CS 5432:

Secret Sharing

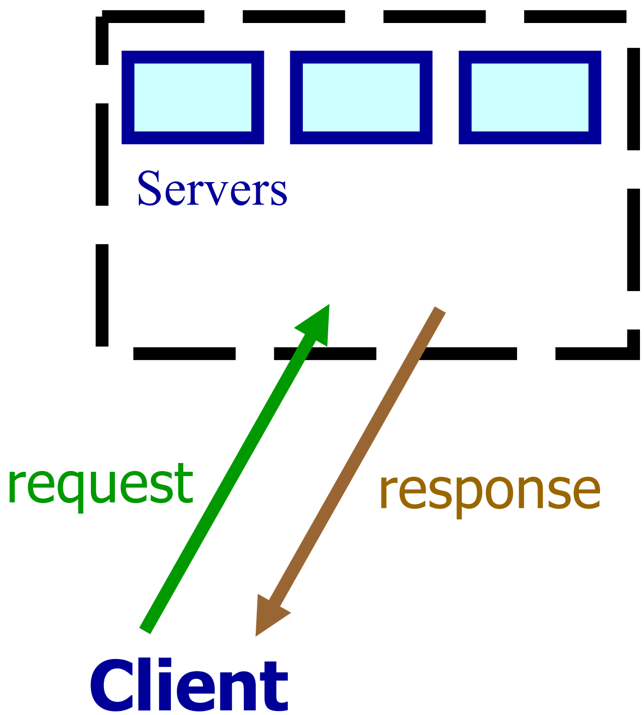
Fred B. Schneider

Samuel B Eckert Professor of Computer Science

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.

Cornell CIS
Computer Science

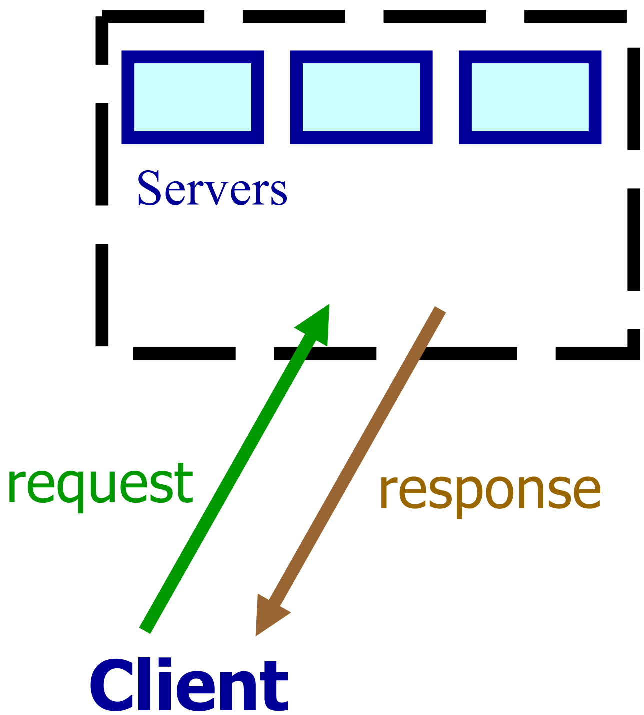
State Machine Replication



The basic recipe …

- Servers:
  - deterministic state machines
  - assumed to fail independently
- Clients:
  - make requests
  - synthesize service response from individual server responses

State Machine Replication

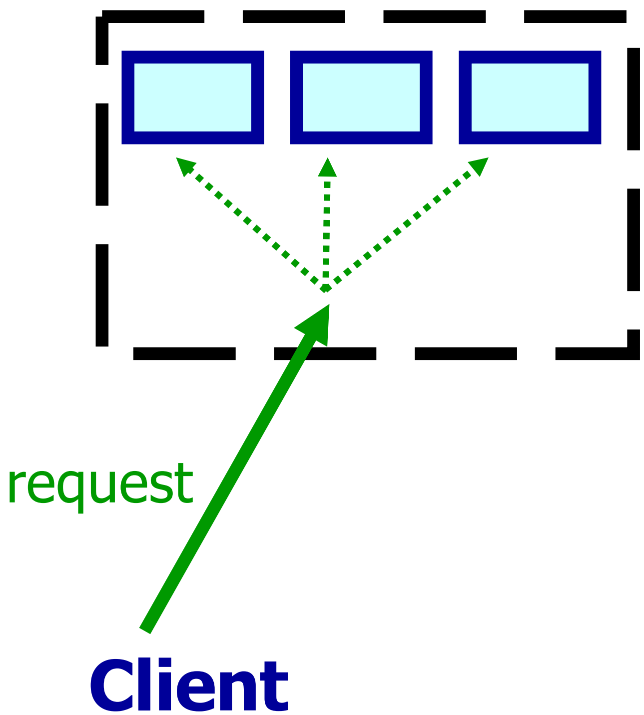


Supports:

- ~~Confidentiality~~
- Integrity
- Availability

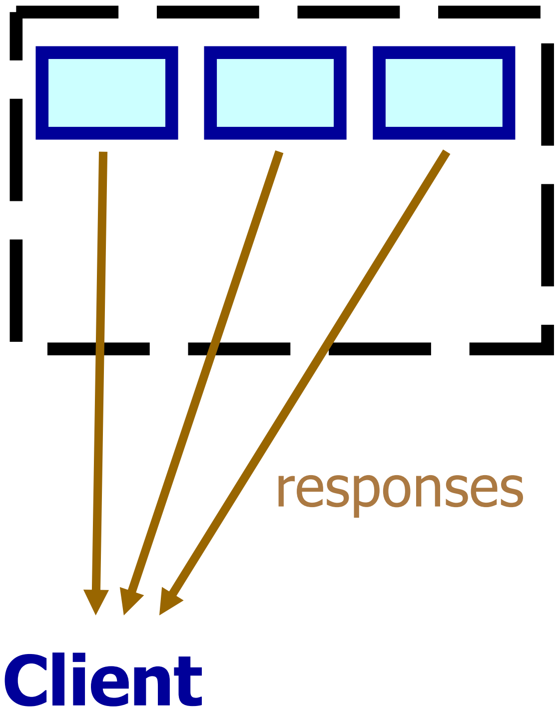
of whatever service is provided by a single replica.

State Machine Replication: Internals



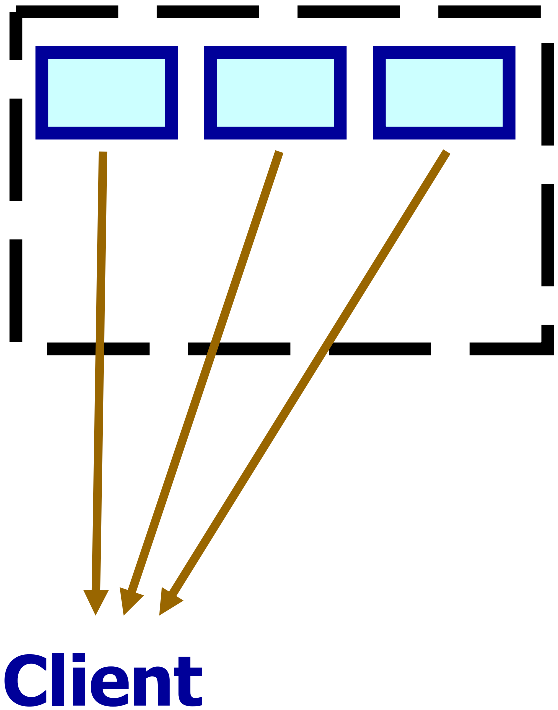
- **Agreement protocol** so all correct servers process requests in same order.

State Machine Replication: Internals



- **Agreement protocol** so all correct servers process requests in same order.
- **Authentication protocol** so client can distinguish and synthesize responses from different servers.
  - Servers need (different) secrets.

Big Picture: State Involving Secrets



Alternative Implementations:

- Secret stored at every replica; client counts votes.
- Pieces of secret stored at every replica; client combines pieces.
- Every replica performs computation using secret pieces; client combines results of those computations.

(n, t) Secret Sharing

(n, t): n shares, where t suffice to reconstruct.

Variations:

- (n, n) secret splitting
- (n, t) using (n, n) secret splitting
- (n, t) using polynomials
- verifiable secret sharing
- function sharing
  - … authentication of replica responses
- proactive secret sharing

(n, n) Secret splitting

Goal: Given a secret s:

- compute shares $s_1, s_2, \ldots, s_n$
- Knowledge of <u>all</u> shares allows s to be recomputed
- Knowledge of <u>fewer</u> shares reveals nothing about s.

Assume: $s, s_1, s_2, \ldots, s_n$ come from a finite field.

Naïve non-solution for (2,2) split

- $b_1 b_2 b_3 b_4 \quad \Rightarrow \quad b_1 b_2$ and $b_3 b_4$
- Knowledge of $b_1 b_2$ potentially quite revealing. 🙁

(2,2) Secret Splitting Solution

Given a secret bit string $s = b_1 b_2 \ \dots \ b_m$

- Choose a random bit string $s_1 = r_1 r_2 \ \dots r_m$
- Compute $s_2 = x_1 x_2 \dots x_m$
  - where $x_i = (b_i \oplus r_i)$ for all i.

Recovery of secret bit $s[i]$ from $s_1[i]$ and $s_2[i]$:

- $s_1[i] \oplus s_2[i]$
- $= r_i \oplus x_i$
- $= r_i \oplus (b_i \oplus r_i)$
- $= r_i \oplus (r_i \oplus b_i)$
- $= (r_i \oplus r_i) \oplus b_i$
- $= 0 \oplus b_i$
- $= b_i$

(2,2) Secret Splitting: Correctness

- Secret can be reconstructed from shares
  - Proof: Calculation on previous slide.

(2,2) Secret Splitting: Correctness

- Secret can be reconstructed from shares
  - Proof: Calculation on previous slide.

- Neither s_1 or s_2 reveals anything about the secret
  - Proof:
    - $s_1 = r_1 r_2 \ \ldots r_m$ conveys no information. It's random.
    - $s_2 = x_1 x_2 \ldots x_m$ conveys no information. For any s_2, any value of s is possible.

(n, n) Secret Splitting Solution

Given a secret $s = b_1 b_2 \ldots b_m$

- Choose $n - 1$ random shares s_1 , s_2 , … s_{n-1}
- Construct s_n

$$s_n = s \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_{n-1}$$

Construction also works for integers $s = z_1 z_2 \ldots z_m$

- Choose $n - 1$ random shares s_1 , s_2 , … s_{n-1}
- Construct s_n

$$s_n = s - (s_1 + s_2 + \ldots + s_{n-1}) \bmod q$$

(n, t) Sharing: Using Splitting

- (n, t)-shares built using shares from (L, L)-splitting.
- Each (n, t)-share is a set of (L, L)-shares.
  - Union of t (n, t)-shares contains all of the (L, L)-shares
    - So t (n, t)-shares suffices to recover secret.
  - Union of $t - 1$ or fewer (n, t)-shares omits at least one (L, L)-share.
    - So $t - 1$ or fewer (n, t)-shares reveals nothing about the secret.

Building (n, t)-shares

- Construct (L, L)-split $s \Rightarrow s_1, s_2, \dots s_L$ where $L = \binom{n}{t-1}$
- Construct subsets $P_1, P_2 \dots P_L$ of $\{1, 2, \dots n\}$ with $|P_i| = t - 1$.
  - Elements of each P_i identify a set $\{hs_1^i, hs_2^i, \dots\}$ of (n, t)-shares
  - Should not be possible to reconstruct s using only (n, t)-shares identified in P_i or in a subset of P_i. [Defn of (n, t) secret sharing]
- Define each hs_j^i is a set of (L, L)-shares
  - Should not be able to reconstruct s using (L, L)-shares contained in (n, t)-shares $\{hs_1^i, hs_2^i, \dots\}$ for any P_i.
  - Associate the share s_i from (L, L)-split with P_i:

$$hs_j^i \in P_i \text{ if and only if } s_i \notin hs_j^i$$

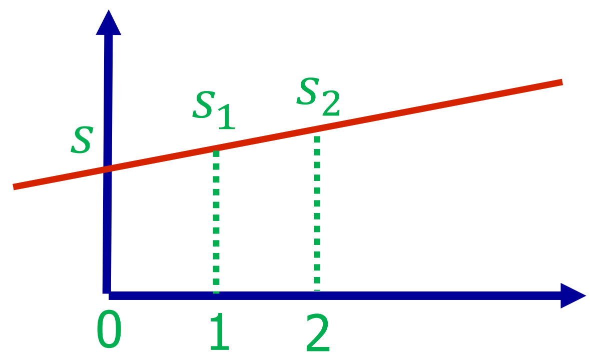
Building (n, t)-shares: Example

(4,2) sharing of s :

- $L = \binom{n}{t-1} = \binom{4}{1} = \frac{4!}{1!(4-1)!} = 4$
- Create (L, L) split $s \Rightarrow s_1\ s_2\ s_3\ s_4$

hs_1	$\{s_2, s_3, s_4\}$
hs_2	$\{s_1, s_3, s_4\}$
hs_3	$\{s_1, s_2, s_4\}$
hs_4	$\{s_1, s_2, s_3\}$

$(n, 2)$-sharing Direct Implementation



- Infinite number of lines intersect $(0, s)$.
- A line $y = f(x)$ is a sharing of s if that line intersects $(0, s)$
  - Any point $(x, f(x))$ is a share.
    - Infinite number of lines pass through a share $(x_i, f(x_i))$.
  - $f(x)$: $mx + b$ can be recovered from (only!) 2 shares
    - y intercept s can be recovered: It's b

(n,t)-sharing: Polynomials [Shamir 79]

Facts about (t-1)-degree polynomials:

$$f(x)\colon\ a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \ldots + a_0$$

- $(0, a_0)$ satisfies $f(x)$.
- An infinite number of polynomials are satisfied by $(0, a_0)$.
- Unique polynomial $f(x)$ can be recovered from t points.
  - Construct LaGrange Interpolating polynomial.
- $t-1$ or fewer points defines an infinite number of polynomials.

(n, t)-sharing: Direct Implementation

(n, t)-sharing of s:

- Choose a random $t - 1$ degree polynomial where $f(0) = s$.
- Calculate shares …
  - s_1: $(1, f(1))$, s_2: $(2, f(2))$, …, s_n: $(n, f(n))$,

Verifiable Secret Sharing (VSS)

Given (n, n) secret splitting

$$s \Rightarrow s_1 \; s_2 \; s_3 \; \ldots \; s_n$$

Is $\hat{s}$ one of those shares or a bogus share?

Soln: Add information to each share s_i:

$$\langle \, s_i \, , i, a, a^s, a^{s_1} \, , \, \ldots \, , \, a^{s_n} \, \rangle$$

where a is generator for a large finite field, so

- $\langle \, i, a, a^s, a^{s_1} \, , \, \ldots \, , \, a^{s_n} \, \rangle$ reveals nothing about s, s_1 , ... s_n.

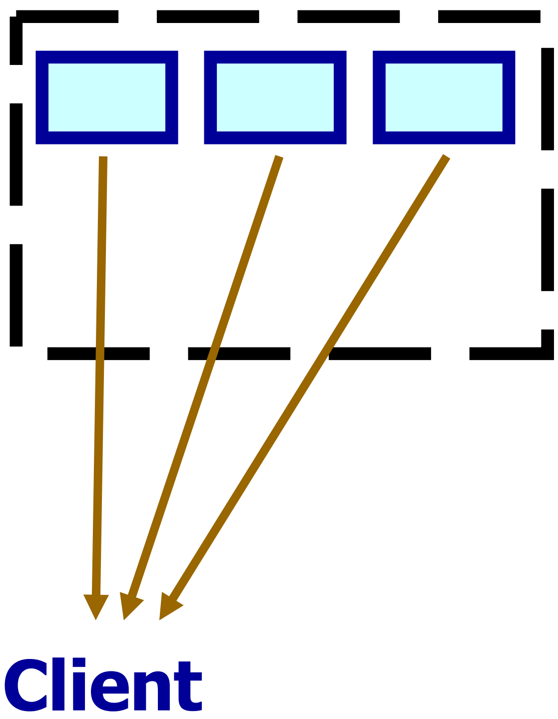
VSS Checks

How to check $\langle\, s_i\,, i, a, a^s, a^{s_1}, \ldots, a^{s_n} \,\rangle$?

- Is it a share from a splitting of s?
  - Compute and check: $a^s = a^{s_1} \cdot \ldots \cdot a^{s_n}$?
  - … simplifies to: $a^s = a^{(s_1 + s_2 + \ldots + s_n)}$?
  - If true then $a, a^s, a^{s_1}, \ldots, a^{s_n}$ from a sharing of s.

- Is s_i the i^{th} share?
  - Compute a^{s_i} using s_i and (public) a.
  - Compare a^{s_i} with a^{s_i} value found in check vector.

Back to Replication…



Authentication protocol so client can distinguish and synthesize responses from different servers.

- Signing key for each server?
- Signature verification key for service?

(n, t)-Function Sharing: Definition

Let s-$F(x)$ be a function that depends on secret s and on argument x.

$(\boldsymbol{n}, \boldsymbol{t})$-Function Sharing for $\boldsymbol{s}$-$\boldsymbol{F}(\boldsymbol{x})$

- Can compute s-$F(x)$ for any x by using t or more shares s_i from a sharing of s.
- No information about s-$F(x)$ can be deduced by using fewer than t shares s_i from a sharing of s.

(n, t)-Function Sharing: Implement

(n, t)-Function Sharing for s-$F(x)$

- $s \Rightarrow s_1, s_2, \ldots, s_n$
- Compute $partial_i = g(s_i, x)$
- Compute $result := Comb(partial_1, \ldots partial_t)$

- $g(\cdot, \cdot)$ and $Comp(\ldots)$ depend on s-$F(x)$.
- Not all functions can be shared.
  - RSA digital signatures and decryption can be shared.

(n, t)-Function Sharing: Example

Define s-$sign(m)$: m^s

$s = (s_1 + s_2) \bmod p$

$g(s_i, m)$: m^{s_i} $\quad$ $Comp(ps_1, ps_2)$: $ps_1 \times ps_2$

$--------------------------------------$

$Comp(ps_1, ps_2)$ ...

$= Comp\big(g(s_i, m), g(s_i, m)\big)$

$= Comp(m^{s_1}, m^{s_2})$

$= m^{s_1} \times m^{s_2}$

$= m^{s_1 + s_2}$

$= m^s$

Proactive Secret Sharing (PSS)

Mobile adversary accumulates shares of secret. Even if at most one server is compromised at any time, a majority of shares still eventually compromised.

Defense: Periodically re-share key.

- Create new, independent sharing of key.
- Replace old shares with new shares.

PSS Requirements

Given: sharing $s_1, s_2, \dots, s_n$ of secret s.

Goal: Compute a new sharing $u_1, u_2, \dots, u_n$ of secret s where:

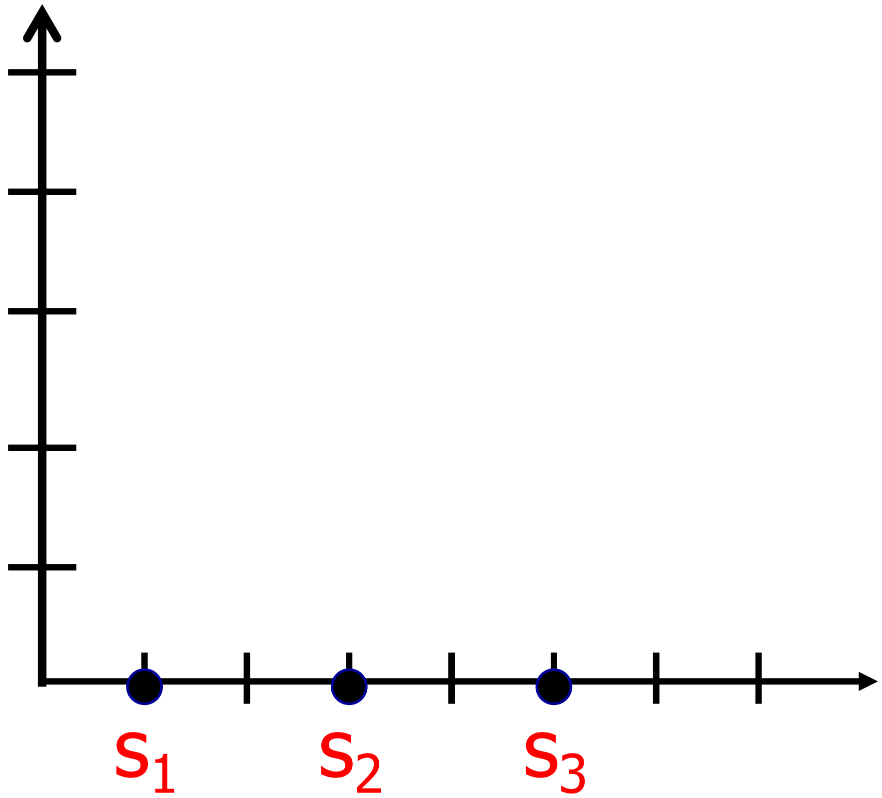
> Fewer than t old shares $s_1, s_2, \dots, s_n$ cannot be combined with fewer than t new shares $u_1, u_2, \dots, u_n$ to learn anything about secret s.

Obvious solution: Compute s from shares; calculate a new sharing for s.

Obvious problem: Materializing s risks compromise.

PSS for Splitting via Splitting

$s = s_1 + s_2 + s_3$

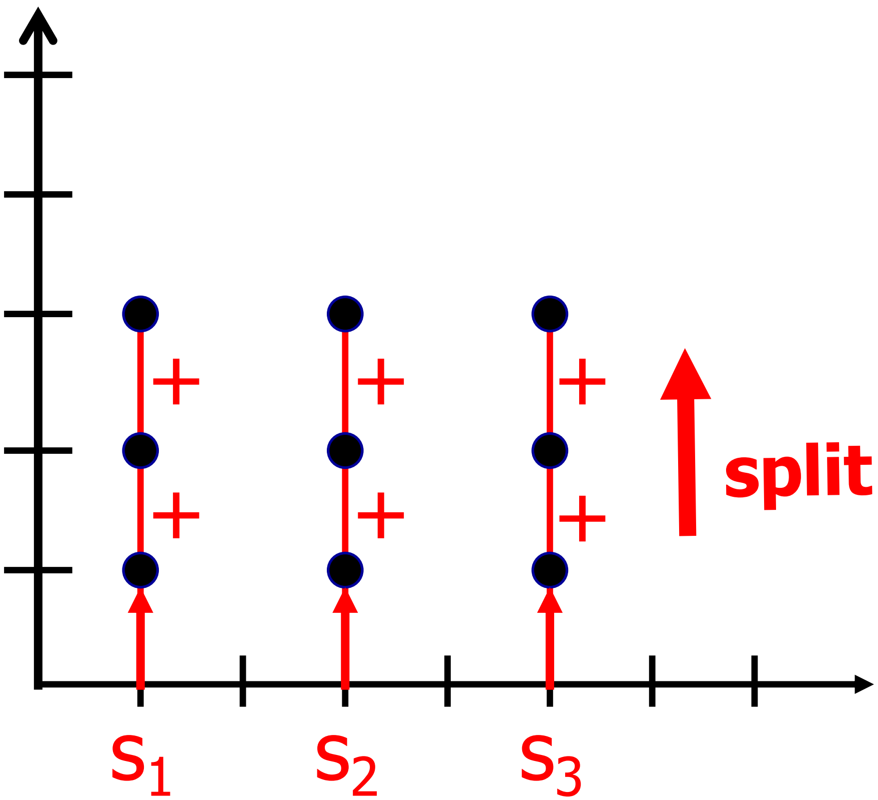


old sharing

old share: s_i

PSS for Splitting via Splitting

$s = s_1 + s_2 + s_3$



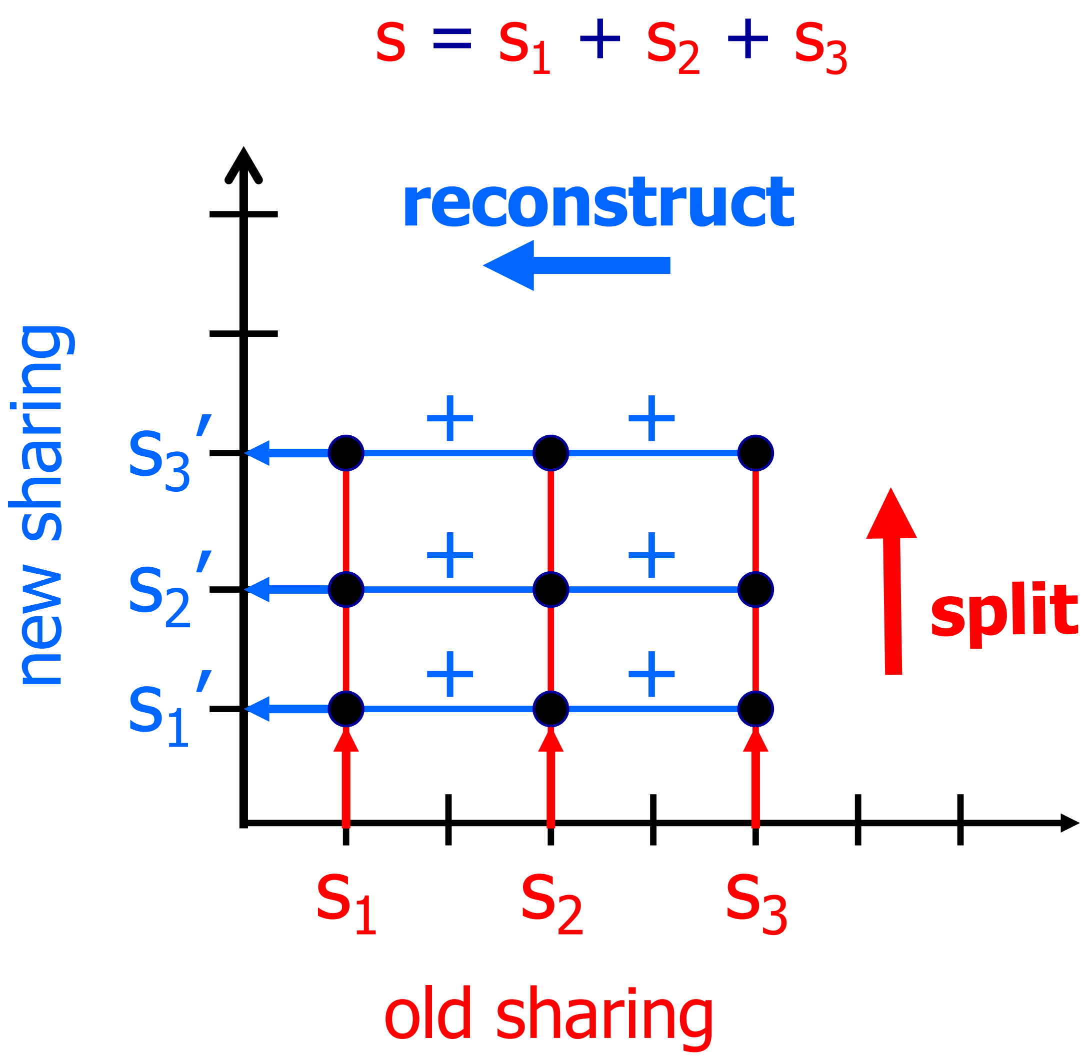
old sharing

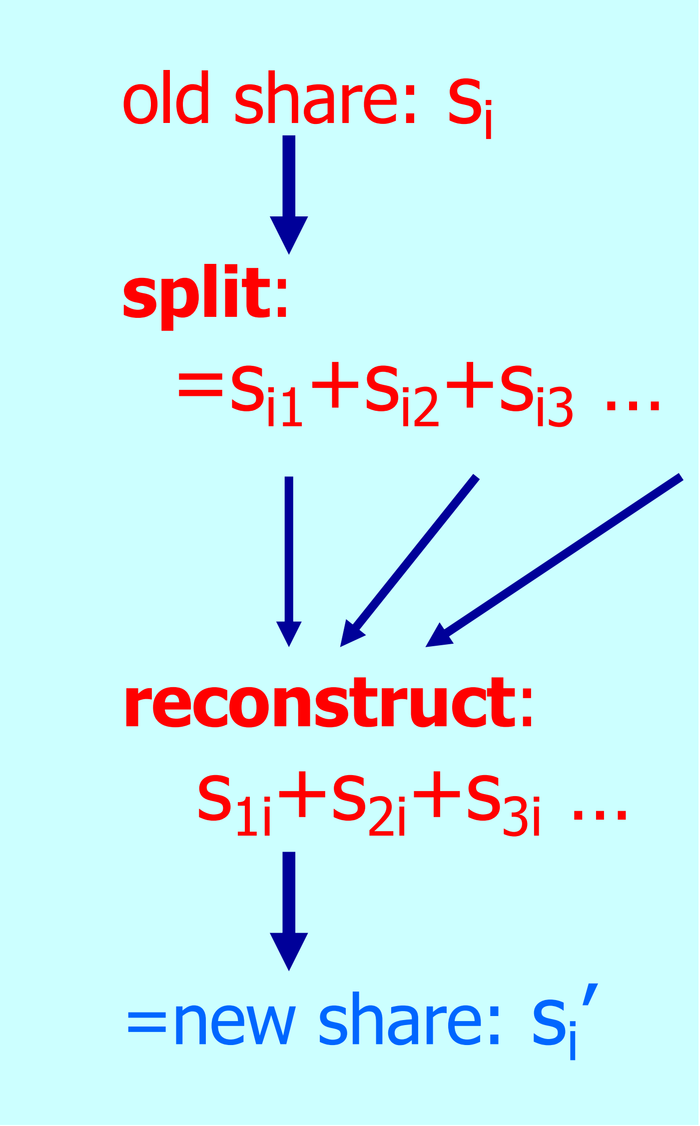
old share: s_i

split:

$= s_{i1} + s_{i2} + s_{i3}$ ...

PSS for Splitting via Splitting





PSS for Polynomial Secret Sharing

(n, t)-sharing of s using a $(t-1)$-degree polynomial:

$$f(x):\ a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \ldots + a_0$$

where

$$f(0) = s,\ \ f(1) = s_1,\ \ f(2) = s_2,\ \ f(3) = s_3\ , \ldots$$

Goal: Find a new $(t-1)$-degree polynomial $g(x)$:

$$g(0) = s,\ g(1) = u_1,\ g(2) = u_2,\ g(3) = u_3\ , \ldots$$

Adding a Random Function to $f(x)$

To re-share secret $f(0) = s$, each share s_i holder invents a random $(t-1)$ - degree polynomial that is a sharing for 0:

$$f_i(x){:}\ \ a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \ldots + a_1 x + 0$$

Polynomial $g(x)$ is a re-sharing of $f(0) = s$:

$$g(x){:}\ \ f(x) + f_1(x) + f_2(x) + \ldots + f_n(x)$$

Dissemination of the $f_i(x)$

$$g(x):\ \ f(x) + f_1(x) + f_2(x) + \ \ldots + \ \ f_n(x)$$

Suffices to distribute (using secure channels)

1 → j: Enc($f_1(1)$)

2 → j: Enc($f_2(2)$)

…

(n, t) Secret Sharing: Summary

(n, t): n pieces, where t suffice to reconstruct.

- (n, n) secret splitting
- (n, t) using (n, n) secret splitting
- (n, t) using polynomials
- verifiable secret sharing
- function sharing
  - … authentication of replica responses
- proactive secret sharing