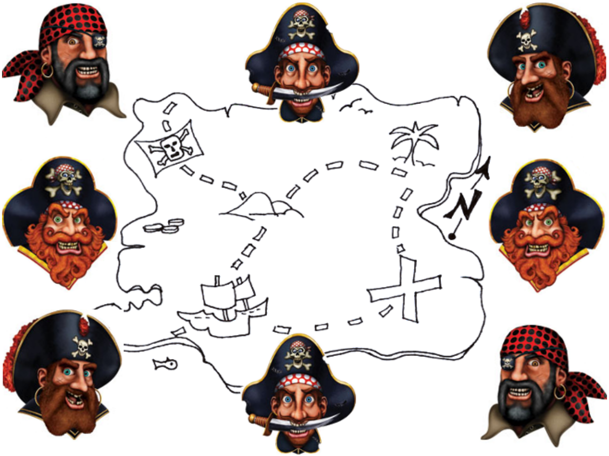
Scalable Rational Secret Sharing

Dani et al. University of New Mexico

Talk by Tanja Werthmüller

April 04, 2012

Starting Example



Starting Example

Setting

- A secret should be divided into different shares.

Setting

- A secret should be divided into different shares.
- The shares are distributed among the players.

Setting

- A secret should be divided into different shares.
- The shares are distributed among the players.
- Each share on its own should not reveal the secret.

Setting

- A secret should be divided into different shares.
- The shares are distributed among the players.
- Each share on its own should not reveal the secret.
- Combining all the shares reconstructs the secret.

Setting

- A secret should be divided into different shares.
- The shares are distributed among the players.
- Each share on its own should not reveal the secret.
- Combining all the shares reconstructs the secret.
- The players are selfish and rational.

Setting

- A secret should be divided into different shares.
- The shares are distributed among the players.
- Each share on its own should not reveal the secret.
- Combining all the shares reconstructs the secret.
- The players are selfish and rational.
- Each player prefers to
  1. learn the secret by him self
  2. learn the secret together with other
  3. not learn the secret at all.

Starting Example

Secure Secret Sharing

A **secure secret sharing scheme** distributes shares so that anyone with fewer than n shares has no additional information about the secret than someone with no shares at all.

Classical secure secret sharing Scheme

Shamir's Scheme: A polynomial of degree $n-1$ can be reconstructed using n points.

Classical secure secret sharing Scheme

Shamir's Scheme: A polynomial of degree $n-1$ can be reconstructed using n points.

- Encode the secret as the first coefficient of a random polynomial f of degree $n-1$
- The shares are points $(i, f(i))$ on the polynomial
- Any n players can reconstruct the polynomial using interpolation

The Problem with Selfish Players

A selfish player will never send his share to the other players, as he has to fear, that he will not get the other players shares back.

The Problem with Selfish Players

A selfish player will never send his share to the other players, as he has to fear, that he will not get the other players shares back.

$\Rightarrow$ **We need to adapt the protocol.**

Solution

Our Goals

- If our group of n agents follow the protocol they will **all** learn the secret.

Our Goals

- If our group of n agents follow the protocol they will **all** learn the secret.
- No player can improve substantially by deviating from the protocol.

Our Goals

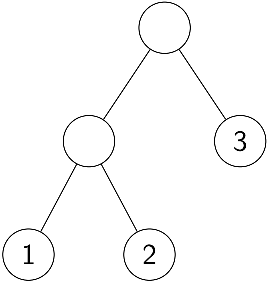
- If our group of n agents follow the protocol they will **all** learn the secret.
- No player can improve substantially by deviating from the protocol.
- The protocol is scalable:
  - message complexity per agent: $O(1)$
  - time complexity: $O(\log |agents|)$

Outline

- Dealer's Protocol
  - is only active at the beginning
  - prepares the input for the player's protocol
- Player's Protocol
  - is played in rounds
  - in round X the secret is revealed

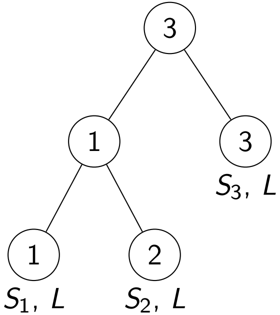
Dealer's Protocols

Dealer's Protocol



- assign all players to leaves

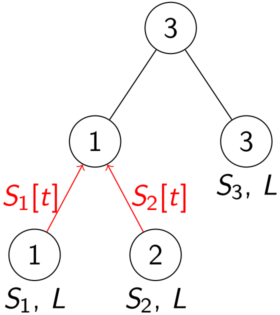
Dealer's Protocol



- assign all except for one player to remaining nodes
- give each player his *share list* S_i and a *list of potential secrets* L

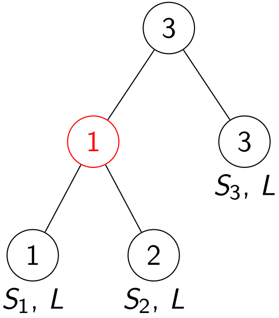
Player's Protocols

Player's Protocol – Up-Stage



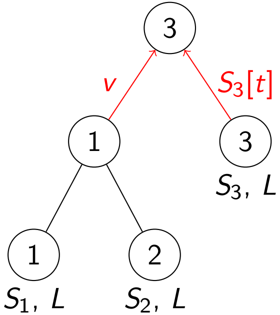
- children send their shares to parent

Player's Protocol – Up-Stage



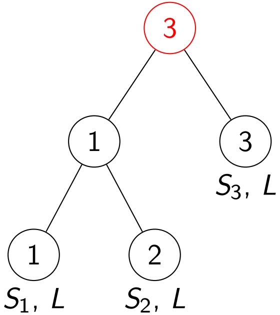
- verify $S_1[t]$
- verify $S_2[t]$
- construct new share v

Player's Protocol – Up-Stage



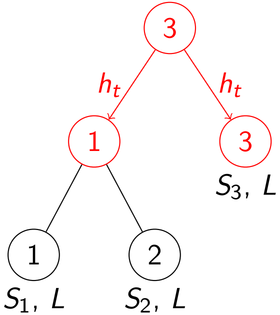
- children send their shares to parent

Player's Protocol – Up-Stage



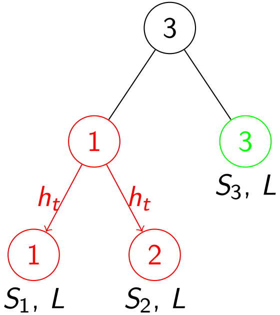
- verify v
- verify $S_3[t]$
- construct h_t

Player's Protocol – Down-Stage



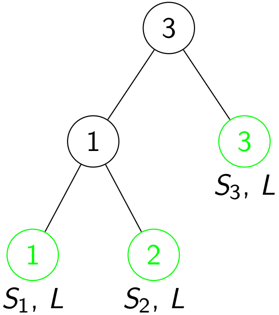
- send h_t to children

Player's Protocol – Down-Stage



- verify h_t
- send h_t to children

Player's Protocol



- verify h_t
- all players know h_t
- if $h_t = 0$ then $t = X$

Analysis

Truthfulness

There are only two ways in which a player can deviate from the protocol:

1. Send fake messages
2. Leave the protocol before the secret was revealed

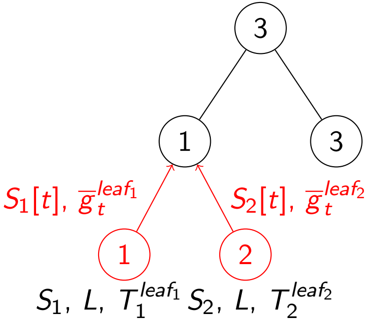
Verification by Tag Values

- Dealer prepares
  - T_i^w: a tag list for the sending node w
  - $H_j^{w',w}$: a list of verification tokens for the receiving node w'
- Node w sends a tag $\overline{g}$ from T_i^w along with its share v.
- Node w' asserts

$$c = a \cdot v + b \cdot \overline{g}$$

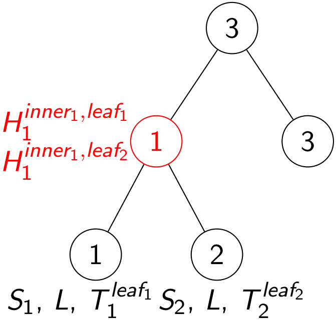
  where $(a, b, c) \in H_j^{w',w}$.
- All entries in the lists S, T and H are elements taken from a finite field F_q.

Verification by Tag Values



- children send their shares and tags to parent

Verification by Tag Values



- verify that $c_t^{leaf_i} = a_t^{leaf_i} \cdot S_i[t] + b_t^{leaf_i} \cdot \overline{g}_t^{leaf_i}$
- construct new share v

Send a Fake Message

Proposition (3.1)

The probability that a faked message will satisfy the verification function is $\frac{1}{q-1}$.

Send a Fake Message

Proposition (3.1)

The probability that a faked message will satisfy the verification function is $\frac{1}{q-1}$.

- Send a fake message v' with the corresponding g' (not known by the sender).

Exit Protocol

If at any time during a round t of the player's protocol some player i catches some other player cheating, i outputs the current secret $L[t]$ and leaves the protocol.

Leave the Protocol

- Don't transmit the fact $h_t = 0$.
- Guess the real secret with a sufficiently high probability.

Leave the Protocol

Lemma (3.2)

A player deviating from the protocol cannot increase his expected payoff by more than ϵ unless his probability of successfully learning the secret by deviating is at least $p = \frac{(U - U_- + \epsilon)}{(U_+ - U_-)}$.

Lemma (3.3)

A player who initially received a list of length α has at most $\frac{1}{\alpha - t}$ chance of (correctly) guessing the position of the secret on round t if it has not already been revealed.

Ensuring Trustfulness

We need $p \geq \frac{(U - U_- + \epsilon)}{(U_+ - U_-)}$ for cheating to be profitable and we know that $p = \frac{1}{\alpha - t}$:

Ensuring Trustfulness

We need $p \geq \frac{(U - U_{-} + \epsilon)}{(U_{+} - U_{-})}$ for cheating to be profitable and we know that $p = \frac{1}{\alpha - t}$:

$$\frac{1}{\alpha - t} \geq \frac{(U - U_{-} + \epsilon)}{(U_{+} - U_{-})}$$

Ensuring Trustfulness

We need $p \geq \frac{(U - U_{-} + \epsilon)}{(U_{+} - U_{-})}$ for cheating to be profitable and we know that $p = \frac{1}{\alpha - t}$:

$$\frac{1}{\alpha - t} \geq \frac{(U - U_{-} + \epsilon)}{(U_{+} - U_{-})}$$

$$\alpha - t \leq \frac{(U_{+} - U_{-})}{(U - U_{-} + \epsilon)}$$

Ensuring Trustfulness

We need $p \geq \frac{(U-U_- +\epsilon)}{(U_+ - U_-)}$ for cheating to be profitable and we know that $p = \frac{1}{\alpha - t}$:

$$\frac{1}{\alpha - t} \geq \frac{(U - U_- + \epsilon)}{(U_+ - U_-)}$$

$$\underbrace{\alpha - t}_{>Y} \leq \frac{(U_+ - U_-)}{(U - U_- + \epsilon)}$$

Ensuring Trustfulness

We need $p \geq \frac{(U - U_- + \epsilon)}{(U_+ - U_-)}$ for cheating to be profitable and we know that $p = \frac{1}{\alpha - t}$:

$$\frac{1}{\alpha - t} \geq \frac{(U - U_- + \epsilon)}{(U_+ - U_-)}$$

$$\underbrace{\alpha - t}_{>Y} \leq \frac{(U_+ - U_-)}{(U - U_- + \epsilon)}$$

$\Rightarrow$ Choose padding $Y \geq \frac{(U_+ - U_-)}{(U - U_- + \epsilon)}$ to ensure truthfulness.

Proof of Efficiency

- The expected number of messages sent by each player is $O(1)$.
- The expected number of bits sent is $O(log(q))$.
- The expected overall latency is $O(log(n))$.

Conclusions

Summary

- An algorithm for rational secret sharing, where no player can improve substantially by deviating from the protocol.
- The mechanism is scalable in terms of latency and number of messages sent by each player.

Summary

- An algorithm for rational secret sharing, where no player can improve substantially by deviating from the protocol.
- The mechanism is scalable in terms of latency and number of messages sent by each player.

	Scalable RSS	**Previous Work**
messages per player per round	$O(1)$	$O(n)$
latency per round	$O(log\ n)$	$O(n)$
E[# rounds]	$O(1)$	$O(n)$

Questions?