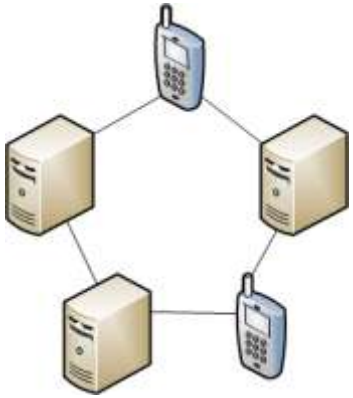
University of Duisburg-Essen
Bismarckstr. 90
47057 Duisburg
Germany

Detecting Opportunistic Cheaters in Volunteer Computing

Matthäus Wander, Torben Weis, Arno Wacker

July 31, 2011

Motivation



- Distributed computing platform
- Embarrassingly parallel application
- Volunteering untrusted users
- Incentive: virtual credits or competitive ranking
- Cheaters obtain reward by fraud
- Usual approach: redundant recomputation

Goal: Detect cheaters efficiently

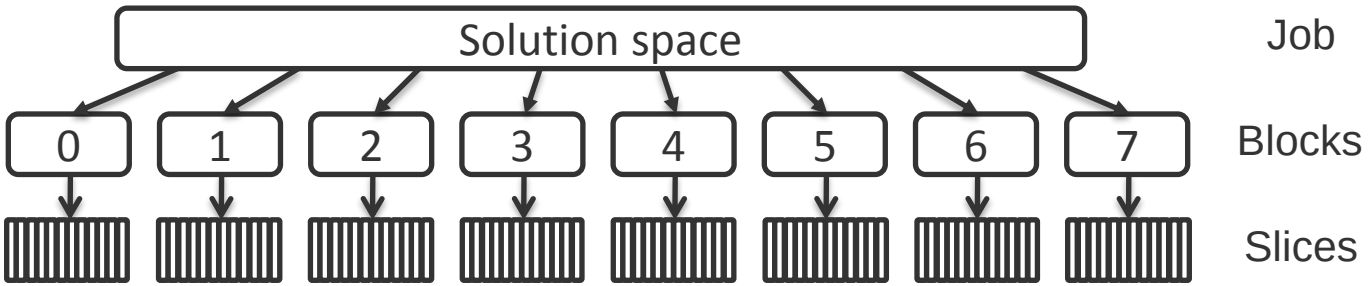
Cheating

Cheating: using a different algorithm than intended, eventually leading to wrong result

- **Two types of cheaters**
  - Opportunistic (invest least, gain most)
  - Malicious (disturb most)
- **May occur with different frequency**
  - We detect specifically opportunistic cheaters

Example Application

- **Brute-force attack on symmetric-key cipher**
- **Represents class of search problems**
- **Solution space: all possible keys**
  - Decrypt given ciphertext with key
  - Rate result with score function
- **Group job slices into blocks**



Challenges and Idea

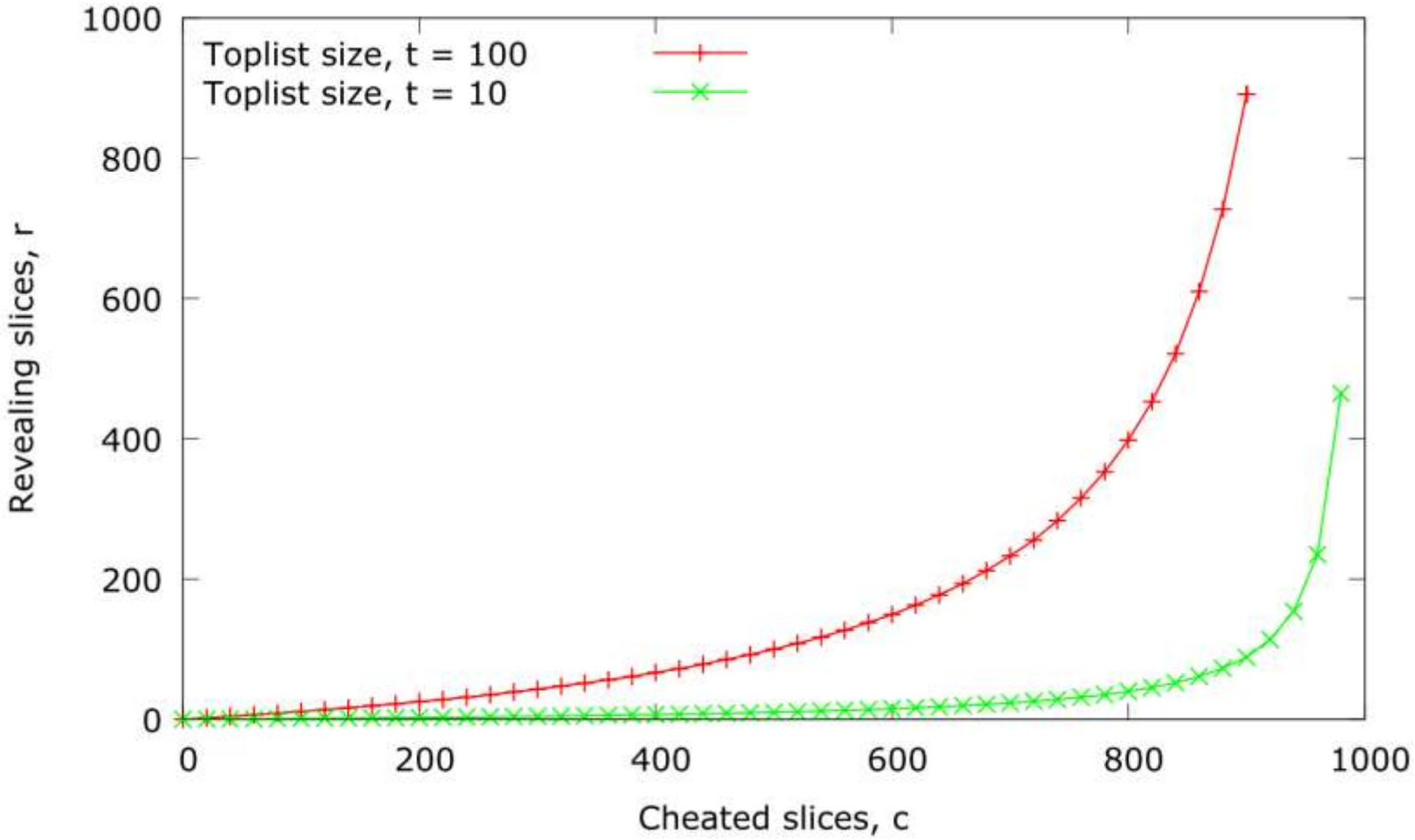
- **Block result shall be unlikely to guess**
  - Hit/miss not suitable
- **Cope with limited bandwidth**
  - List of all slice scores not suitable
- **List of *t* top slices per block**
- **Use sample testing to verify subset of slices**

Detection Approach

- **Positive verification**
  - Recalculate top list
  - Cheater detected, if any of them is wrong
- **Negative verification**
  - Select random subset of other slices
  - Cheater detected, if any of them achieves top score
- **Smart opportunist calculates part of block**
  - Top list seems valid
  - Only some slices can reveal cheater

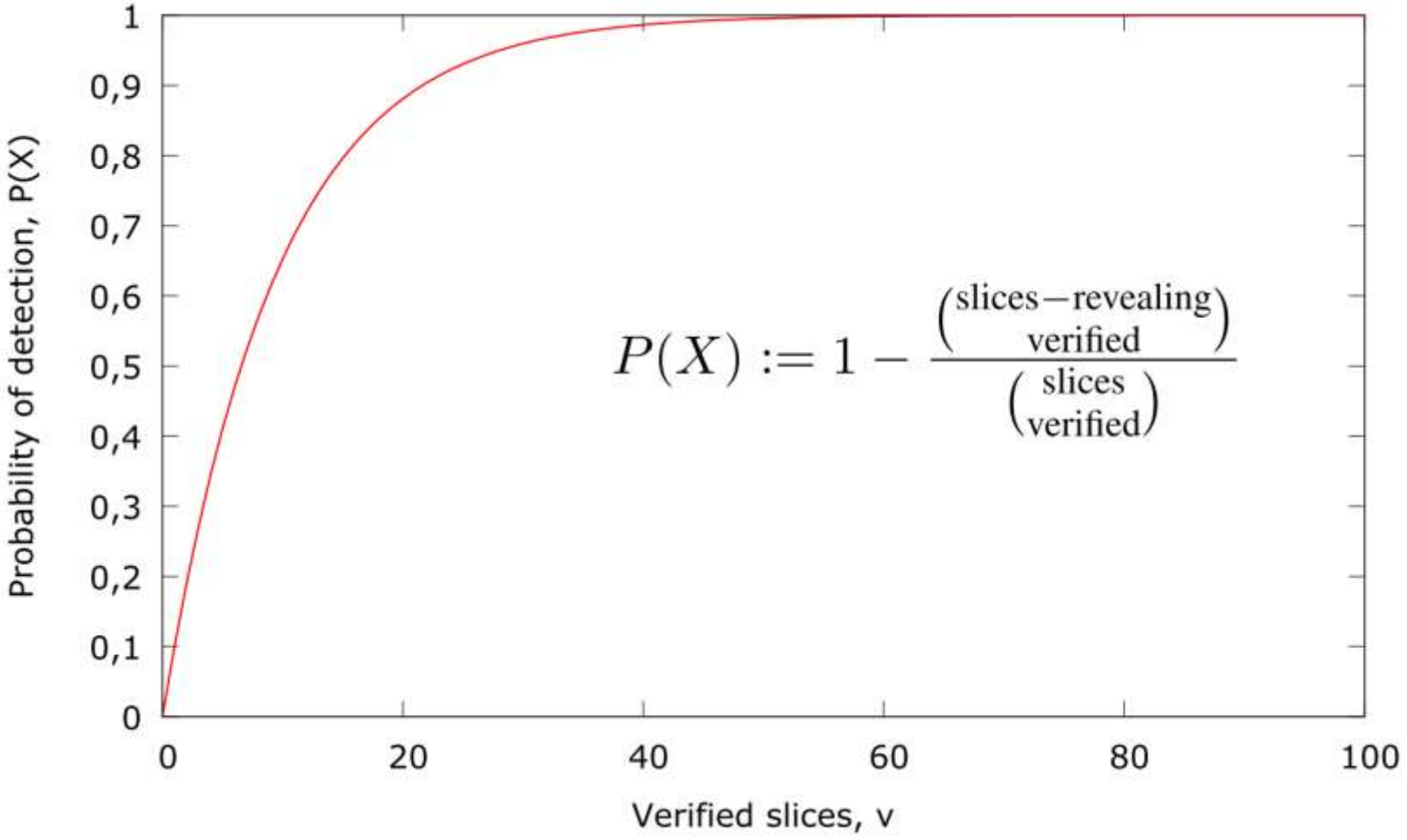
Revealing Slices

- **1000 slices per block, each with different score value**



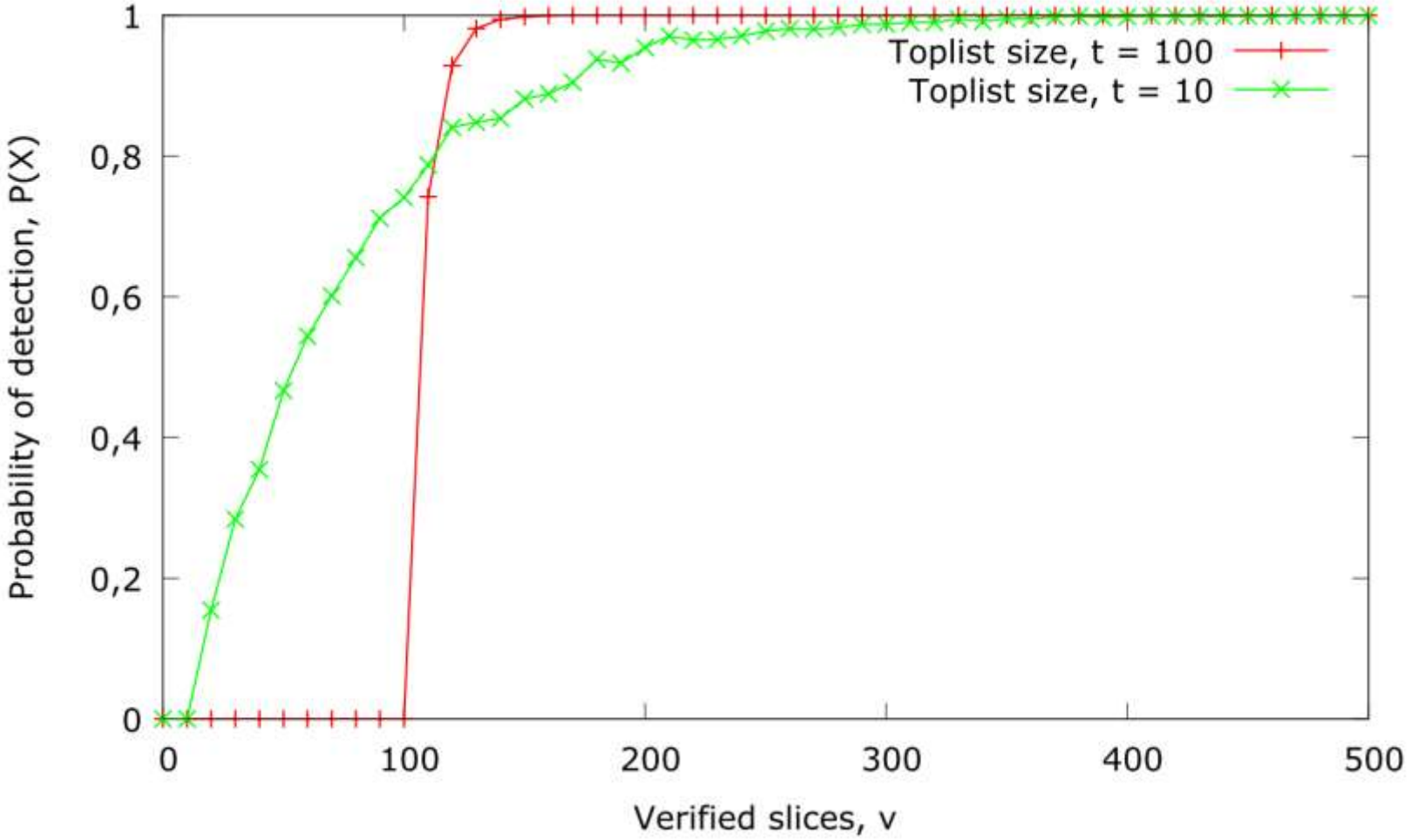
Sample Testing Effectiveness

- **1000 slices per block, 10% revealing**



Detection Probability

- **1000 slices per block, 50% cheated**



Example Verification

- **10^7 slices per block, 50% cheated**
- **1000 top list entries (0.01%)**
- **$5 \cdot 10^4$ verified slices (0.5%) → P = 0.9945**
- **10^5 verified slices (1%) → P = 1**
- **Estimated data size of top list: 16 KB**
  - 8 bytes identifier
  - 8 bytes score value

Summary and Outlook

- **Efficient detection of opportunistic cheaters**
- **Sample testing with result aggregation**
- **Interdependency between computational complexity and required bandwidth**
- **Outlook**
  - Analyze suitable score functions
  - Probabilistic voting to catch malicious cheaters
  - Use checkpointing in sequential applications

Related Work

- **Trust/credibility systems**
  - [Sarmenta2001], [Golle2002], [Germain-Renaud2003]
- **Bait approaches**
  - [Golle2001], [Szajda2003], [Zhao2005], [Du2005]
- **Commit-based sampling**
  - [Du2004]
- **Checkpointing**
  - [Monrose1999], [Domingues2007]

Impact of Parameters

- Cheated slices: chosen by cheater, e.g. 50%
- Slices per block: constrained, e.g. 1000
- Top list size
- Number of verified slices