# Class 3: Digital Signatures

## Schedule

**Monday, September 7**: Check-up 1. This will be a short in-class quiz to test your understanding of the main concepts covered so far. It will cover material from the readings (see Class 2 for details) and classes 1-3.

**Tuesday, September 15** (8:29pm): Problem Set 1 due.

## Signatures

Real-life signatures. Properties:

- Easy to verify
- Forging unlikely
- Hard to repudiate

Digital signatures. Should have same properties, in the absence of legal forces on the internet.

Topics:

- Assymetric cryptography
- Digital signatures
- Elliptic curve cryptography
- Implementation pitfalls

Ordinary (or symmetric) Crypto

- Both parties have to agree on a key

Diffie-Hellman key exchange

- Proposed in 1976
- Establishes a private secret key, unknown to any evesdropper

Discrete Logarithm Problem

- Given $g, y, p$, find $x$ such that $g^x \bmod p = y$

~~Discrete~~ Logarithm Problem

- Easy to solve for real numbers

Random Element out of . . . ?

- What is the range of elements out of which we are randomly selecting?

Mod 5 exponentiation

- "Multiplicative order" is the number of multiplication after which the result repeats. I.e. Multiplicative order of g is n if n is the smallest positive integer such that $g^n = 1$.

Exponent Modulus

- Multiplicative order is at most $p - 1$
- Pick random $x$ such that $0 \leq x < p - 1$
- $g^a g^b \bmod p = g^{a+b} \bmod p = g^{(a+b) \bmod n} \bmod p$

Public-key Cryptography

1. Google announces $g^a$
2. Bob picks random secret $b$, computes $(g^a)^b = g^{ab}$.
3. Bob encrypts message $m$ and sends: $g^b, g^{ab}m$,

Man-in-the-middle (MITM)

- Active adversary can still read everything. We have to know messages are coming from the right person.

## Digital Signatures

Discrete-log based signature

ElGamal Signature Scheme

- Fixed global parameters: $g, p$
- Private key: $a$
- Public key: $g^a \bmod p$
- Signing:
  1. Input: message $m$
  2. Pick random $k$
  3. Compute $r = g^k \bmod p$; $s = (m - ar)k^{-1} \bmod (p - 1)$
  4. Send $(r, s)$ with message $m$
- Verification:
  1. Input: message $m, (r, s)$
  2. Check if $r^s (g^a)^r = g^m (\bmod p)$

### Avoiding (overly) long numbers

- Real-life keys are long. We can use any group where discrete log is hard.

A group is a set of elements and an associated operation such that it satisfies the following:

- Closure: $a * b$ is also a group element
- Associativity: $\forall a, b, c : (a * b) * c = a * (b * c)$
- Identity element: $a * e = a = e * a$
- Inverse: $a * b = e = b * a$

Additional Cryptographic Properties:

- Discrete logarithm should be hard
- Group operation should be efficient

Elliptic Curve Cryptography (ECC)

- Group elements are points on the curve $y^2 = x^3 + 7$
- Point "addition" using "geometry"

Elliptic Curve Digital Signature Algorithm

- Follows the same structure as ElGamal signature, but only on $x$-coordinate.

### Pitfalls

If we ever repeat signing nonce, we leak private key

Sony actually did this with Playstation 3 consoles.

Poor randomness makes private keys predictable. Use /dev/urandom (Linux) or java.security.SecureRandom. Common mistake was to use Math.random() or srand(time(0))

Logjam attack: downgrade security during handshake.

### Notes

- How are digital signatures and real-life signatures different in terms of why we trust them? What stops each from being forged by others?
- Assume somebody really clever has a way of solving the discrete logarithm problem easily. That is, for any given g,y,p, the adversary can compute $x$ such that $g^x \bmod p = y$. How can this algorithm be used to break security of Diffie-Hellman protocol?
- What is a nonce? What breaks if we reuse it between encrypted messages?
- In elliptic curve cryptography, why do we use mod $p$ integers? What would go wrong if we used real numbers?

  