Users and security

- Authentication
  - Making sure a user is who they say they are
  - ...on every request!
- Authorization
  - Making sure a user can only get to information they are supposed to see
  - Making sure a user can only perform actions they are supposed to

Authentication

- Username/password combination
  - Most basic level of authentication
    1. Get username/password from user
    2. Verify against username/password stored in database
  - Security concerns
    - Passwords stolen from database
    - Passwords intercepted in transit
    - Passwords sent to a rogue server
    - Password strength
    - Social engineering

Database-level security

- The obvious stuff
  - Deny everything, allow what is necessary
    - Isolate, firewall
- Storing passwords (and other confidential information)
  - Don’t unless you have to!
  - Hash the password and store that instead
    - One-way, cannot recover original
    - No one can get the actual passwords from the db
  - For verification, hash the incoming password and compare to the stored hash

Hashing

- Vulnerable to brute-force attacks
  - Attacker gets the hash
  - Attacker guesses passwords and hashes them until one matches
  - Not as hard as it sounds
    - Faster hardware, weak passwords, lookup tables
- MD5, SHA1
  - Commonly available, out of date
    - Public tables exist to crack any MD5 hash for passwords up to 8 characters
- SHA256, SHA521, BLOWFISH
  - Much better options, designed to run slowly
    - But still can be brute-forced

Hashing with salts

- Make brute-force less efficient, leverage complexity
  - Longer passwords
  - Slower hashing algorithms
  - Larger space of possible hashes
- Salting
  - Concatenate a random string to each password before hashing
  - Store the random string (not secret) with the hash
  - Defeats look-up tables that pre-calculate hashes

Example Hash

```
$2a$10$KssILxWNR6k62B7yiX0GAe2Q7wwHlrzhF3LqtVvpyvHZf0MwvNfVu
```

- Bcrypt MCF format:
  - $<type>$<cost>$<salt><hash>
  - Type identifies the algorithm:
    - 1 = md5
    - 2, 2a, 2y = blowfish variants
  - Cost is the number of iterations to run (making it slower)
  - Salt is 22 characters, hash is 31

Encryption

- Two-way encryption
  - Allows data to be encrypted and decrypted
  - AES is the standard
    - Implemented in MySQL and in PHP (Mcrypt)
  - Relies on a secure key
- If the key is compromised, all encrypted data can be decrypted!
  - Again, only use if recovery is absolutely necessary (credit cards, soc sec #s, etc)

Use tested code

- Don’t roll your own security code!
  - Too easy to make errors
  - Especially with complex systems like AES
- Use an established library
  - Already well tested
  - Verified by people who actually understand the math
  - PHPass
  - MySQL AES_ENCRYPT/AES_DECRYPT

Network-level security

- What’s going over the wire?
  - Data from client to server
    - Passwords, for instance
  - Data back from server to client
    - URL query strings
    - Hidden form fields
  - Data from web app to database?
    - Where does encryption happen?

Encrypted network traffic

- Everything on the internet wires is public!
  - Too many points of failure to control
  - You *must* encrypt any private data
- A secret message for you:

BDB FKHHVH

Encrypted network traffic

- Everything on the internet wires is public!
  - Too many points of failure to control
  - You *must* encrypt any private data
- Encrypting a conversation requires *a priori* information
  - You must have a trusted, private conversation first
- Solution: asymmetric encryption

Asymmetric encryption

- Public key/private key
  - Public key is given out to everyone
  - Private key is kept secret
- To send a private message:
  - Encrypt with the public key
  - Can only be decrypted with the private key
  - Message is private
- To receive a message:
  - Encrypted with private key
  - Can be decrypted by anyone with the public key
  - Verifies that it was sent by the private key holder

How does it work?

- Math competition!

How does it work?

- Math competition!
  - 71 and 37 are prime numbers
  - What is 71 * 37?

How does it work?

- Math competition!
  - 158987 is the product of two prime numbers
  - What are those prime numbers?

How does it work?

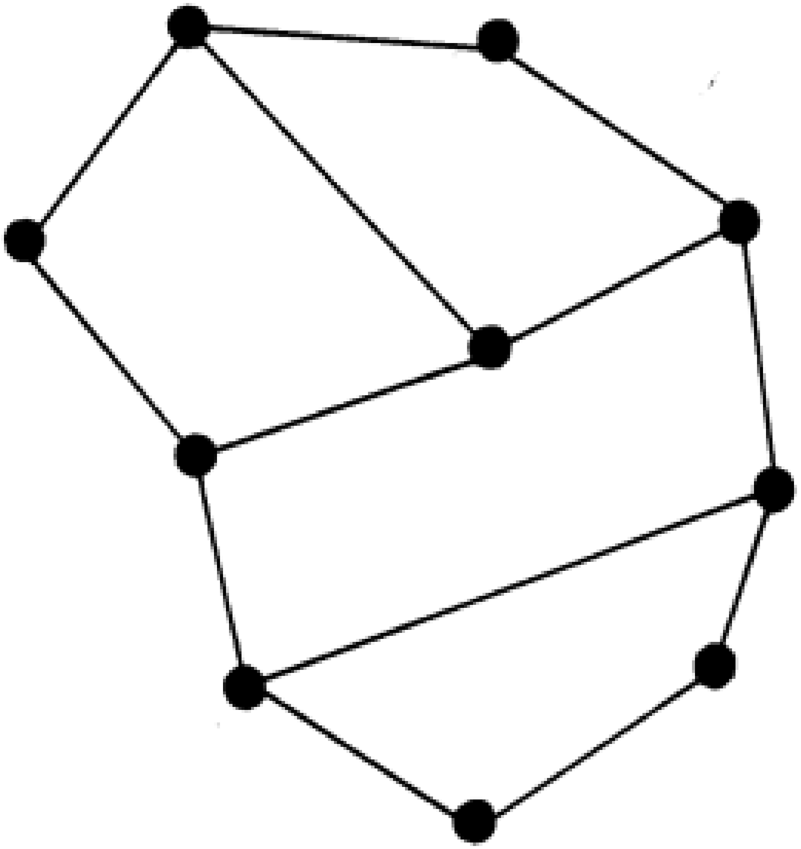
- Math competition!
  - 158987 is the product of two prime numbers
  - What are those prime numbers?
    - (919 and 173)

How does it work?

- Based on a problem that is:
  - Very hard to solve in one direction
  - Easy to solve in the other direction
- Factoring prime numbers
  - Find the largest prime factors of 293492849128492911
    - Very hard to solve, a lot of guessing and checking
  - But given the factors, easy to generate the original number

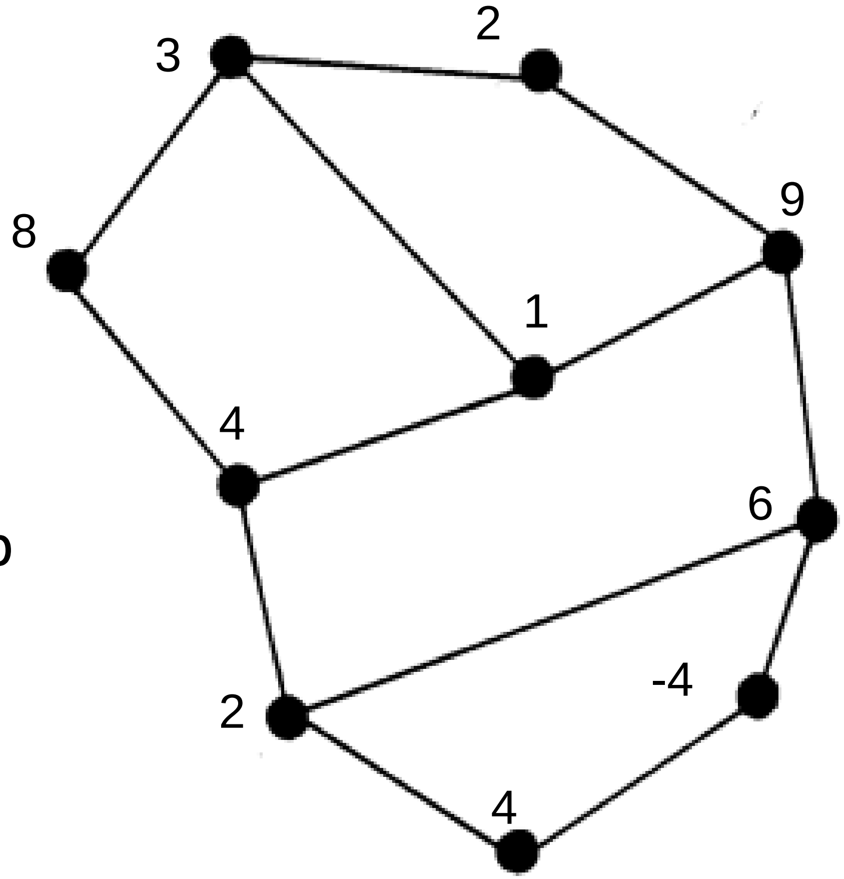
Public-key encryption

- This map is my public key (everyone can see)
- To send me a secret number:
  - Draw out that map



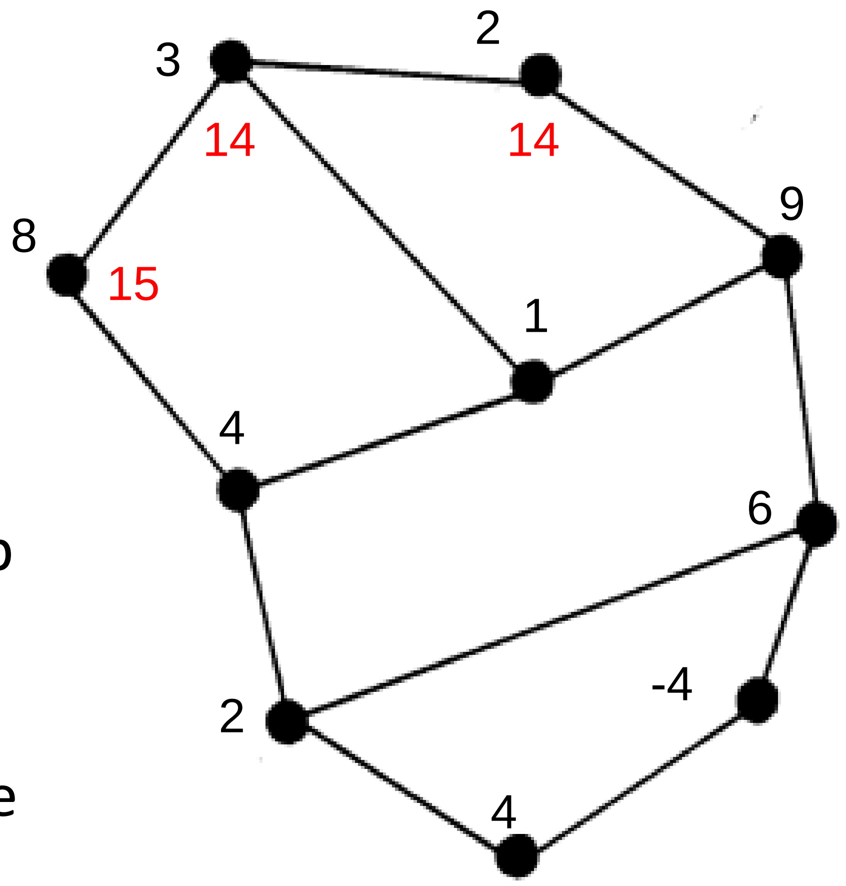
Public-key encryption

- This map is my public key (everyone can see)
- To send me a secret number:
  - Draw out that map
  - Put numbers on each corner (can be negative) that add up to the number you chose



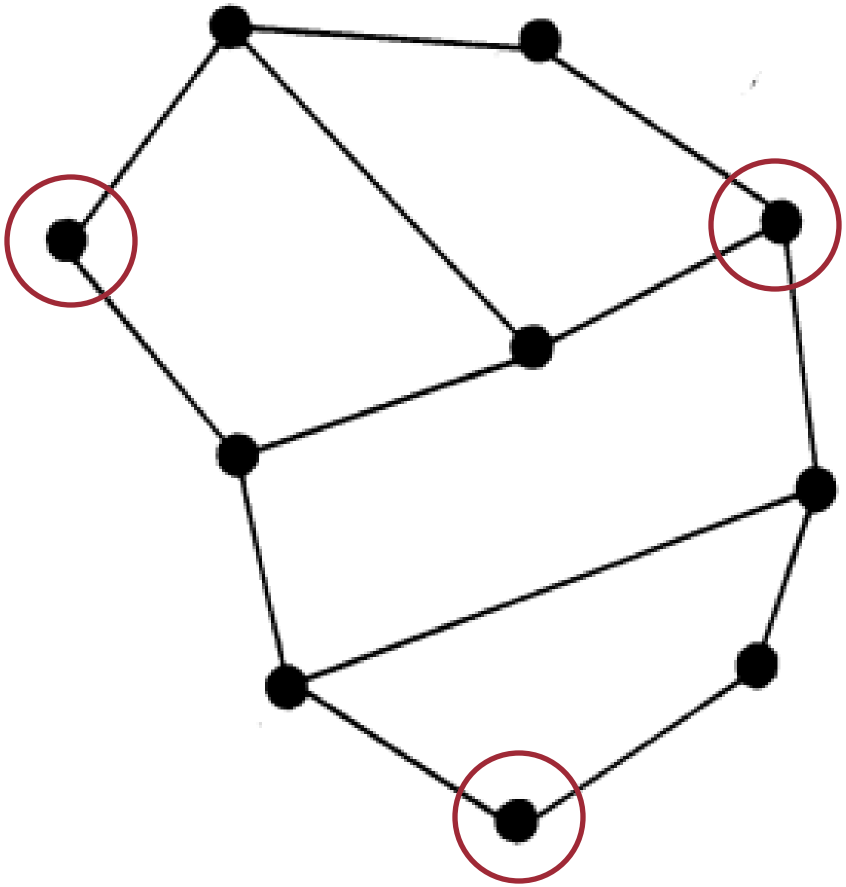
Public-key encryption

- This map is my public key (everyone can see)
- To send me a secret number:
  - Draw out that map
  - Put numbers on each corner (can be negative) that add up to the number you chose
  - For each corner, add the number on that corner to the numbers on all connected corners
  - Tell me those totals only



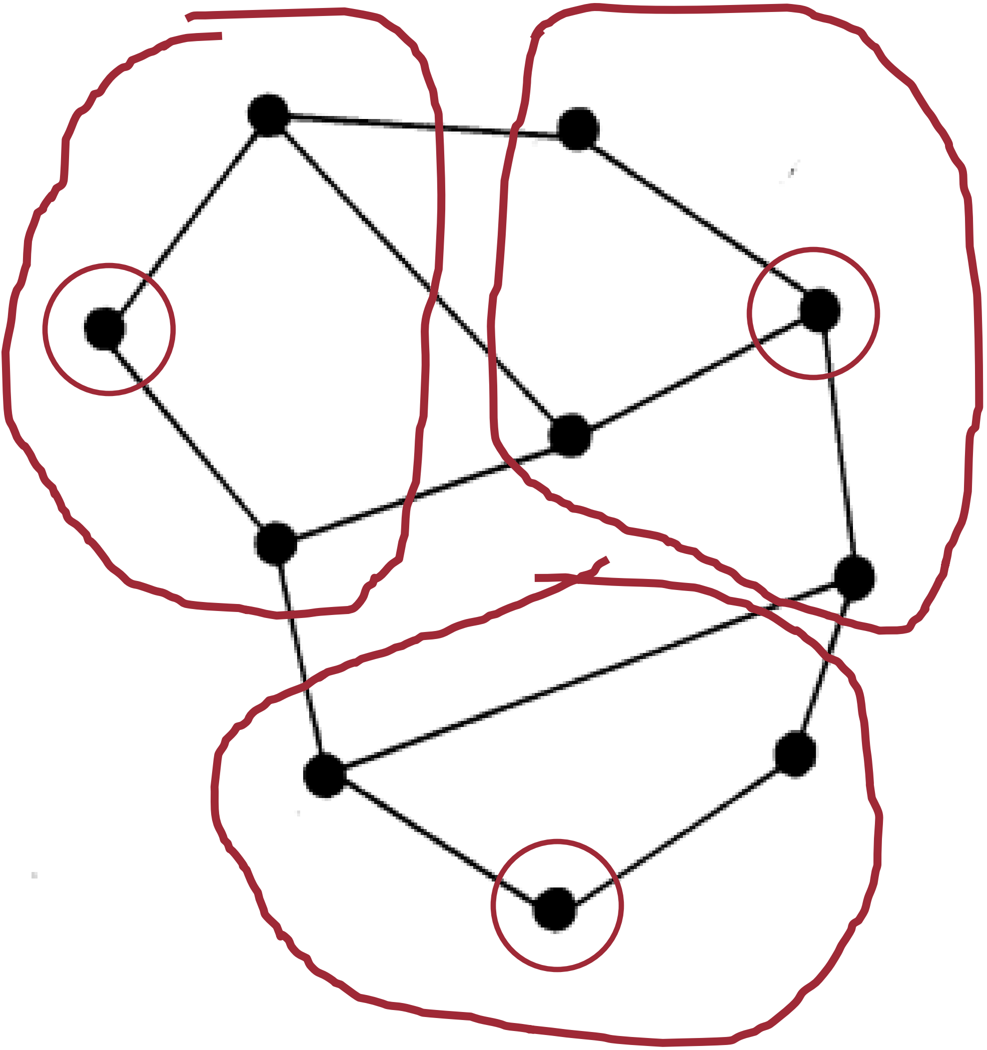
Public-key encryption

- This map is my private key



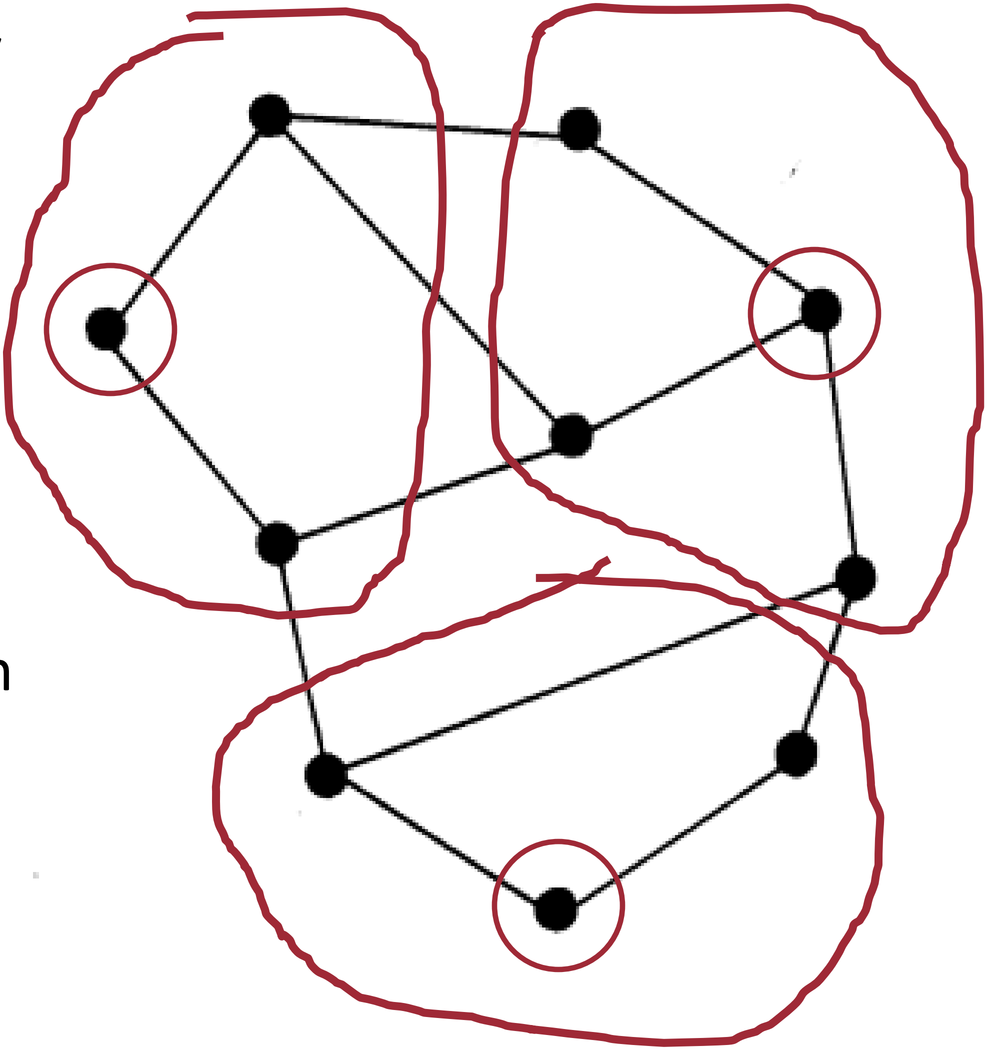
Public-key encryption

- This map is my private key
- Marked intersections
  - Indicate nodes that separate the graph
  - The sum of those nodes is the original number



Public-key encryption

- This map is my private key
- Marked intersections
  - Indicate nodes that separate the graph
  - The sum of those nodes is the original number
  - Finding the separating intersections on a map with 100 nodes is a hard problem
  - Factoring primes is harder



Encrypted network traffic

- Transport Layer Security (TLS)
  - Encryption of HTTP traffic
  - Used to be called SSL
  - Pretty universally supported
- Starting a private (encrypted) conversation
  1. Get the public key of the server
  2. Encrypt a message with the public key and send
     - Typically parameters for further encryption
  3. Only the server can decrypt it!

Encrypted network traffic

- Transport Layer Security (TLS)
  - Encryption of HTTP traffic
  - Used to be called SSL
  - Pretty universally supported
- Starting a private (encrypted) conversation
  1. Get the public key of the server
  2. Encrypt a message with the public key and send
     - Typically parameters for further encryption
  3. Only the server can decrypt it!

  (See any problems?)

Encrypted Network Traffic

- Anyone can claim to be the server
  - Man-in-the-middle attack
  - Send you bogus public key
- Solution?
  - Certificate authorities
    - Ask CA to verify public key actually belongs to server

Encrypted Network Traffic

- Anyone can claim to be the server
  - Man-in-the-middle attack
  - Send you bogus public key
- Solution?
  - Certificate authorities
    - Known reliable source
    - Ask CA to verify public key actually belongs to server

    (See any problems?)

Encrypted Network Traffic

- Anyone can claim to be the server
  - Man-in-the-middle attack
  - Send you bogus public key
- Solution?
  - Certificate authorities
    - Known reliable source
    - Ask CA to verify public key actually belongs to server

    (See any problems?)
    - Man-in-the-middle attack
    - Send you bogus verification
    - Solution?

Encrypted Network Traffic

- Anyone can claim to be the server
  - Man-in-the-middle attack
  - Send you bogus public key
- Solution?
  - Certificate authorities
    - Known reliable source
    - Ask CA to verify public key actually belongs to server

    (See any problems?)
    - Man-in-the-middle attack
    - Send you bogus verification
    - Solution?
      - Web browser has public key for known CAs *a priori*

Back to authentication

- Security concerns
  - Passwords stolen from database
  - Passwords intercepted in transit
  - Passwords sent to a rogue server
    - Certificate Authorities
  - Password strength
  - Social engineering
- Session IDs
  - Login credentials not resent with every request
  - Encryption to prevent session hijacking
  - Rotating session IDs