brief notes for crypto discussion, 22 Sept 02.

these notes are not necessarily intelligible to anyone but me, but they are made available in case someone finds them useful.

What is "crypto"?

Paul Garrett[1] (*Making, Breaking Codes: An Introduction to Cryptology*) defines cryptosystem as "a procedure to render messages unintelligible except to the authorized recipient."

Uses: keeping secrets, authentication
Also oblivious transfer, zero-knowledge proofs, bit commit, other things we're not gonna care about.

Why do we want this? Keeping secrets is good. SMTP traffic is really easy to read, if you're an ISP or the government, and possible if you're someone else.

Envelope analogy. There exist messages that you wouldn't want to write on a postcard so that they can be read by everyone who handles them in transit. However, if everyone used postcards (analogous to current situation where everybody sends unencrypted email), envelopes would be suspicious. But, they are not necessarily suspicious—and encryption doesn't have to be either.

Continuing the envelope analogy: envelopes do not obscure who you communicate with or the fact that you are communicating; they only obscure the content of the communication. Notably, for email, the subject line is not encrypted. So, it may make sense to always use the same subject line on all your encrypted email. (I write 'hey').

Garrett again, defining several terms we'll use: "The **encryption** process, performed by the sender, is intended to render the message unintelligible to any eveasdropper or interceptor of the encrypted message. The **decryption** process is conducted by the legitimate intended receiver, recovering the original message (the **plaintext**) from the obscured version (the **ciphertext**). Paraphrasing: it is intended to be significantly harder for anyone evaesdropping or intercepting the ciphertext to recover the plaintext than for the intended recipient to decrypt it. In symmetric cryptosystems, at least, this is accomplished by having the sender and receiver **share a secret**, called the **key**. Not knowing the key should be sufficient to prevent an eavesropper from decrypting the cyphertext."

Easy ciphers: Caesar, monoalphabetic substitution. examples if time.

Problem: really fucking easy to break. Witness publication of "cryptograms" in newspapers, Poe's "The Gold Bug".

One-time pad. Totally unbreakable crypto for which everybody in the room can understand all the math.

[1] not to be confused with Paul Garrin

Garrett: OTP, when used prpoerly, has **perfect security**, which he defines as "an attacker who intercepts the ciphertext has no more infomration about the plaintext than if they didn't even have the ciphertext." ∃ a proof of this.

```
example encryption:
  L  E  A  R  N     C  R  Y  P  T  O   message
 12  5  1 18 14     3 18 25 16 20 15   convert to numbers
  6 14 15  5 10    23 11  1 23 26 24   add random OTP
 18 19 16 23 24    26  3 26 13 20 13   encoded message is indistinguishable
  R  S  P  W  X     Z  C  Z  L  T  L   from random numbers

decryption:
 18 19 16 23 24    26  3 26 13 20 13   recv encrypted string
  6 14 15  5 10    23 11  1 23 26 24   subtract random OTP
 12  5  1 18 14     3 18 25 16 20 15   result of subtraction
  L  E  A  R  N     C  R  Y  P  T  O   convert to letters => message!
```

Very cumbersome: cannot reuse OTP, else encryption can be broken. Lack of a good RNG: during WWII, used to have sec'ys draw balls out of a bin. But turned out to be a bias: people were looking at the balls as they drew them out, and tended (subconsciously) to favor letters that are common in English. This weakness could, given a large enough body of text, be exploited by the enemy.

Modern symmetric ("private-key") crypto as pseudo-OTP. [This is technically not always quite accurate, but more than good enough for our purposes.] So, rather than having to exchange CDs full of random numbers, we only have to exchange one key (define) per person with which we want to communicate—a huge savings! And, as far as we know, modern 128-bit cryptosystems still need more time to break than the age of the universe. Not quite as good as impossible, but still good enough for me `:-)`

Still, a problem: what if I want to communicate with someone I've never met in person? More generally, symmetric-key crypto assumes that there is some secure communication channel by which we can exchange a key. But if we have such a means, why wouldn't we just use it to exchange the message itself?

Solution! Asymmetric ("public-key") crypto. These are new—first developed around 1975. This relies on the mathematical notion of a **one-way function**, or **trapdoor**, which *Nature* describes thus: "... one of the central components of cryptography today. A one-way function is like an answer to which the original question is very hard to guess. Answering the question 'how many months in a year?' is easy. But working out what question elicited the answer '12' is almost impossible. It could have been how many eggs in a dozen, or how many disciples did Jesus have, for example."

So, now I have to explain public-key crypto.

The idea is that rather than one key per set of people who comminucate privately, we have one keypair per person. This is much better (in CS lingo, $\mathcal{O}(n)$ total keys, rather than $\mathcal{O}(n^2)$).

How does this work? Each person has a **private key** and a **public key**. The former is kept secret and must never be divulged to anybody. The latter is shared with the whole world. Mine is on my website, and there are several websites (e.g., `wwwkeys.pgp.net`, `www.keyserver.net`, `pgp.mit.edu`) where people can go and post their public keys so that other people will be able to find them easily.

It is perhaps worthwhile to think of the public & private keys as two halves of the same key, rather than as two completely separate entities.

Garrett, perhaps a bit obtusely for the non-technical audience: "The alternative terminology 'asymmetric' reveals a little of what these cuphers are about. First, by contrast, all classical ciphers, as well as certain contemporary ciphers such as DES and AES, are *symmetric* in the sense that knowledge of the decryption key is equivalent to, or often exactly equal to, knowledge of the encryption key. By contrast, knowledge of encryption and decryption keys for asymmetric ciphers are not equivalent (by any feasible computation). That is, one or the other of the encryption/decryption keys might be kept secret, while the other is made public, allowing many different people to encrypt, but only one person to decrypt. This asymmetry is the crucial point in understanding the new possibilities that arise once we know about public-key ciphers."

So, let's say Alice wants to send a secret message to Bob. First, she locates Bob's public key. She (or, more likely, her software) takes the secret message and the public key, and creates some encrypted message. She then takes this ciphertext—which even she cannot read—and sends it to Bob. Bob opens up the message, gets out his private key, and decrypts the message.

Pause for questions. Make sure people understand. Demonstration time: show the software. Reasonable key lengths. How to export your public key, and send it to people. How to encrypt things. How to decrypt things.

If time: one nifty thing we can do is **digital signatures**. These work in some sense the opposite of encryption: one needs a *private key* to create them, and a *public key* to view (verify) them. This provides proof (subject, of course, to certain unproven mathematical assumptions) that the sender of the message is who ey says ey is.

If audience is composed of quick learners: **web of trust**. Blah.