MATH 314 - Class Notes

5/9/2017

Scribe: William Petty

Summary: The Basic Goals of Cryptopgraphy, Hash Functions, and RSA-Backward, including digital signatures

Notes:

Basic Goals of Cryptography

1. Confidentiality: Eve can't learn what Alice and Bob are sending each other.
2. Data Integrity: Eve can't modify the messages that Alice and Bob are sending each other.
3. Authentication: Bob should be able to verify that the messages are coming from Alice.
4. Non-repudiation: Bob should be able to prove that Alice sent a message.

Hash Function

$h(x) \Rightarrow d$
Takes a long string x and converts it into a much shorter d(digest)
Ex: h(x) = x (mod 2) ¡- this is a sort of parity function

Error Correction

Send a message m
Send pair (m, h(m))
Bob checks if h of the message he receives is what Alice claims it was. If not, then he knows that the message was corrupted.
This catches accidental errors in transmission, but not malicious errors. It does not prevent Eve from altering the message.

Ideal Properties of Hash Functions

- Fast to compute.
- Able to take long inputs and convert them to a much shorter digest.
- If $X_1 \neq X_2$, but $h(X_1) = h(X_2)$, then we call this a collision of h.

Ideal Collision Properties

- Pre-image resistant: given y, it is difficult to find x, such that $y = h(x)$. In other words, it is hard to invert $h(x)$.

- Weak-collision-resistant: given x_1, it is hard to find another x_2 with $h(x_1) = h(x_2)$
- Strong-collision-resistant: it should be hard to find any two x_1 and x_2, where $h(x_1) = h(x_2)$

Simple hash:
fix N (large)
H(x) = $x(modN)$

- Fast to compute
- Arbitrarily, x is reduced to something smaller(almost N)
- Not pre-image resistant. Given y, then $x = y, y + N, y + 2N...$ all hash to Y

Discrete-Log-Hash

Pick a large prime q such that p = 2q + 1, p also being a prime number.
Pick α, β to be two primitive roots mod p
h(x) takes in x E [1, (a^2)–1]
write x = a + bq, where $0 \leq a \leq q$ and $0 \leq b \leq q$
$h(x) = h(a + bq) = \alpha^a * \beta^b(modp)$
Taking x and converting it to something much smaller in size, about $\sqrt{x}$.
This has strong collision resistance.

Show that if we can find a collision, then we can solve the discrete log problem $\alpha = \beta^x$
The discrete log problem says this should be difficult.
Proof: Suppose we find a collision. $h(x_1) = h(x_2)$ but $x_1 \neq x_2$
$x_1 = a_1 + b_1 * q$
$x_2 = a_2 + b_2 * q$
$h(a_1 = b_1 * q) \equiv h(a_2 + b_2 * q) \equiv \alpha^{(a_1)} * \beta^{(b_1)} \equiv \alpha^{(a_2)} * \beta^{(b_2)}$ (mod p)
($\star$ function) $\Rightarrow \alpha^{(a_1 - a_2)} * \beta^{(b_1 - b_2)} \equiv 1$(mod p)

Since α and β are primitive roots, there exists some number C, such that $\alpha = \beta^c$(mod p)

Plug this into $\star$, this becomes $(\beta^c)^{(a_1 - a_2)} * \beta^{(b_1 - b_2)} \equiv 1$(mod p) $\equiv B^{c((a_1 - a_2) + (b_1 - b_2))} \equiv 1$(mod p)
By Fermat's Little Theorem: $C(a_1–a_2) + (b_1–b_2) \equiv O$(mod p-1)
We can solve this for C using modular arithmetic. We found C so that α congruent to β^C (mod p).
This solves the discrete log problem (hard) Thus finding any collision must be hard. So we have strong collision resistance.
However, Discrete log hash is too slow and doesn't work for long inputs, so it doesn't get used in practice. In practice MD5, SHA1, SHA-a, RUPEMD-60 were used. In 2005, a way to produce collisions in MD5 was found. In January 2017, a collision attack was found against SHA1.

These hashes will be used to produce **Digital Signatures**. "regular signatures" prove that you signed a physical piece of paper How do we do this digitally? Not by scanning your signature, because this is easy to forge. Need a way to digitally sign things, and for your signature to depend on the message being sent in a way that can be verified. This can be done by:

RSA – Signature (RSA-backward)

Alice produces a public key
(n, e) $\Leftarrow$ publishes this
$n = pq$ d $\Leftarrow$ secret
She wants to send m to Bob and sign m. She computes $S = m^d(modn)$. She sends (m, s) to Bob. This is her message (m) along with her signature (s). Now, Bob needs to verify the signature.
Bob computes $V = S^e(modn)$. The signature checks out if V $\equiv m(modn)$. This works because V $\equiv S^e \equiv (m^d)^e \equiv m(modn)$.
Why is this secure? Suppose Eve wanted to pretend to be Alice. Eve wants to send m'. To forge Alice's signature, Eve needs to find some S' so that S'$^e(modn)\equiv$ m'(mod n)
This is equivalent to computing $m'^d(modn)$. To do this, you need to know d. Finding d is supposed to be hard. Eve can't forge signatures without breaking Alice's RSA key.