CS 591, Lecture 18

Jared Saia
University of New Mexico

Class Outline

- Digital Signatures
- Peer Join
- Message send under fail-stop fault model
- Message send under Byzanting fault model

Digital signatures and ID's

What we want:

- unique numerical id for each entity in the network
- signature that can be easily checked for validity by anyone
- signature that can be forged by no one
- We'll consider three ways to get this

Scenario 1: IP-addresses

- Assume technical advances make it more difficult to spoof IP-addresses
- Internet II is moving towards this goal
- Can simply use IP-addresses as the ID and signature
- The "certificate" of this ID is just an acknowledgement by direct ping.

Scenario 2: A Neutral, Trusted Party

- A trusted party distributes "black boxes" (these can be software byte code or actual hardware) to each entity
- Each black box contains a unique id x and a secret encryption function e (which can not be recovered)
- The public key for e, e^{-1}, is released to everyone, and the trusted party "forgets" e
- The "black box" creates the signature $(x, e(x))$ whenever called (the secure id is x and the certificate of uniqueness is $e(x)$)

Digital Signatures

- Note that it is easy to add "time stamps" to the digital signatures
- This ensures that the "freshness" of a message can be verified (i.e. another peer won't resend an old message from v and try to pass it off as new)
- Alternatively, for each message, can create a digital signature for the message by encrypting the entire message.

Scenario 3: Mass Paranoia

In this scenario, no entity is trusted

- When a new peer, v, joins the network, it chooses some secret encryption function e_v and publishes e_v^{-1} (we need to ensure that if one good peer receives this message, then all peers receive it)
- All peers in the network decide on some random ID x for v - this ID is known to all peers (the alg for doing this is to be shown).
- v uses the signature $(x, e_v(x))$, where x is the ID and $e_v(x)$ is the certificate.

Choosing a Random Number

- Want to choose a random number which is known by all good peers
- Assume that each peer has a unique digital signature
- Assume every peer can communicate with every other peer
- Assume synchronous communication
- Assume a majority of good peers

Choosing a Random Number

The Algorithm:

1. Each good peer u chooses a random number r and sends out a message which consists of r *and* u's digital signature to everyone.
2. For each peer u, each good peer v sends out a message to everyone which consists of the message that v received from u along with v's digital signature.
3. Each good peer now receives for each fixed peer u, a collection of messages signed by every other peer concerning what they claim they received from u. If for some fixed peer u, two messages signed by u disagree on u's chosen value, then u is assumed to be "bad". Otherwise r_u is set to be the single unique number that was provably received by the other peers.
4. Each good peer now computes r to be the bitwise exclusive-or of every verified r_u for each peer u.

Cryptography

- We still haven't shown how to get e_v and e_v^{-1}
- This requires cryptographic machinary
- Depends on the difficulty of factoring large numbers

Analysis

- Note: all good peers have the same value set for r_u for all peers u.
- If u sent the same values to all good peers, this is trivially true
- If u sent different signed values to two good peers, then in the third step of the algorithm, all good peers invalidate u
- This implies then that all good peers will choose the same number, r, in the last step of the algorithm
- Since at least one of the peers is good, at least one of the number chosen will be random, hence r will be a random number

How to get e_v and e_v^{-1}

1. Select two large prime numbers p and q at random. Let $n = p * q$
2. Select a small odd integer u that is relatively prime to $\phi(n) = (p-1)(q-1)$
3. Compute v which is the multiplicative inverse of u mod $\phi(n)$
4. $e^{-1}(x) = x^u \pmod{n}$ - public
5. $e(x) = x^v \pmod{n}$ - private

"Analysis"

- Number theoretic arguments can show that for any number m:

$$e^{-1}(e(m)) = m^{uv} = m \pmod{n}$$

- Number theoretic arguments further show that computing v given u and n is as hard as factoring n
- Hence, if factoring is "hard" then the function e remains private even though e^{-1} is known

Peer Join

- An idea: Use consistent hashing (from the Akamai paper)
- We've got a family of $k \log n$ base hash functions that map an ID to a point on the unit circle
- For each circle, let the assigned informer be the peer that is closest in the clockwise direction to the point the ID hashed to
- This gives us $k \log n$ "random" peers in the network to be used as informers
- Let f be the function that maps ID's to sets of ID's on these $k \log n$ unit circles

Now what?

- So assume that we've got unique id's and digital signatures for each of the peers
- Now how do we use these id's to set up an informer network?
- What we "want" is for there to be a mapping from each ID to a "random" set of $k \log n$ peers in the network

Join Protocol for joining peer p

- p broadcasts message that i_p is joining to all peers in network
- Let S be the set of peers p receives replies from that claim to be p's informers. Let S' be the set of verified informer in S (i.e. $S' = \{s \in S | i_s \in f(i_p)\}$
- p sets it's informer set $I(p)$ to be S'
- p establishes connections with all peers in $I(p)$

Join Protocol for non-joining peer o

- o receives join message from p and obtains p's id, i_p
- If $i_o \in f(i_p)$, o broadcasts a claim that it is an informer for p to all other peers. It also adds p to its informant set, $I^{-1}(o)$
- o receives messages from all peers which claim to be informers for p. Let $I(p)$ be the verified set of all such peers
- Let $I^{-1}(o)$ be the set of peers that o informs on. For each $x \in I^{-1}(o)$ such that (x, p) is an edge in the underlying network, o establishes an informer-to-informer link to each peer in $I(p)$

Analysis

- Each peer x has a set $I(x)$ of informers chosen randomly from all the peers. $|I(x)| = k * \log(n)$
- The join algorithm creates a network which mirrors G_p
- For each edge (x, y) in G_p, all peers in $I(x)$ and in $I(y)$ are connected in a full bipartite graph

Analysis

- Assume we are given some protocol for creating a p2p network without informers (e.g. the gnutella protocol or the CRN or DRN protocol)
- This protocol will define some graph G_p over the peers which would be the topology of the overlay network without informers
- The join algorithm creates a network which mirrors G_p

Fail-stop faults

- First consider the situation where bad peers suffer "fail-stop" faults
- Recall "fail-stop" means that bad informers simply stop communicating
- We give a light-weight scheme for informing and communicating through the network in this case

Communication: Peer to Informer

- Peer protocol: When a peer p requests a data item or serves a data item, it sends out a message to a random informer in $I(p)$
- Informer protocol: When an informer receives a message from its peer, it does the following
  - If the message is a data item requested by a previously logged request, it logs this response and broadcasts information on the response to all other informers in $I(p)$
  - Otherwise, with probability p_i, it logs information on the message sent out and broadcasts this information to all other peers in $I(p)$
  - Finally, it sends the message to the next appropriate informer cloud using the protocol described under *Informer to Neighboring Informer Cloud*

Informer to Neighboring Informer Cloud

- When an informer has a message which is to be sent to a neighboring informer cloud, it does the following:
- Until it gets a response, it chooses a random informer in the neighboring informer cloud and pings that informer
- Once a live informer is found, the message is sent to that live informer

Informer to Peer

- When an informer receives a message to its peer, it does the following:
- If the message is a request for data, with probability p_i, it logs information on the request and broadcasts this information to all other informers in the informer cloud.
- Next, it sends the message to the peer

Analysis

- Let p_f be the ratio of the number of failed peers to the total number of peers
- Assume we want to communicate a message from some peer a to some peer b in the original network
- Let p_i be any number in the range 0 to 1
- Let m be the number of informers in each informer cloud (m is polylog in n)

Analysis

Theorem: The above protocol allows for communication along the path from a to b such that

- The message sent from a is known by all peers in $I(a)$ with probability p_i, and is known to all peers in $I(b)$ with probability p_i
- The expected time to send the message is $(1/p_f)l + 2 * p_i$
- The expected number of messages sent is $(1/p_f)l + 2 * p_i * m$
- The time to send and number of messages to send are sharply concentrated around their expected values

Analysis

- Let p_f be worst case ratio of bad peers in the network to good peers
- Q1: What is the probability that there is some peer x, such that $I(x)$ does not have a majority of good peers?
- Q2 (Easier): What is the probability that for some *fixed* peer x, $I(x)$ does not have a majority of good peers?

Byzantine Faults

- Now we'll consider the case of Byzantine faults
- Recall that these Byzantine are strictly worse than fail-stop faults
- We'll assume that at any time no more than a p_f fraction of the peers are bad for some $p_f < 1/2$
- To handle Byzantine faults, we'll first consider the probability that any informer cloud contains a majority of bad peers.

Analysis

- A2: Let x be some fixed peer. Note that $I(x)$ is formed by making $log^k(n)$ independent random choices from the n peers in the network
- Let X_i be 1 if this choice is a bad peer and 0 otherwise
- Note that $E(X_i) = p_f$.
- Let $X = \sum_{i=1}^{k*\log n} X_i$. Note by linearity of expectation that $E(X) = k * \log n * p_f$

Chernoff Bounds

- Assume p_f is a fixed constant less than 1/2
- Then we want to find:

$$P(X \geq (1/2) * k * \log n) = P(|X - E(X)| \geq \delta E(X))$$

where $\delta = (1/2 - p_f)/p_f$

- By Chernoff Bounds,

$$P(|X - E(X))| \geq \delta E(X) leq 2e^{\frac{-E(X)\delta^2}{4}}$$

Chernoff Bounds

- In this case, that means that for $\delta = (1/2 - p_f)/p_f$,

$$P(|X - E(X))| \geq \delta E(X) leq 2e^{\frac{-E(X)\delta^2}{4}}$$

- This probability is $2e^{-Ck * \log n}$ where $C = (p_f * \delta^2)/4$ (note that C depends only on p_f)
- Thus for any $k' > 0$, k can be chosen as a function of k' and p_f so that this probability is no more than $n^{-k'}$

Union Bounds

- Now we can answer the more difficult Q1:
- Q1: What is the probability that there is some peer x, such that $I(x)$ does not have a majority of good peers?
- We know the probability that for a *fixed* peer x, the probability of failure is no more than $n^{-k'}$
- By Union Bound, we can say that the probability of failure for any of the peers is no more than n^{-k+1}