Topic

- ☐ **Disclaimer**
  - ☐ **I'm not a pro**
  - ☐ **No warranty, as-is, etc.**
  - ☐ **This requires some thought to tailor to your situation and understand the risks**
    - ☐ Only you know your threat model
- ☐ **Introduction**
  - ☐ **Introduction**
    - ☐ PGP is a protocol & some ideas
    - ☐ GPG is one implementation of many
    - ☐ I'd love to show you everything straight from GPG, but I'm going to keep it more general at first
- ☐ **Cryptography Primer**
  - ☐ **What is cryptography? Encryption**
    - ☐ Providing confidentiality — plaintext into ciphertext using a "key"
    - ☐ Mathematical operations — security is only dependent on knowledge of the key
    - ☐ Security is provided by the selection of large enough numbers that make reversing/brute-forcing the mathematical operations difficult and time-consuming
  - ☐ **Symmetric Key**
    - ☐ One key for encryption and decryption
    - ☐ Fast, but there's a key distribution problem
    - ☐ Common algorithms: AES, DES, Blowfish, etc.
  - ☐ **Public (Asymmetric) Key**
    - ☐ One key for can encrypt, the other can decrypt, and vice versa
    - ☐ Typically called "public key" because one is made "public" and one is kept "private"/secret — nice properties... different key distribution problems, but more on that later
    - ☐ Relatively slow, but there are good workarounds
    - ☐ Common algorithms: RSA, ElGamel (Diffie-Hellman), DSA
  - ☐ **Hybrid Cryptosystem**
    - ☐ Leveraging benefits of both algorithms
    - ☐ Symmetric encryption for data confidentiality
    - ☐ Asymmetric encryption for symmetric key confidentiality
    - ☐ "Session key"
  - ☐ **The other "half": Signatures**
    - ☐ Provide integrity — assurance that a message was not tampered with
    - ☐ Provide authenticity — assurance that a message came from its purported author
      - ☐ Non-repuditation
    - ☐ Typically implemented as encryption of a compact representation of the message
  - ☐ **Hash/Message Digest Function**
    - ☐ Given some arbitrary input, generates a fixed length output that is unique to the input, and will change if the input accidentally or intentionally changes
    - ☐ Features
      - ☐ Difficult to invert (preimage resistance): given d, find an m for d=H(m)
      - ☐ Difficult to find a second equal-hashing input (second preimage resistance)
      - ☐ Difficult to find arbitrary equal-hashing inputs (collision resistance)

Topic

- [ ] [source: http://en.wikipedia.org/wiki/Cryptographic_hash_function]
- [ ] Common algorithms: SHA, MD5, RIPEMD
- [ ] Difference between authenticity and integrity: CRC, non-HMAC hash, etc.

- [ ] **PGP history**
  - [ ] **Historical backdrop**
    - [ ] Bills outlawing cryptography
    - [ ] Clipper chip telephones, and FBI key escrow
  - [ ] **Phil Zimmerman**
    - [ ] software developer, anti-nuclear activist
    - [ ] PGP = Pretty Good Privacy
    - [ ] original release via ftp in 1991
    - [ ] Email-focused
  - [ ] **Export Controls**
    - [ ] First amendment > ITAR
    - [ ] Publish source as a book
  - [ ] **MIT Involvement**
    - [ ] Pushing to develop a non-encumbered version
    - [ ] Maintenance & Release
    - [ ] Standardization at IETF
    - [ ] jis, jdb, hal, simsong
  - [ ] **Up to present**
    - [ ] OpenPGP RFC
      - [ ] interoperability
      - [ ] 1991, 2440, 4880
    - [ ] PGP Corporation
    - [ ] New legal & policy environment
  - [ ] **PGP "key" ideas**
    - [ ] Public key crypto
    - [ ] No central point of control
    - [ ] Distributed web trust
    - [ ] Strong crypto for everyone
- [ ] **PGP concepts**
  - [ ] **PGP Operations**
    - [ ] Symmetric
      - [ ] compression first
    - [ ] Encrypt/decrypt
      - [ ] compression first
    - [ ] Sign/Verify
      - [ ] canonicalizes text files by changing line-endings to CRLF
    - [ ] ASCII Armor
      - [ ] Transmit binary over base64 email
  - [ ] **Your Keys**
    - [ ] Key Storage Problem: Need to keep these keys somewhere to stay organized and ease of use
    - [ ] Keyring — at least two, one with secret key(s) and one with public keys
      - [ ] some kind of DB (often — as in GnuPG — just a blob) of all public keys

Topic

- ☐ Public Key
  - ☐ pgpdump my public key
  - ☐ uids — binding identity to public key material
  - ☐ expiry dates
  - ☐ algorithms for pre-negotiation
  - ☐ signatures — more on this later
- ☐ Secret Key — encrypted with your hashed passphrase (S2K) by a symmetric algorithm (typically CAST5)
  - ☐ subkeys
- ☐ **Key Distribution**
  - ☐ Key Distribution Problem: How do I get other people's keys? It would be inconvenient to ask them to send them before communicating
  - ☐ You can still do that, and import them
  - ☐ But, a key directory — keyservers — are the trick
    - ☐ http or pgp access
  - ☐ pgp.mit.edu & others — synchronize with eachother, can be updated but not really deleted
    - ☐ need to revoke your key
      - ☐ key compromise, expiry, etc.
      - ☐ how to provide continuity
  - ☐ screenshots — woodrow, jis, etc.
- ☐ **Trust in PGP**
  - ☐ Authentication Problem: How do you know this is my key, and not someone posing as me to MITM me?
  - ☐ Abstract notions
    - ☐ validity — does a key really belong to the person named on it?
    - ☐ trust — do you trust the honesty and judgement of the keyholder to claim that others' keys are valid?
      - ☐ GPG it's 'ownertrust'
  - ☐ X.509 — MIT Certificates
    - ☐ absolute trust of the root, everyone's certificate in the system that isn't expired is valid
  - ☐ PGP — Trust as an individual decision
    - ☐ Directly certify (sign) some keys
    - ☐ Control who you trust to verify others
  - ☐ The Web of trust
    - ☐ Authentication is difficult if you only trust verified friends
      - ☐ By trusting friends, you can treat their signatures as "probably" valid
        - ☐ You might want confirmation — GPG defaults to 1 completely trusted or 3 marginally trusted to view a key as (transitively) valid
        - ☐ You might be able to find a "chain of trust" between multiple people
        - ☐ Most benefit comes from publicity of web of trust, but also problems of inference of association
  - ☐ The Strong Set
    - ☐ The largest strongly connected graph on **public** keyservers — there is a bidirectional path from A to B for every pair (A,B) in the set
  - ☐ What does it mean to sign a key
    - ☐ minimum expectations vary —
    - ☐ explicit signing policies (particularly anal: http://www.nieveler.org/PGP/pgp.htm)

Topic

- ☐ what is actually being signed?
- ☐ degrees of verification
  - ☐ 0x10: Generic certification of a User ID and Public-Key packet.
    The issuer of this certification does not make any particular assertion as to how well the certifier has checked that the owner of the key is in fact the person described by the User ID.
  - ☐ 0x11: Persona certification of a User ID and Public-Key packet.
    The issuer of this certification has not done any verification of the claim that the owner of this key is the User ID specified.
  - ☐ 0x12: Casual certification of a User ID and Public-Key packet.
    The issuer of this certification has done some casual verification of the claim of identity.
  - ☐ 0x13: Positive certification of a User ID and Public-Key packet.
    The issuer of this certification has done substantial verification of the claim of identity.
- ☐ Methods of signing
  - ☐ individuals meeting up
  - ☐ keysigning party (HOWTO)
- ☐ How to sign a key
  - ☐ Bring paper with UIDs and key fingerprint — probably not a laptop
  - ☐ Require whatever you feel you need to verify someone's identity as matching their key UIDs — a valid photo ID, but it depends on your personal signing policy (just like a certificate authority)
  - ☐ Go home and get public key from keysever, match fingerprint, and sign the key
  - ☐ Export their signed public key and encrypt it with their public key to verify their control over their key, mail it back to one of their addresses — question of politeness
  - ☐ Other tricks to verify control of mail addresses in uid, etc.
  - ☐ Problem: not everyone has the same standards, but this is where trust comes in
- ☐ Changing your mind: signature deletion & revocation
  - ☐ generate a revocation certificate and upload it

- ☐ **What PGP is good for/where it's used**
  - ☐ **Encrypting and decrypting/signing files & email**
    - ☐ email a password or other sensitive data
    - ☐ keep your own files private
    - ☐ Things that you sign to CYA?
  - ☐ **Debian & Ubuntu & Debathena**
    - ☐ uploading packages
    - ☐ securing APT
  - ☐ **Signing your code in a repository — git-tag**
  - ☐ **Signing software releases — better than md5 or sha1 because it provides authentication too**
- ☐ **In all these cases where you desire encryption, consider your threat model**
  - ☐ **Technical Threats**
    - ☐ Assume that the GPG source wasn't tampered with
      - ☐ You can inspect the source of major implementations
    - ☐ Assume that your GPG binaries haven't been tampered with
    - ☐ Assume your machine is trusted and files you might depend on can't easily be modified
  - ☐ **Other threats**
    - ☐ Rubber hose cryptanalysis

Topic

- ☐ Government supercomputers?
- ☐ Inference about your relationships
- ☐ **Think about your threat model and make decisions accordingly**

- ☐ **GPG, a PGP implementation**
  - ☐ **Implementations**
    - ☐ GPG is popular, free, OpenPGP compliant (1.4 vs. 2)
    - ☐ PGP Corp — non-free, desktop software, business-oriented
    - ☐ Graphical frontends for gpg (kgpg, gnome-gpg, firegpg)
    - ☐ MUA/Mail clients (Enigmail, GPGMail, etc.)
    - ☐ apt-cache search "gpglpgplgnupg" (though there's some overlap)
  - ☐ **GPG**
    - ☐ a complex program with lots of options — you'd do well to read the manpage: man gpg | wc -l >> 2950
  - ☐ **GPG Files**
    - ☐ ~/.gnupg/
      - ☐ pubring.gpg & secring.gpg — default public & secret keyrings
      - ☐ gpg.conf — configuration options (like in manpage, without leading "--")
      - ☐ trustdb.gpg — calculated ownertrusts based on web of trust
      - ☐ WARNINGS/CAVEATS:
        - ☐ Trust storage in GPG, etc. has no integrity protection
  - ☐ **GPG Commands**
    - ☐ gpg -s Sign
    - ☐ --clearsign
    - ☐ --detach-sign
    - ☐ -e Encrypt -r Recipient
    - ☐ -c Symmetric
    - ☐ -d Decrypt
    - ☐ -a Armor
    - ☐ -o Output (or > Output)
  - ☐ **GPG Key Commands**
    - ☐ --export
    - ☐ --import
    - ☐ --list-keys
    - ☐ --check-sigs
    - ☐ --fingerprint
    - ☐ --edit-key
  - ☐ **Referring to keys**
    - ☐ KeyID: last 4 bytes of fingerprint (0x optional)
    - ☐ Full fingerprint — hopefully unique
    - ☐ Exact UID Match (=)
    - ☐ Email match (<a@b.c>)
    - ☐ All words (+word1 word2 ...)
    - ☐ Substring (default)
- ☐ **Setting up & using GPG**
  - ☐ **My motivation here:**

Topic

- ☐ Lots of tutorials about getting started by running default gen-key and uploading keys to a keysever — that's it
  - ☐ What happens if you forget your password?
  - ☐ How do you backup your keys?
  - ☐ Should your secret key be on an internet-connected computer?
- ☐ This may not be an ideal setup to start with, and you never think about your threat model

- ☐ **Things to think about**
  - ☐ Many of these things are tradeoffs of convenience and security
  - ☐ Expiry dates
  - ☐ Key size & hash/encryption algorithm choice
    - ☐ Key longevity
    - ☐ compatibility with OpenPGP and PGP $VERSION
    - ☐ Forward secrecy & attestation given technical advancement
  - ☐ Private key storage
  - ☐ Use across multiple machines
  - ☐ Backup
- ☐ **Getting Started — Set Things Up Right in gpg.conf (use this file almost everywhere)**
  - ☐ Choose better defaults (url: http://csrc.nist.gov/groups/ST/hash/statement.html)
    - ☐ cert-digest-algo SHA256
    - ☐ personal-cipher-preferences AES256 AES192 AES CAST5
    - ☐ personal-digest-preferences SHA512 SHA384 SHA256 SHA224
    - ☐ personal-compress-preferences ZLIB BZIP2 ZIP Uncompressed
    - ☐ default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 ZLIB BZIP2 ZIP Uncompressed
  - ☐ Multiple keyrings to stay organized:
    - ☐ no-default-keyring
      primary-keyring default.gpg
      keyring signed-by-<ID>.gpg
      keyring username-<ID>.pub.gpg
      secret-keyring username-<ID>.sec.gpg **(or subsec)**
- ☐ **Generating keys**
  - ☐ Subkey approach (URL: http://tjl73.altervista.org/secure_keygen/en/index.html, http://fortytwo.ch/gpg/subkeys, etc.)
    - ☐ Large, signing-only key that never changes and is maintained securely
    - ☐ Smaller, sign and encrypt subkeys that are used day-to-day without risk to primary key
    - ☐ Allow subkeys to be changed without throwing out all of my signatures
    - ☐ Subkeys can't sign other keys
  - ☐ On a disconnected trusted laptop, on a separate volume (USB key, etc.) --homedir ** also can use PGPcard**
    - ☐ generate primary sign-only key (4096R) and sub signing+encryption (2048R and 2048g)
    - ☐ add UIDs
    - ☐ generate primary key revocation certificate
    - ☐ --export public.gpg and --export-subkeys subsec.gpg
    - ☐ copy public & subkeys to other volume (i.e. ~/.gnupg on laptop) and import
- ☐ **Backing up keys — do this after creation**
  - ☐ Backup KRC, public & private (full keyring) to CD
  - ☐ Paperkeys for secret and subsecret keys

Topic

- ☐ If you want, you could base64 encode the KRC
- ☐ store in a very safe place — secret keys are still encrypted, but KRC is "armed"

- ☐ **Signing new keys**
- ☐ **Lifecycle — Retiring old keys & rolling over new keys**

- ☐ **Doing more with PGP & GPG**
  - ☐ **utilities**
    - ☐ pgpdump
    - ☐ gpgsplit
  - ☐ **packages**
    - ☐ signing-party — caff and others
  - ☐ **keysigning party!**