MATH 314 - Class Notes

11/29/2016

Scribe: Andy Gonzalez Campos

Summary: Digital Signatures General Ideas

Hash Functions: They take a message and convert it into a much shorter digest (hash).

- h: M - S

functions "h" should be harder to reverse, difficult to find "X" such that:

- h(x)=y

Collision Resistant:Hard to find two messages m1 and m2 $h(m1) = h(m2)$

Birthday Paradox:
Probability that 2 people in a room with "k" people share a birthday is:

$$1 - \frac{364}{365}\frac{363}{365}\frac{362}{365}...\frac{366-k}{365}$$

If "k" = 23 then the probability is .503 In general if you have "n" things (1,2,3,...n) then pick 'k' at random (unlimited supply of each). Probability 2 are the same

$$1 - \frac{n-1}{n}\frac{n-2}{n}\frac{n-3}{n}...\frac{n-(k+1)}{n} = 1 - e^{\frac{k^2}{2n}}$$

- Digital Signatures

Used especially for:
Authentication: Bob can be sure Alice sent the message.
Non-Repudiation: Bob can prove to someone else that Alice sent the message.

- **1st. Idea:** Scanning a digital copy of your real signature and put it at the bottom of the message.

Need Signature and Message to be connected

One Method: RSA Backward

Like RSA Alice has (n,e) - Public Key
(p,q,d) - Private (Only Alice Knows)
Alice wants to send a message, 'm' to Bob
She computes $m^d \mod n = S$
She sends (m,s)
Bob uses Alice's public key, he computes $S^e \mod n$ checks to see if this is $= m \mod n$
If it is Bob accepts the signature, if not he rejects it

Why does this work?

$(S^e \mod n = (m^d)^e \mod n) = ((m^d e) \mod n = m \mod n)$

Why is it secure?

If Eve wants to pretend to be Alice she needs to be able to compute m^d, but this requires her to know Alice's private key.

General Idea of Signatures

Produce a signature using the message in a way that is only possible with knowledge of a private key. Want the signature to depend on the message in a way that can be verified using a public key.

- **Set of Messages M**
- **Set of Signatures S**
- **Set of Keys K**

Signature Function:
$hk : M -> S$
Verification Function:
$V(x, y) =$ ..... True if $hk(x) = y$
.... False if $hk(x) = /y$

-El Gamal-

P - a large primer number
α - Primitive root $\mod p$
a - Randomly Chosen in (2,3,4....p-2)
β - $\alpha^a \mod p$
Public key is (P, α, β)

If Alice wants to sign a message 'm' using El Gamal.

Pick K randomly from (2,3,4...P-2)

Compute $r = \alpha^k \mod p$

Compute $S = k^-1(m - ar) \mod p - 1$

Signature is pair (r,S)

She sends (m, (r,s)) to Bob

To verify Bob computes
$V1 = B^r * r^S \mod P$
$V2 = \alpha^m \mod P$
Accept the signature if $V1 = V2 \mod P$
Rejects Otherwise

Why does this work?

- $S = K^-1(m - ar) \mod P - 1$
- $SK = (m - ar) \mod P - 1$
- $m = (Sk + ar) \mod P - 1$
- $\alpha^m = \alpha^s k + ar \mod P$
- $(\alpha^a)^r * (\alpha^k)^S$
- $(B)^r * (r)^S V1$

Why is this secure?

If Eve wants to pretend to be Alice. She needs to obtain a value of S that is valid for Alice's secret key 'a' she needs an 'S' where:
$\beta^r * r^S = \alpha^m \mod P$
$r^S = \alpha^m * \beta^- r \mod P$
Eve has to solve this for S—DLP is Hard

—Digital Signatures—

DSA = Digital Signature Algorithm
Introduced by NIST in 1991

Pick q = a prime with around 160 bits
pick p = a (q+1) fr some a around 1024 bits
pick g = primitive rood $\mod P$
pick $\alpha = g^{(p-1)/q} \mod P$
pick $\alpha^a = g^{p-1} = 1 \mod P$

—Steps for DSA—

1. Pick K randomly in (1,2,3...q-1)
2. Compute $r = \alpha^k (\mod P)(\mod a)$
3. Compute $S = k^-1(m + ar) \mod q$

4. Digital Signature is (r,S)

—Verification—

1. Compute: $U1 = S^{-1}m \mod q$
2. Compute: $U2 = S^{-1}r \mod q$
3. Compute: $V = (\alpha^{U1}\beta^{U2} \mod p) \mod q$

If V = r Accept; if otherwise, Reject

—Check That this works—

- $m = (SK - ar) \mod q$
- $s^{-1}m = K - ars^{-1} \mod q$
- $K = S^{-1}m + ars^{-1} \mod q$
- $= u1 + au2 \mod q$

—Now—

- $r\alpha^k = \alpha^{U1+au2} \mod P$
- $\alpha^{u1} * (\alpha^a)^u 2 \mod P$
- $\alpha^{u1}\beta^{u2} \mod P$

Last step will only work if the Signature is Valid

—-Key Improvement is...—-

$(r, s) \leqslant q$ - Much smaller than P
Key arithmetic occurs $\mod P$ large and so can be made secure using a large value of 'P'.
Since we want digital signatures to work for arbitrarily large messages 'm,' we can just use large messages; but it is better to use a hash function to make them short first instead.
In general we use a hash function to generate a digest $h(m)$, which is used to produce a signature instead,