Diffie-Hellman

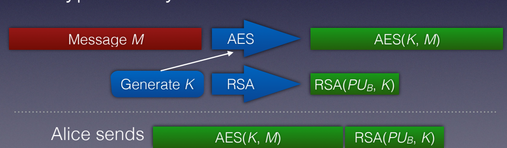
CMSC 426/626 - Fall 2014

Outline

- Key Exchange
- The discrete logarithm problem
- Diffie-Hellman
- Man in the Middle
- Elliptic Curve Cryptography

Key Exchange with RSA

- Alice and Bob want to share a secret key for use with a symmetric algorithm such as AES.
- It is more efficient to encrypt data with AES and encrypt the key with RSA.



Discrete Logarithms

- The security of the Diffie-Hellman algorithm is based on the **discrete logarithm problem**.
  - Let *p* be a prime number
  - An integer *a*, $0 < a < p$, is a **primitive root mod p** if the powers of *a* mod *p* are distinct and consist of all the numbers from 1 to $p - 1$.
  - Given *b*, $0 < b < p$, there is a number *x* such that $b = a^x$ mod *p*.
  - The number *x* is the **discrete logarithm of *b* base *a* mod *p*.**

Dlog Example

- Find the discrete logarithm of 17 base 3 mod 29 ($p = 29$, $a = 3$, $b = 17$)

```
>>> x = 1
>>> while pow(3,x,29) != 17:
...     x = x + 1
...
>>> x
21
>>> pow(3,21,29)
17
```

- What happens if *a* is not primitive? The discrete log of *b* may not exist.
- For large primes *p* finding the discrete logarithm of a number is infeasible.

Diffie-Hellman

System Parameters

q	a large prime
α	primitive root mod q

Alice's Parameters

X_A	Random secret $0 < X_A < q$
Y_A	$\alpha^{X_A} \bmod q$

Bob's Parameters

X_B	Random secret $0 < X_B < q$
Y_B	$\alpha^{X_B} \bmod q$

Alice		Bob
Send Y_A	⟶	Compute $K_B = Y_A^{X_B}$
Compute $K_A = Y_B^{X_A}$	⟵	Send Y_B

$$K_A = K_B$$

Alice and Bob have a shared secret key!

Example

- Use the same values as in the previous example ($p = 29$, $a = 3$, $b = 17$).
- Alice's private value (X_A) is 12.
- Bob's private value (X_B) is 5.

```
>>> Xa = 12
>>> Xb = 5
>>> Ya = pow(3, Xa, 29)
>>> Yb = pow(3, Xb, 29)
>>> # Alice receives Yb and computes Ka
...
>>> Ka = pow(Yb, Xa, 29)
>>> Ka
23
>>> # Bob receives Ya and computes Kb
...
>>> Kb = pow(Ya, Xb, 29)
>>> Kb
23
```

"Real" DH

- In reality, DH is a bit more complicated.
- Large prime p (at least 1024 bits); α generates a subgroup of prime order q (at least 160 bits):

$$\alpha^0 \bmod p$$
$$\alpha^1 \bmod p$$
$$\vdots$$
$$\alpha^{q-1} \bmod p$$
$$\alpha^q \bmod p = \alpha^0 \bmod p$$

Man in the Middle

- Unfortunately, the protocol as described is susceptible to a man-in-the-middle attack (MitM).
- Eve can pretend to be Bob to Alice *and* pretend to be Alice to Bob - all communication flows through Eve!
- Certificates can fix this problem. The CA would sign the public values (e.g. Y_A and Y_B).
- There are other DH-based protocols to prevent MitM.

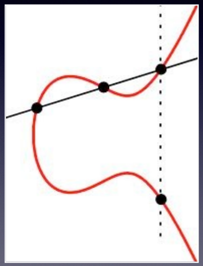
Elliptic Curve Cryptography

- Elliptic curves are a complex mathematical object that can be used in place of mod p arithmetic.
- What that means is that elliptic curves provide us with a finite collection of numbers which we know how to add and for which addition acts as we would expect.
- **Notation:** F_p denotes the set of integers mod p along with addition and multiplication.

Elliptic Curves

- Solutions (x, y) to equations of the form

  $E: y^2 = x^3 + ax + b$
- For cryptography, x and y are integers mod p.
- The addition rule can be derived geometrically.



Addition

- Given points P = (x_P, y_P) and Q = (x_Q, y_Q)
  - -P = $(x_P, -y_P)$
- Sum P + Q = R = (x_R, y_R) is given by
  - $x_R = s^2 - x_P - x_Q$
  - $y_R = s\ (x_P - x_R) - y_P$
- Where $s = (y_P - y_Q) / (x_P - x_Q)$

Rational Points

- $E(\mathbf{F}_p)$ - $\mathbf{F}_p$ *rational points;* P with x and y in $\mathbf{F}_p$
  - $|\ E(\mathbf{F}_p)|$ is finite; cryptographic subgroup?
- Especially interested in p a NIST prime.
  - Generalized Mersenne primes
  - E.g. $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
- $[m]$ P = P + P + ··· + P (m-fold sum)

EC Diffie-Hellman

- Alice and Bob agree on an elliptic curve $E(\mathbf{F}_p)$ and a group generator G of order q
- Alice's public and private values
  - Private random value m_A
  - Public $P_A = [m_A]G$, a point on the curve
- Bob's values: private m_B, public $P_B = [m_B]G$

EC Diffie-Hellman

Alice		Bob
m_A, $P_A=[m_A]G$		m_B, $P_B=[m_B]G$
	P_A →	Compute $[m_B]P_A$
Compute $[m_A]P_B$	← P_B	
$K = x([m_A][m_B]G)$	Shared Secret	$K = x([m_B][m_A]G)$

ECC vs. Classical DH

Classical DH	ECC DH
System parameters α, q	**System parameters** G, $E(F_p)$
Fundamental Operation Exponentiation mod p α^x mod p	**Fundamental Operation** EC Point Addition $[m]\ P$
Parameter Sizes q at least 160 bits p at least 1024 bits	**Parameter Sizes** q at least 160 bits p about the same size as q

- ECC gives comparable security for much smaller parameter sizes!
- There are other ECC algorithms besides ECC DH, but we won't go into those.

Homework is posted on the website.