Modular Arithmetic and RSA Encryption

Stuart Reges Principal Lecturer University of Washington

### Some basic terminology

- Alice wants to send a secret message to Bob
- Eve is eavesdropping
- Cryptographers tell Alice and Bob how to encode their messages
- Cryptanalysts help Eve to break the code
- Historic battle between the cryptographers and the cryptanalysts that continues today

# **Public Key Encryption**

- Proposed by Diffie, Hellman, Merkle
- First big idea: use a function that cannot be reversed (a humpty dumpty function): Bob tells Alice a function to apply using a public key, and Eve can't compute the inverse
- Second big idea: use asymmetric keys (sender and receiver use different keys): Bob has a private key to compute the inverse
- Primary benefit: doesn't require the sharing of a secret key

# **RSA Encryption**

- Named for Ron Rivest, Adi Shamir, and Leonard Adleman
- Invented in 1977, still the premier approach
- Based on Fermat's Little Theorem: a<sup>p-1</sup>≡1 (mod p) for prime p, gcd(a, p) = 1
- Slight variation:

 $a^{(p-1)(q-1)} \equiv 1 \pmod{pq}$  for distinct primes p and q, gcd(a,pq) = 1

Requires large primes (100+ digit primes)

### Example of RSA

- Pick two primes p and q, compute n = p×q
- Pick two numbers e and d, such that:

 $e \times d = (p-1)(q-1)k + 1$  (for some k)

Publish n and e (public key), encode with:

(original message)<sup>e</sup> mod n

Keep d, p and q secret (private key), decode with:

(encoded message)<sup>d</sup> mod n

#### Why does it work?

 Original message is carried to the e power, then to the d power:

 $(msg^e)^d = msg^{e d}$ 

- Remember how we picked e and d: msg<sup>ed</sup> = msg<sup>(p-1)(q-1)k + 1</sup>
- Apply some simple algebra:
  msg<sup>ed</sup> = (msg<sup>(p-1)(q-1)</sup>)<sup>k</sup> × msg<sup>1</sup>
- Applying Fermat's Little Theorem:
  msg<sup>ed</sup> = (1)<sup>k</sup> × msg<sup>1</sup> = msg