M29p8eCt9BY+
kid94h9+16ZT8JLF0iEwGapZvpaTucCNoC8t6CKPto0dGpkYp7uBYoSzLgNvUh2n
BjGVEmLuioabqbOaomDErITY2iNcW3CCgjjYvgg/Hnu7HB2xKzuVUN1NTGogcuNI
Yx88mi+d/HxTY6YNr9xNW0f0pWkZDVB0iQBMBBgRAgAMBQI6rSz8BRsMAAAAAAoJ
EJVO3RVVn3orYhIAoIQPxGvHmX8c6kaAZqkolzYCeixcAJ9tp5h/KQZrIN/BpyTW
9Xgv4qxKEA==
=Pv5O
-----END PGP PUBLIC KEY BLOCK-----
```

RSA

- Named after Rivest, Shamir and Adleman
  - Only receiver receives message:
    - Encode message using receivers public key
  - Only sender could've sent the message
    - Encode message using sender's private key
  - Only sender could've sent the message and only receiver can read the message
    - Encode message using receivers public key and then encode using our private key

Strength

- Strength of crypto system depends on the strengths of the keys
- Computers get faster – keys have to become harder to keep up
- If it takes more effort to break a code than is worth, it is okay
  - Transferring money from my bank to my credit card and Citibank transferring billions of dollars with another bank should not have the same key strength

Public Key Infrastructure (PKI)

- Process of issuing, delivering, managing and revoking public keys
- E.g. Secure Sockey Layer (SSL)
  - Client C connects to Server S
    1. C requests server certificate from S
    2. S sends server certificate with Spublic to C
    3. C verifies validity of Spublic
    4. C generate symmetric key for session
    5. C encrypts Csymmetric using Spublic
    6. C transmits Csymmetric(data) and Spublic(Csymmetric) to S

Authentication

- Identification verification process
  - E.g. kerberos certificates, digital certificates, smart cards
- Used to grant resources to authorized users

Practical Public Key Cryptosystem

1. Decrypt(Encrypt(Message)) = Message
2. Encrypt() and Decrypt() are easy to compute
3. Encrypt() does not reveal Decrypt()
4. Encrypt(Decrypt(Message)) = Message

- Function satisfying 1-3: Trap-door one-way function
  - One way: easy to compute in one direction, difficult in the other direction
  - Trap-door: Inverse functions are easy to compute once certain private “trap-door” information is known.
- 1-4: permutation

Signature

- Encrypt using private key of sender. Anyone can decrypt using the public key of sender to verify signature

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hello world!!
-----BEGIN PGP SIGNATURE-----
Version: PGPfreeware 7.0.3 for non-commercial use
   <http://www.pgp.com>

iQA/AwUBOq8LO5VO3RVVn3orEQLFZwCdGi9AWvlhollaYmr9TPvtdbK
   oe20AoLLr
vbJ8SgkIZ73lCy6SXDi91osd
=L3Sh
-----END PGP SIGNATURE-----
```

Privacy

- Encrypt with receivers public key

-----BEGIN PGP MESSAGE-----
Version: PGPfreeware 7.0.3 for non-commercial use <http://www.pgp.com>

qANQR1DBwU4D30m79rqmjHMQB/4q1mu3IP8AsMBYSUW6udXZnF0/LVL51eYzVnAW
lxgbxhHmBoZf9YEltoXw82gkgVebz+3Xfj6T5mLNy5FA6cgKKw57AY9Bl3aEKlJK
/nV5qR8E/VZOhaPoog8dtV1Hpi5Z0vNCI7s5Ibp3C2tlrgYtvyYfe86bqCNe3yAI
btTUT+bA9HL3pXqhOoWlRB+N58T9ybn/9FyonYYfGuPdMTj+ZciK37R+ezWg5YmZ
jdDMf/CxgllMF/Tv2jQ8KgmrKIyi6gWQmEtWzFUlAPgdpOC7TQC3sQqVjK4GyOY6
WnrXiWqO3895ukBGyHzqyssUTJFe5qnclkrmCvA3tph+uc7pCACKrYaGLSWWoQSB
L6zch2GnhG4+JpDCVKF/poJ1URkB2Odd9/OCReR0sFXZFvW14IJQznu3HOhjtA+y
g7Nn736fqMD9jpBZFfUtKv/v4JMyWcRdp3R3icm03zi24n+244r1DQj+cVlFYPfd
zRAGTLORVjXH2amGqilKyxqMU7ZYXIMI43bFIviu4tabKYnZJxpM8keUKA3u+vPs
X9ksSoBSiT6Kow3Lac2t3Qo5TimYlS5ODFnC6Pp9aRZzNcBOKmiYO4IIbdFH2jta
RbcmesEjH5RpbDV4BfcOMdm2UGWZe6kAaKkSdxHlUVZAJnesbT+lQf4AZjXkmsOM
8qnBKi5xyS/wrhS4zamV/Mp+5qIGNASXUHPsp3rukovaZANdZ/Y6zNQQVim0kphd
5ECybmVrHQ==
=S9ph
-----END PGP MESSAGE-----

Algorithm

- To break their algorithm requires that you factor a large prime
  - Computationally very hard. Can't be "proven" yet
  - With present technology, 512 bit key takes a few months to factor using "super computers", 1024 takes a long time and 2048 takes a very long time
  - Takes 2 seconds to generate a 2048 bit key on a 933 Mhz Pentium
  - Algorithm has remained secure for the past 17 years
  - One of the most successful public key system

Case study: Multics

- Goal: Develop a convenient, interactive, useable time shared computer system that could support many users.
  - Bell Labs and GE in 1965 joined an effort underway at MIT (CTSS) on Multics (Multiplexed Information and Computing Service) mainframe timesharing system.
- Multics was designed to the swiss army knife of OS
- Multics achieved most of the these goals, but it took a long time
  - One of the negative contribution was the development of simple yet powerful abstractions (UNIX)

Multics: Designed to be the ultimate OS

- "One of the overall design goals is to create a computing system which is capable of meeting almost all of the present and near-future requirements of a large computer utility. Such systems must run continuously and reliably 7 days a week, 24 hours a day in a way similar to telephone or power systems, and must be capable of meeting wide service demands: from multiple man-machine interaction to the sequential processing of absentee-user jobs; from the use of the system with dedicated languages and subsystems to the programming of the system itself; and from centralized bulk card, tape, and printer facilities to remotely located terminals. Such information processing and communication systems are believed to be essential for the future growth of computer use in business, in industry, in government and in scientific laboratories as well as stimulating applications which would be otherwise undone."

Contributions

- Segmented memory
  - The Multics memory architecture divides memory into segments. Each segment has addresses from 0 to 256K words (1 MB). The file system is integrated with the memory access system so that programs access files by making memory references.
- Virtual memory
  - Multics uses paged memory in the manner pioneered by the Atlas system. Addresses generated by the CPU are translated by hardware from a virtual address to a real address. A hierarchical three-level scheme, using main storage, paging device, and disk, provides transparent access to the virtual memory.
- High-level language implementation
  - Multics was written in the PL/I language, which was, in 1965, a new proposal by IBM. Only a small part of the operating system was implemented in assembly language. Writing an OS in a high-level language was a radical idea at the time.

Contributions (cont)

- Shared memory multiprocessor
  - The Multics hardware architecture supports multiple CPUs sharing the same physical memory. All processors are equivalent.
- Multi-language support
  - In addition to PL/I, Multics supports BCPL, BASIC, APL, FORTRAN, LISP, C, COBOL, ALGOL 68 and Pascal. Routines in these languages can call each other.
- Relational database
  - Multics provided the first commercial relational database product, the Multics Relational Data Store (MRDS), in 1978.
- Security
  - Multics was designed to be secure from the beginning. In the 1980s, the system was awarded the B2 security rating by the US government NCSC, the first (and for years only) system to get a B2 rating.

Contributions (cont.)

- On-line reconfiguration
  - As part of the computer utility orientation, Multics was designed to be able to run 7 days a week, 24 hours a day. CPUs, memory, I/O controllers, and disk drives can be added to and removed from the system configuration while the system is running.
- Software Engineering
  - The development team spent a lot of effort finding ways to build the system in a disciplined way. The Multics System Programmer's Manual (MSPM) was written before implementation started: it was 3000 or so pages and filled about 4 feet of shelf space in looseleaf binders. (Clingen and Corbató mention that we couldn't have built the system without the invention of the photocopier.) High level language, design and code review, structured programming, modularization and layering were all employed extensively to manage the complexity of the system, which was one of the largest software development efforts of its day.

Multics to UNIX

- the group effort initially failed to produce an economically useful system.
  - Bell Labs withdrew from the effort in 1969
  - Bell Labs Computing Science Research Center in Murray Hill -- Ken Thompson, Dennis Ritchie, Doug McIlroy, and J. F. Ossanna -- went on to develop UNIX (note pun)

Legacy - Positive and negative

- UNIX:
  - Ken Thompson and Dennis Ritchie, the inventors of UNIX, worked on Multics until Bell Labs dropped out of the Multics development effort in 1969. The UNIX system's name is a pun on Multics attributed to Brian Kernighan. Some ideas in Multics were developed further in UNIX.
- GCOS 6
  - Honeywell's GCOS 6 operating system for the Level 6 minicomputers was strongly influenced by Multics.
- Primos
  - Prime's Primos operating system shows a strong Multics influence. Bill Poduska worked on Multics at MIT before founding Prime, and several other senior Multicians worked at Prime. Poduska referred to Primos as "Multics in a shoebox."

Legacy

- VOS
  - Stratus's VOS operating system shows a strong Multics influence. Bob Freiburghouse, former Multics languages manager, was one of the founders of Stratus; many Multicians are still Stratus employees.
- Apollo Domain
  - Bill Poduska went on from Prime to help found Apollo, and Domain was known as "Multics in a Matchbox." Apollo's OS shows strong Multics influence. For instance, the basic access to stuff on disk is via a single-level store directly based on Multics. Supposedly some of the motivation for the object-store style of file system came from Multics too. (Info from Frederick Roeber) [Jerry Saltzer adds:] In addition, it uses a shared memory model, despite being distributed across a network
- NTT DIPS
  - NTT undertook a massive effort to clone Multics, which led to their DIPS (Denden Information Processing System) series of mainframes. DIPS machines are still in widespread use in Japan today by NTT, but everyone agrees that they are going away. I believe that Intermetrics developed the DIPS PL/I compiler for NTT.

Legacy

- Amber
- IBM System 360
- Tenex, TOPS 20, GCOS etc etc

- Most of the the project members were influential in shaping the computer industry and they took Multics ideas with them

- Monolithic vs microkernel debate