MATH 314 Fall 2019 - Class Notes

11/20/2019

Scribe: Cameron Crow

Summary: This series of notes covers the Digital Signature Algorhithm along with an explanation of why it works and a code example.

Notes: Part One Digital Signature Algorithm Explanation

1. Alice's public key is (p, q, α, β) where
   - p, q are two large prime numbers where $p = k(q) + 1$ for some integer k
   - In general p is typically about 300 digits and q is about 80 digits
   - $\alpha = g^k \equiv g^{(p-1)/q}(modp)$
   - $\beta = \alpha^a(modp)$
2. Alice wants to send a message M along with a signature S to prove she sent it
   - She picks an ephemeral key b, which is only used one time, such that $1 < b < q-1$
   - $r \equiv (\alpha^\beta(modp))(modq)$
   - $t \equiv \beta^{-1} * (m + \alpha * r)(modq)$
   - she sends M along with the signature (r,t)
   - so the message consists of (M,r,t)
3. Bob wants to verify that (r,t) is a valid signature so, he computes
   - $U \equiv t^{-1} * M(Modq)$
   - $V \equiv t^{-1} * r(Modq)$
   - $X \equiv \alpha^U * \beta^V(modp)(modq)$
   - if $X \equiv r(modp)(modq)$ bob accepts the message
4. We know that this works because
   - $bt \equiv (M + a * r)(modq)$
   - $M \equiv bt + ar(modq)$ and if we multiply the whole equasion by t^{-1}
   - we get $t^{-1} * M \equiv b + (t^{-1} * ar)(modq)$
   - $b \equiv (t^{-1} * m) + (t^{-1} * ar)(modq)$
   - note we stated earlier that $U \equiv t^{-1} * M(Modq)$ and $V \equiv t^{-1} * r(Modq)$
   - so $r \equiv \alpha^\beta \equiv \alpha^{U+\alpha*V}$

- $r \equiv \alpha^U * \alpha^{\alpha(V)}$
- finally $r \equiv \alpha^U * \beta^V \equiv X(modp)(modq)$

5. These signatures verify the sender but don't adiquitly verify the integrity of the message. To tie these signatures and messages together we use a cryptographic Hash function.

Part Two Digital Signature Algorithms: Hashes

Definition: Hash function is a function that takes in a large (potentially arbitrarily large) input and produces a much smaller output called a digest.
Definition: Collisions is two input (x,y) to a hash where $H(x) \equiv H(y)$, these are unavoidable.

for a cryptographic hash we desire 3 properties in increasing strength.

1. Preimage resistant: given a digest D it should be hard to find an input x where $H(x) \equiv (d)$
2. Weakly Collision resistant: given an input x it should be hard to find a y where $H(x) \equiv H(y)$ for any non changing x
3. Weakly Collision resistant: given any two inputs x,y it should be hard to find any occurrences of $H(x) \equiv H(y)$

Suppose we have a cryptographic hash function and a signature algorithm $S(x)$ to sign a message M we send $S(H(M))$ this is Signature(Hash(Message)).

Our end goal is to make it hard for eve to find another message which shares the same signature that is valid. If the signature is inside of the hash, we can find lots of messages with the same signature which can lead to the same hash. With the hash on the inside, it is hard to predict the hash of a message. So the output of a hash function can safely be signed.