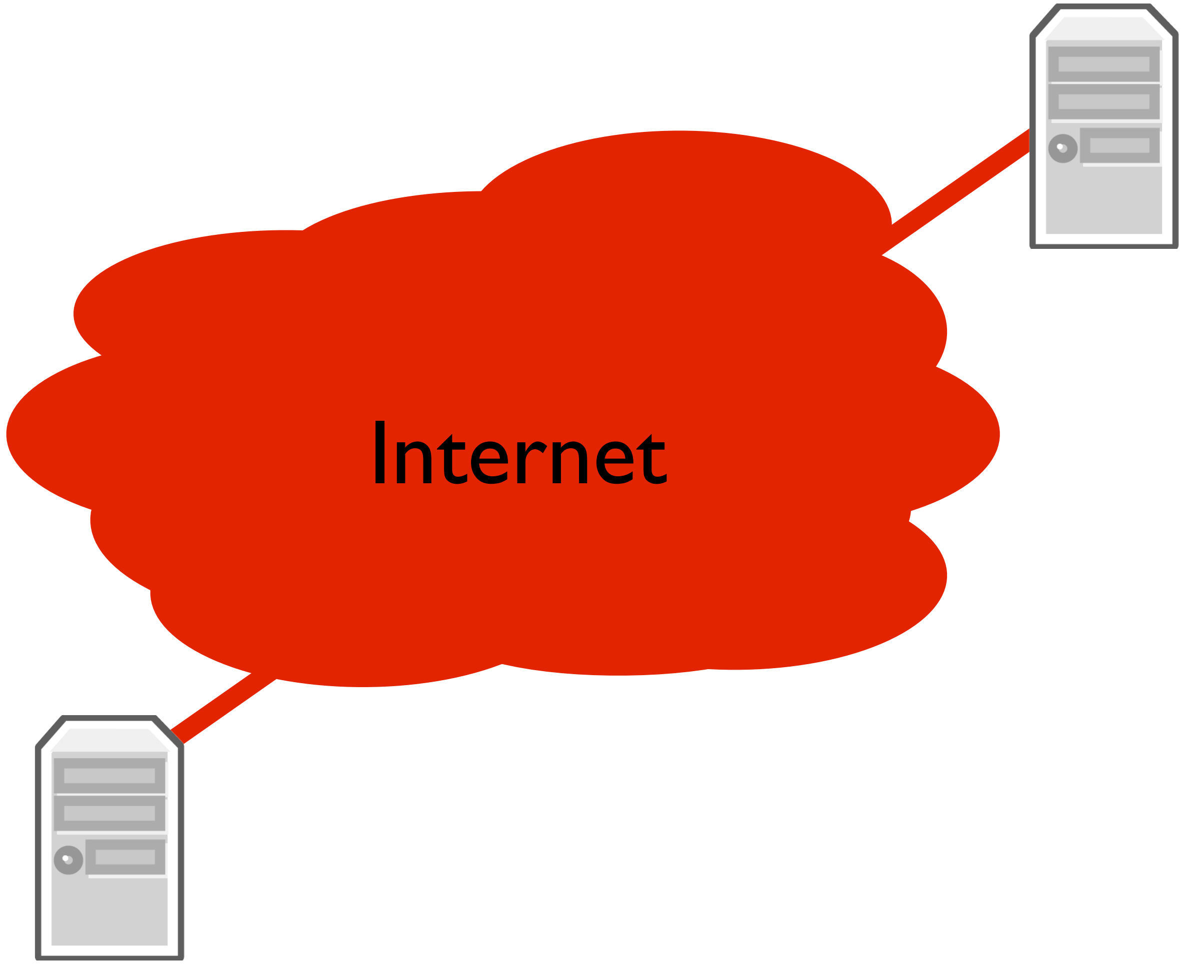
Security Principles

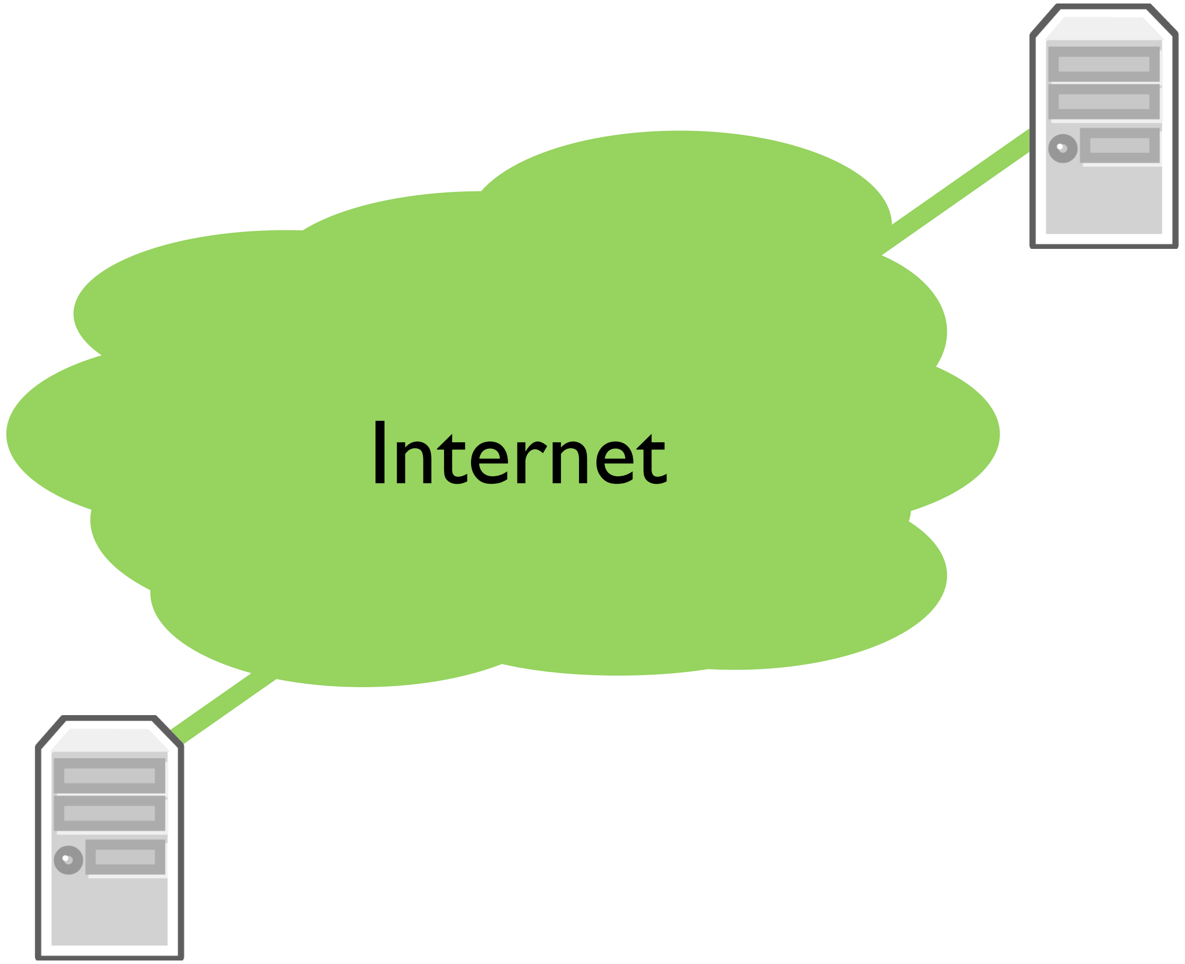
Confidentiality, Integrity, and Availability

Basic Problem



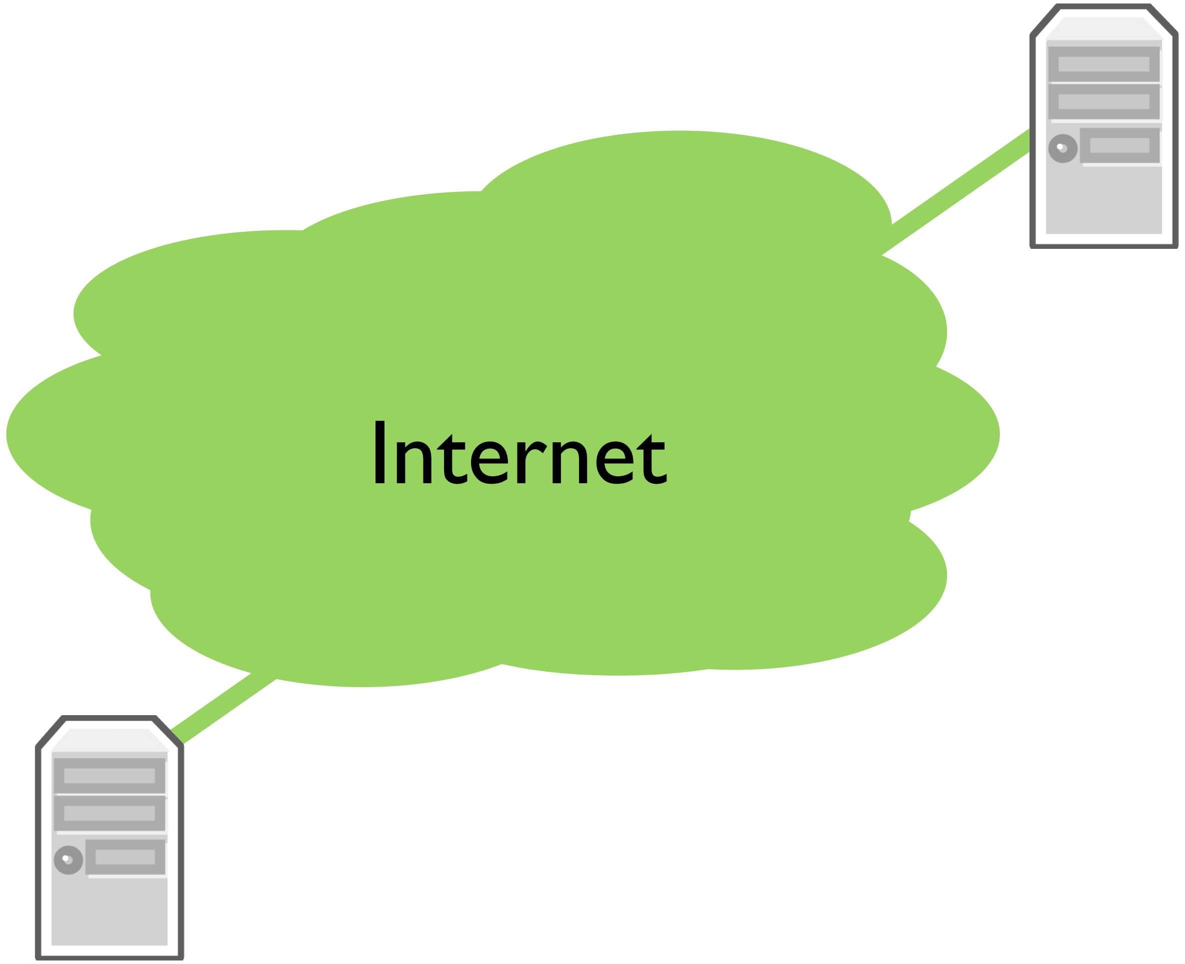
- To first approximation, attackers control the network
  - Can snoop, replay, suppress, send
- How do we defend against this?
  - Communicate securely despite insecure networks -- *cryptography*
  - Secure small parts of network despite insecurity of wider network
  - Design systems to scale well in response to attacks
- Two approaches: cryptography and scalable system design

Cryptography



- A set of mathematical principles and ideas for securing communication
- Be careful: easy to mess up!
  - Often misused!
  - Need to understand what it guarantees and what it doesn't
- How cryptography helps
  - Confidentiality: we can communicate privately (encryption)
  - Integrity: protect from tampering (hashes, signatures, MACs)
  - Authenticity: you are whom you say you are (certificates, MACs, signatures)

Cryptography



- A set of mathematical principles and ideas for securing communication
- <u>Be careful: easy to mess up!</u>
  - <u>Often misused!</u>
  - <u>Need to understand what it guarantees and what it doesn't</u>
- How cryptography helps
  - Confidentiality: we can communicate privately (encryption)
  - Integrity: protect from tampering (hashes, signatures, MACs)
  - Authenticity: you are whom you say you are (certificates, MACs, signatures)

Confidentiality

- I want to tell you something secretly, so no-one else knows what I said
  - "My credit card number is...."
- Perfect confidentiality: one-time pad
  - You and I share a perfectly random *key* of zeroes and ones, *K*, nobody else has it
  - I XOR my message *M* with *K*, producing *C*, send *C* to you ($C = M \oplus K$)
  - You XOR *C* with *K*, you have reconstructed *M* ($M = C \oplus K$)
- Advantages: informationally theoretic secure and fast
  - Given any *C*, any *M* is equally likely
- Disadvantage: need a *K* as long as all data I might ever send

- Ways to exchange a small *K* such that there are 2^k possible Cs

Integrity

- I want to make sure you received my message unchanged/untampered
  - "I recalculated his grade and it is a C...", program to run on your computer
- I want to make sure you sent the message
  - Nick says: "I said it was alright if he handed in the assignment late"
- Cryptographic hash: *H(M)*
  - Turn arbitrary length input into fixed length hash
  - Collision-resistance: given $x \neq y$, intractable to find $H(x) = H(y)$
- Message authentication code: *MAC(M,K)*
  - Use key *K* to generate *MAC(M,K)*, use *K* to check *MAC(M,K)*
  - Intractable to generate *MAC(M,K)* unless you have *K*

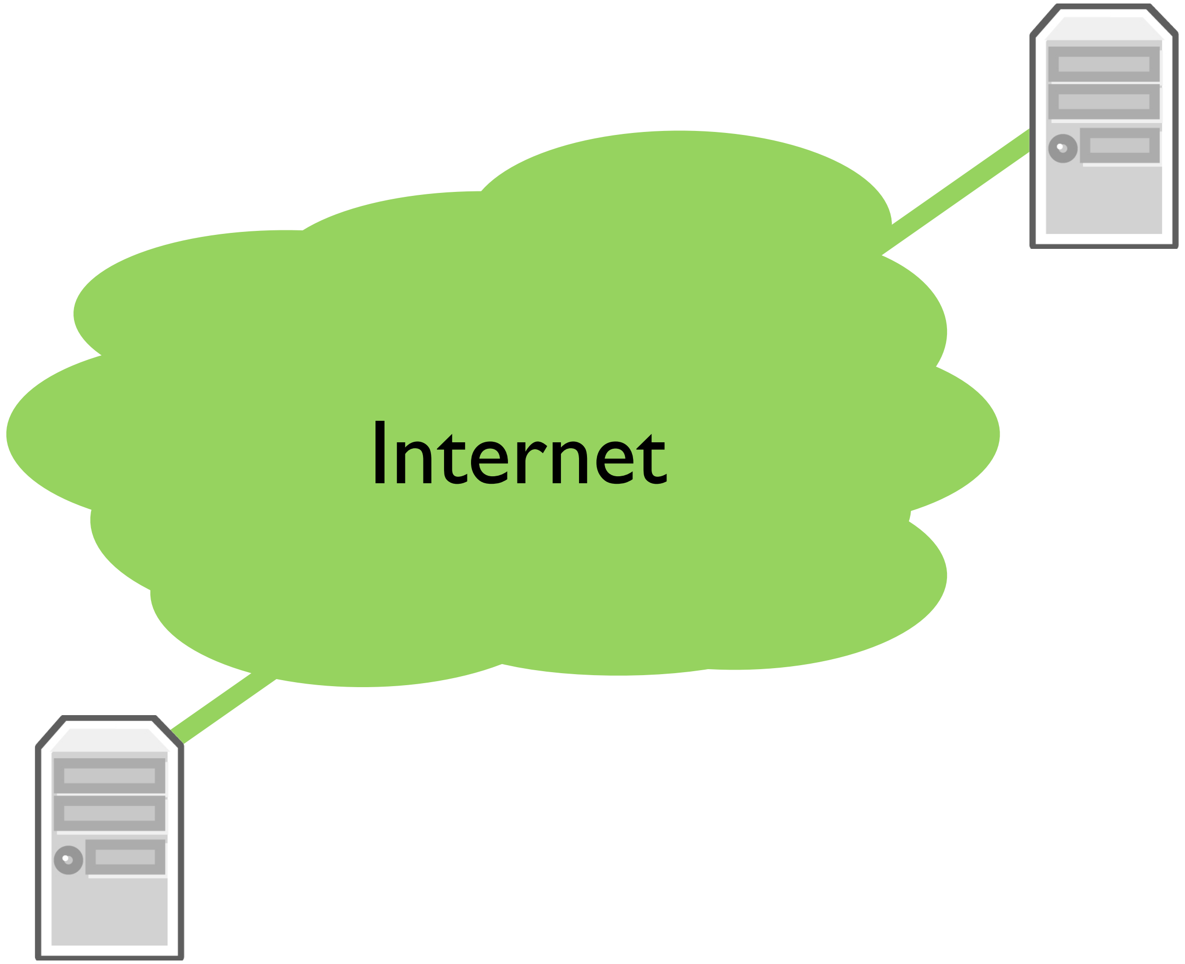
Authenticity

- I want to be sure you are whom you say you are
  - "This is the provost... I need your credit card number."
  - "You should send all of my traffic through this third party"
- We've exchanged a secret *K* beforehand: MAC("This is the...", K)
- If we haven't: chain of of trust
  - We can trust Verisign by design (root of trust)
  - Verisign says "here's a secret for Stanford"
  - Stanford says "here's a secret for the provost"

High Availability Design

- Denial of service (DoS), Distributed Denial of Service (DDoS)
- Many kinds of attacks, many defenses
    - Replication (scale-out)
    - Keeping costs symmetric
    - Upstream filtering (stop letting those pings through, router!)
- Continual arms race: not going to talk much more about it

Cryptography



- A set of mathematical principles and ideas for securing communication
- Be careful: easy to mess up!
  - Often misused!
  - Need to understand what it guarantees and what it doesn't
- How cryptography helps
  - Confidentiality: we can communicate privately (encryption)
  - Integrity: protect from tampering (hashes, signatures, MACs)
  - Authenticity: you are whom you say you are (certificates, MACs, signatures)