DTTF/NB479: Dszquphsbqiz Day 18

- Announcements:
  - HW4 – DES due Thursday
    - I have installed, or will install: Java, C (gcc), Python.
    - What other languages? Please make appointment to help install required software
  - Thursday: bring textbook and Maple in class.
  - Friday: Ch 3 written exam
  - Term project groups and topics due at end of week 6
    - Use ch 10 – 19 as inspiration
- Today
  - Finish Rijndael
  - RSA concepts
- Questions?

Rijndael/AES

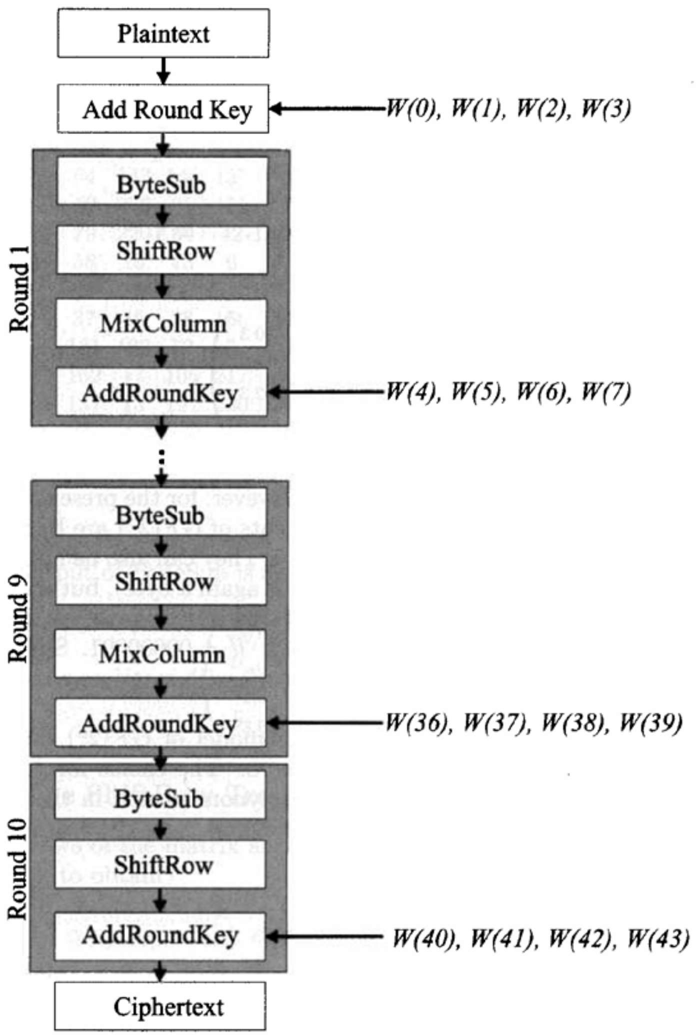

Figure 5.1: The AES-Rijndael Algorithm

- Tie-ins with Galois field, $GF(2^8)$:
  - S-box implements $z = Ax^{-1} + b$ in $GF(2^8)$
  - MixColumn multiplies by a matrix in $GF(2^8)$ to diffuse bits
  - Key schedule (next) uses S-box and powers in $GF(2^8)$
- Wikipedia's visuals

Public-key cryptography is used to send "session" keys for AES to encode messages.

- Do you trust 128-bit encryption now?
- You should, especially when keys are sent using public key cryptography (next)
- Relationship between RSA and AES in http://www.grc.com/securitynow.htm#183. (To get to the point, jump to 51:57; thanks to Matthew Jacobs '09 for reference)

Public-key Cryptography

- Problem: how can I send my AES key without Eve intercepting it?
- Consider a scheme in which everyone publishes a (public) method by which messages can be encrypted and sent to them … but only the publisher can decrypt.
  - Knowing how to encrypt does not reveal how to decrypt!

RSA (Rivest – Shamir – Adelman) relies on the inability to quickly factor products of large primes

For Alice to send a message to Bob.

- Bob chooses primes p,q (large, ~100 digits each)
- He publishes his public key (n,e):
  - $n = pq$
  - e, a large number such that $\gcd(e, (p-1)(q-1)) = 1$

- Alice has a message $m < n$.
  - Otherwise (if $m > n$), break message into chunks $< n$
- Alice sends $c = m^e \pmod{n}$
- Bob computes $c^d \pmod{n} = (m^e)^d = m \pmod{n}$.
- What does he use for d?

Why does decryption work?

- Alice – (m) → Bob
- Bob's key:
  - n = pq
  - e: gcd(e, (p-1)(q-1)) = 1
  - This is so
    $d=e^{-1}$ mod (p-1)(q-1) exists
- Alice sends c = m^e(mod n)
- Bob computes c^d (mod n)
  = $(m^e)^d$ = m (mod n),
  where
  d = e^{-1} (mod (p-1)(q-1)).

- Recall Euler's theorem:

$$m^{\phi(n)} \equiv 1 (\mathrm{mod}\, n)$$

- as long as gcd(m,n) = 1
- So m^{ed} = m (mod n)
  iff ed = 1 (mod ϕ(n))
  = 1 (mod (p-1)(q-1))
- So d = e^{-1} mod (p-1)(q-1)

Toy example

- Alice has (m) → Bob
- Bob's key:
  - $n = pq = (13)(17) = 221$
  - $e = 35$: $\gcd(e, (p-1)(q-1)) = 1$
  - $d=e^{-1} \bmod 192$ exists:
    $d =$ _11__
- m = 20 (letter t)
  - 1-based, so leading 'a' = 1 not ignored
- $c = m^e \pmod{n} =$ 197___
- $c^d \pmod{n} =$ 20__

Issues:

How does Alice compute $20^{35} \pmod{221}$?

Modular exponentiation

Efficiency is O(log e)

How does Bob compute d?

Extended Euclidean alg.

Efficiency is O(log n)

- And why is this secure?
  - Why can't Eve calculate d herself?

The security if RSA lies in the difficulty of factoring large numbers

- Eve knows e, n, and c only
- To find $d = e^{-1} \pmod{\phi(n)}$,
  Eve needs to know $\phi(n) = (p-1)(q-1)$
- If she knows n, she can factor it into p and q to find $\phi(n)$, right?
- That's a big *if*, since n is ~200 digits long in practice!
- Large numbers are **hard** to factor!
  - Can't just test every prime from 1 .. sqrt(n)

The security if RSA lies in the difficulty of factoring large numbers

- $c = m^e \pmod{n}$
- Can Eve just compute e-th root of c?
  - Not since mod n
  - Unless we brute force, but not when n is large!

Is $\phi(n)$ as hard to find as the factors of n?

Claim:

factoring n hard → finding $\phi(n)$ hard

Hint: write n and $\phi(n)$ in terms of p and q.

Next week: finding d is as hard to do as finding the factors of n

So Eve has no shortcuts to factoring!

You will need your computer with Maple on it next class to do a real example of RSA

Today I demo’ed in Matlab.

Unfortunately, Matlab can’t generate large primes quickly.

Maple’s nextprime method works well