CS395t: Secret Sharing

Razvan Surdulescu
surdules@cs.utexas.edu
March 10, 2004

How to Share a Secret

Adi Shamir, MIT
Communications of the ACM
November 1979, Vol. 22, Nr. 11

Motivation

- We have a safe that contains a secret
- We wish to give n people access to this safe
- Access is granted only if k (or more) of the n people are present (k <= n)

Motivation cont'd

- In general, the secret is some data D
- We wish to divide D into n pieces (D_1, ..., D_n) such that:
  - Knowledge of k (or more) D_i pieces makes D easily computable
  - Knowledge of k-1 (or fewer) D_i pieces leaves D completely undetermined
- This is a (k, n) threshold scheme

Applications

- Reliability
  - Protecting a secret key that resides in a single location is difficult
  - By splitting the key in n=2k-1 pieces, we can re-construct it even if half the pieces are lost
- Convenience
  - Use a (3, n) scheme to share the company's digital signature among n executives
  - At least 3 executives must be present to sign

Applications cont'd

- Threshold schemes are ideal when mutually suspicious individuals, with conflicting interests, must cooperate
- A sufficiently large majority can take action
- A sufficiently large minority can veto

Implementation

- Polynomial interpolation
  - Given k 2D points $(x_1, y_1), \ldots, (x_k, y_k)$, with distinct x_i's, there is only one polynomial q(x) of degree k-1 such that $q(x_i) = y_i$ for all i.
- Assume the secret data D can be made into a number
  - Let $q(x) = D + a_1x + \ldots + a_{k-1}x^{k-1}$, where a_i are randomly chosen
  - Let $D_i = q(i)$
  - Note that $q(0) = D$

Implementation cont'd

- Given k (or more) D_i values, we can uniquely determine the coefficients of the polynomial q(x), and therefore, D
- The computations are performed over a field [0, p)
  - p is a prime number greater than both D and n
- If k-1 (or fewer) D_i values are known
  - For each D' in [0, p), we can construct one polynomial q'(x) of degree k-1 with the required properties, therefore nothing is revealed about the real value of D

Implementation cont'd

- Efficient interpolation schemes run in $O(n \log^2 n)$ time
  - Even naïve $O(n^2)$ schemes are generally sufficiently fast
- Large values of D can be broken down into shorter pieces that are handled separately
- Individual D_i pieces can be deleted without affecting the other D_i pieces

Implementation cont'd

- All the D_i pieces can be changed at once without affecting the original D
- The D_i pieces can be shared differently based on their importance
  - The CEO gets 3 pieces
  - The VPs get 2 pieces
  - The middle managers get 1 piece

Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance

Michael O. Rabin, Harvard

Journal of the ACM

April 1989, Vol. 36, Nr. 2

Information Dispersal

- Consider a distributed network
  - Nodes are sparsely connected (not every two nodes are connected by a single edge)
  - A user sends a file F from node A to B via some path π consisting of 1 or more edges
    - Although the probability of any edge failing is low, the probability of the path failing can be high

Information Dispersal cont'd

- In case of failure
  - Re-transmit the file
    - Loss of time
  - Choose k paths π_i and send the file along each one simultaneously
    - Loss of bandwidth
- IDA disperses the file F into n pieces
  - The file can be reconstructed from any m pieces
  - Each piece is of size |F|/m, and the total amount of information sent is (n/m) * |F|

4/12/04 13

Information Dispersal cont'd

- Space efficiency
  - We can choose n and m such that (n/m)~1, therefore the overhead is low
- Time efficiency
  - The splitting and reconstruction algorithms are efficient (more later)
- File pieces can be transmitted in parallel, which better utilizes network resources

4/12/04 14

IDA Theory

- Let $F=b_1b_2...b_N$ be a file, where b_i are in the range [0, B]
- We want to disperse pieces of F with the assumption that no more than k pieces will be lost in transmission
- Choose p such that p > B
  - If the file consists of bytes, p = 257
  - All the following computations are in Z_p

4/12/04 15

IDA Theory cont'd

- Choose n and m
- Choose n vectors $a_i = (a_{i1}, ..., a_{im})$ in Z_p^m such that every m different vectors are linearly independent (with high probability)
- F is segmented into sequences of length m
  - $F = (b_1, ..., b_m),(b_{m+1}, ..., b_{2m}),...=S_1,S_2,...$

4/12/04 16

IDA Theory cont'd

- Let $F_i = a_i S_1, a_i S_2, \ldots, a_i S_{N/m} = c_{i1}, c_{i2}, \ldots, c_{iN/m}$
  - $|F_i| = |F|/m$
- Say we have m pieces of F $(F_1, \ldots, F_m)$
- Let A be the m * m matrix whose i^th row is a_i

$$A \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix}, \text{therefore} \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = A^{-1} \begin{bmatrix} c_{11} \\ \vdots \\ c_{m1} \end{bmatrix}$$

IDA Theory cont'd

- The matrix A^{-1} can be computed once
  - For sufficiently large F, the cost of this computation is majorized by the cost of reconstructing F, even if we use an $O(m^3)$ inversion algorithm
- Each character of F requires 2m mod p-operations
  - The split and reconstruction involve just inner products that are readily optimized in hardware

IDA Theory cont'd

- In order for the matrix A to be invertible, it is necessary that the vectors a_i be linearly independent
  - Select a_i randomly from Z_p^m!
  - It can be shown that A is nonsingular (invertible) with probability nearly 1 – (1/p)
  - The randomness of the a_i vectors further prevents eavesdroppers from reconstructing *some* of F by intercepting *some* of the F_i pieces

IDA Theory cont'd

- The vector a_i can be included as the header of the piece F_i
  - The matrix A can be constructed when all m pieces F_i are received
  - It is obviously essential that all F_i's be encrypted in this case, to prevent against eavesdropping

Securing Replicated Data

- Two major issues in distributed systems data security:
  - Secrecy: cannot observe confidential data
  - Integrity: cannot corrupt or modify data
- Distributed data across multiple computers compounds the risk of data theft/corruption and availability

Routing for Parallel Computers

- PC_n = parallel computer with $N=2^n$ nodes
  - Each node x contains a processor Cx and memory Mx
  - A node is of the form $\{0, 1\}^n$
    - E.g. {0, 1, 0, 1}
  - The notation x//i means that we flip bit i
    - {0, 1, 0, 1}//2 = {0, 0, 0, 1}
  - Each node x is connected by two-way links to each of the nodes x//i, where 1<=i<=n

RPC cont'd

- Seminal paper by L. Valiant
  - Each node x has a packet of information Px that has to be sent to a destination node $\pi(x)$ (π:C_n->C_n is a permutation over the nodes of C_n)
  - Two phase approach:
    - 1. Route packets from x to a random node R(x)
    - 2. Route packets from R(x) to $\pi(x)$
  - With probability $1-N^{-k}$ each packet reaches its destination in time c $\log_2(N)$ and the queues at each node are shorter than d $\log_2(N)$
    - k is a function of c and d

RPC using IDA

- The packets Px are large
  - Break them into pieces Pxi, such that $m=\lfloor 5n/6 \rfloor$ pieces suffice to reconstruct Px
  - Each piece Pxi has a ticket Txi and is routed independently
    - The ticket is a vector of integers from 0 to n of length 2(n+1)
    - Txi specifies the route from the source node (x) to the destination node (y)
      - Pxi → x // Txi[0] → x // Txi[0] // Txi[1] → ... → y

RPC using IDA cont'd

- At any time $1 <= t <= 2(n+1)$, there are pairs (P, T) at any node Cy
  - For $1 <= j <= n$, if $T[t] = j$, send P to to y//j
  - By time t+1, this completes for all nodes and all links from y to its neighbors
- Assume each node has a buffer large enough to hold 6 packets

RPC using IDA cont'd

- Simultaneously for all x in Cn
  - Split Px into Px1, ..., Pxn
  - Randomly choose n pairwise different nodes $R_1(x)$, ..., $R_n(x)$
  - Select pairwise vertex-disjoint paths $D_1(x)$, ..., $D_n(x)$ from x to $R_1(x)$, ..., $R_n(x)$, each of length at most n+1
  - Select vertex-disjoint paths $E_1(x)$, ..., $E_n(x)$ $R_1(x)$, ..., $R_n(x)$, to $\pi(x)$, each of length at most n+1
  - Attach appropriate ticket Txi to Pxi for routing from x to $\pi(x)$ along $D_i(x)$, then $E_i(x)$

RPC using IDA cont'd

- Observations
  - m pieces of Px1, ..., Pxn suffice to reconstruct Px
  - A separate proof will be given to show that such paths as $D_i(x)$ can be constructed
  - If $\text{length}(D_i(x)) = k < n+1$, pad with zeros; same for $\text{length}(E_i(x))$
  - If buffers overflow, packets are rejected and lost

RPC using IDA cont'd

- Theorem 1: for any given permutation π, the probability that all packets reach their destination is $1-(1/N^4)$
  - Let Y(y,x,t) = random variable showing number of pieces Pxi arriving at node y at time t
    - Trivial that Y(y,x,t) can only be 0 or 1
  - Let p(y,x,t) be probability that Y(y,x,t) = 1

RPC using IDA cont'd

- If for some y, Σ Y(y,x,t) >= 5n for all x
  - There are 5n |Pxi| = 5n |Px| / m = 5n |Px| / (5n / 6) = 6 |Px| packets at node y, which means overflow at node y
- We want to know the probability of the condition above being true
  - At t = 1, we have n pieces Pxi at each node, so Σ p(y,x,t) = n for all x

4/12/04 29

RPC using IDA cont'd

- The random variables Y(y,x,t) are pairwise independent
- Use Raghavan-Spencer theorem
  - If Y1, ..., Yn are independent Bernoulli trials with expected sum n, then for δ>0:

$$\Pr\left(\sum_i Y_i \geq (1+\delta)n\right) \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^n$$

4/12/04 30

RPC using IDA cont'd

- The probability of the buffer overflow event (Σ Y(y,x,t) >= 5n) is bounded by δ = 4
  - Using Spencer-Raghavan, the theorem claim is immediate (for n >= 4)
  - The probability that all packets reach their destination (enough IDA pieces reach the destination to allow for reconstruction of the original packet) is $1-(1/N^4)$

4/12/04 31

RPC using IDA cont'd

- Theorem 2:
  - Assume that within a transmission round, fewer than N/n links fail
  - We break Px into n pieces such that m=⌊n/2⌋ pieces suffice for reconstruction
  - Allow for large enough buffers to make buffer overflow very unlikely
  - Then the probability of all packets reaching their destination is $1 - 2N(4e/n)^{n/4}$

4/12/04 32

RPC using IDA cont'd

- Lemma:
  - Let $C_n = \{0, 1\}^n$, $S = \{y_1, ..., y_n\}$ subset of C_n, x in C_n – S.
  - There exist paths $D_1, ..., D_n$ from x to $y_1, ..., y_n$ so that for $i \neq j$, D_i and D_j only have node x in common and length(D_i) <= n+1 for 1 <= i <= n

4/12/04 33

How to Make Replicated Data Secure

Maurice P. Herlihy, J. D. Tygar

August 1987

CMU-CS-87-143

4/12/04 34

Replication

- Store long-lived data in multiple places (repositories)
  - This provides fault-tolerance
- Start with a threshold value t
  - An adversary cannot determine or corrupt the original data by inspecting fewer than t repositories
- Analyze costs of
  - Replication for availability (tolerate t failures)
  - Replication for security (tolerate t compromised sites)

4/12/04 35

Costs of replication

- Secrecy is cheap
  - Private and public key encryption schemes
  - Key distribution is of particular interest, since storing the key in a volatile medium exposes it to compromise
    - Distribute the key directly in the replication protocol
- Integrity is expensive
  - Communicate with additional sites

4/12/04 36

Terminology

- Bit security
  - No processor with randomized polynomial resources can derive information about any bit in the ciphertext with certainty greater than _ + ε, for any $\varepsilon > 0$
  - This assumes some generally accepted limits of complexity theory (e.g. taking k^{th} roots modulo pq cannot be done in randomized polynomial time)
- Perfect security
  - No processor with unlimited resources can derive a probability distribution of the corresponding cleartext other than a uniform distribution

4/12/04 37

Quorum Consensus Repl.

- Repository
  - Long term storage for object state
- Quorum
  - A set of repositories whose cooperation suffices for an operation
- Assignment
  - Associate an operation with a set of quorums

4/12/04 38

Quorum Consensus Repl.

- Replicated file
  - A collection of timestamped versions
  - To read: take latest version from read quorum
  - To write: generate new time-stamp, record new version at write quorum
- An assignment is correct iff each read quorum has a non-empty intersection with each write quorum

4/12/04 39

Private Key Secure Quorum Consensus (SQC)

- Protect secrecy against an adversary who can observe < t repositories
  - Depends on a bit-secure, probabilistic private key encryption scheme
- Implementation
  - Front-ends: clients, volatile store
  - Repositories: connected, long-term store
  - Dealer: communicate with repositories, has a source of random bits, volatile store

4/12/04 40

Private Key SQC cont'd

- Phases
  - Object initialization: dealer chooses random key K, uses (t, n) secret sharing to send it to each repository; all data stored in each repository is encrypted by K first
  - Front-end initialization: create K by reading t out of n secret shares, store K in volatile cache
  - Operation execution: read data, decrypt, perform the operation, encrypt, store data

4/12/04 41

Private Key SQC cont'd

- An adversary can still glean data
  - For example, if the log timestamp entries are not encrypted, they can provide hints
  - The frequency of read/write operations can also provide hints
- If the threshold is set to the smallest quorum, there is no availability penalty

4/12/04 42

Examples

- Example #1
  - Read and write operations are equally important
  - Read and write quorums, as well as the share threshold require a majority of $\lceil (n+1)/2 \rceil$ repositories
    - Registration does not incur an additional penalty, since it can be done at the first quorum
  - Up to $\lceil (n+1)/2 \rceil$ may fail or be compromised by an adversary

4/12/04 43

Examples cont'd

- Example #2
  - Read is more important than Write
  - Read quorums to have size 1, write quorums have size n
  - Clearly, a threshold of size 1 is not prudent since it can be spoofed, so the read threshold is really somewhere between 2 and n
  - Registration incurs additional penalty

4/12/04 44

Public Key SQC

- Instead of a single key K, use an encryption key K_E, and a decryption key K_D
- Similarly, use an encryption threshold t_E, and a decryption threshold t_D to divide each key into pieces
- This provides more flexibility in terms of performance, availability, and security trade-offs
  - E.g. If integrity is not a concern, set $t_E = 1$

4/12/04 45

On-the-Fly Reencryption

- Used when there is reasonable doubt that the encryption key has been compromised
  - A file is replicated among n repositories, with r read quorums, w write quorums, and threshold t
  - A front-end that knows K, can reencrypt with K' if it has access to max(r, w, n-t+1) repositories

4/12/04 46

Preserving Integrity

- So far, we've been concerned with preserving secrecy from snoopers
- We now want to preserve integrity against an active adversary
  - Detect modifications
  - Treat the repository as if it had crashed

4/12/04 47

Preserving Integrity cont'd

- Encrypt cleartext along with internal redundancy check
  - Rabin and Karp checksum
- Define an integrity threshold t_i
  - $t_i <= t$ (for private SQC) or t_E (for public SQC)
  - Require quorum intersections to have cardinality at least t_i
  - Ensures that each read quorum includes at least one uncompromised repository with the file's current data

4/12/04 48

Preserving Integrity cont'd

- The adversary may take a snapshot of the data and replace it at a later time
  - This means that old timestamps will also be replaced, so the latest timestamp is correct
- The protocol is optimal within the constraints of the problem
  - May seem expensive due to larger minimum intersection of read and write quorums
  - Anything weaker is subject to spoofing

Preserving Integrity cont'd

- Compromise scenario
  - File replicated at n repositories
  - Read quorums of size r, write quorums of size w, read intersect write at x repositories: r+w-x=n
  - R, W, X are disjoint sets of repositories of sizes r-x, w-x, and x
    - The repositories in X are controlled by an adversary

Preserving Integrity cont'd

- Compromise scenario cont'd
  - Client A writes the value a at some write quorum
  - Adversary snapshots X
  - Client B writes the value b at W union X
  - Adversary overwrites X with previous
  - Client C reads the (obsolete) value a from R union X
  - Since B intersect C = X, C can be spoofed

RPC using IDA cont'd

- Lemma follows from following claim
  - Let U_n be the set of unit vectors in C_n (vectors e_i where $e_i[j] = \delta_{ij}$)
  - Let U subset of U_n, H subset C_n, |H| = |U| = k, H intersect U = empty set
  - There exist k vertex disjoint paths $F_1, \ldots, F_k$ connecting the nodes in U to the nodes in H

4/12/04 53