Privacy and Reliability in an Untrusted Cloud

A private and secure cloud

Distributing computation onto untrusted machines.

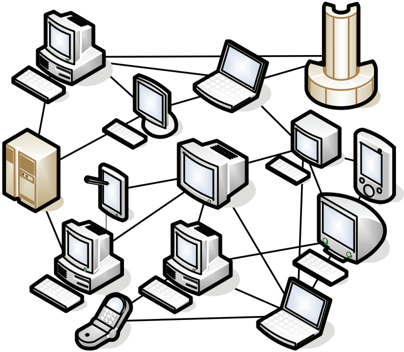
Today's focus on privacy: sTile

sTile

A technique for privately solving computationally-intensive problems (3-SAT) on untrusted computers.

Our approach: intelligent distribution

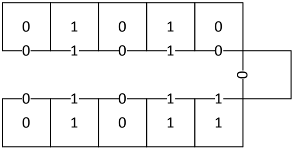
Obstacle: Private computation is hard and inefficient [Childs 2005; Gentry 2009].



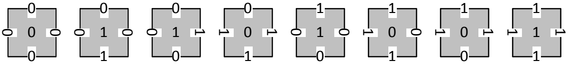
Solution:
1. Divide computation into elemental subcomputations.
2. Distribute subcomputations onto network.

Computing with tiles

Input:



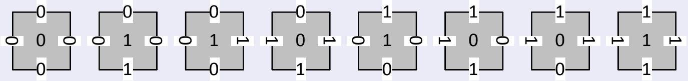
Program:



Computation: Copies of the program tiles self-attach to the input.

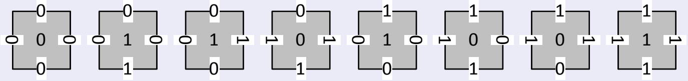
Addition with tiles

adding program

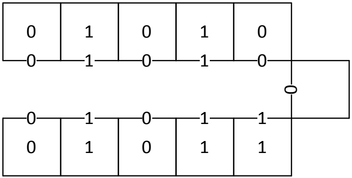


Addition with tiles

adding program

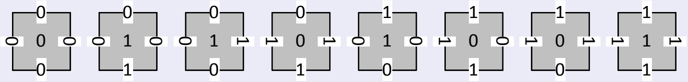


Encode input to add 10 $(= 1010_2)$ and 11 $(= 1011_2)$

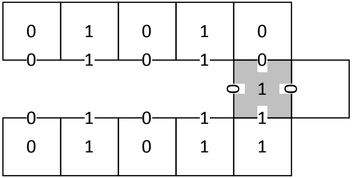


Addition with tiles

adding program

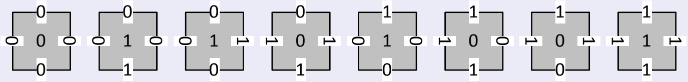


Add the two least significant bits

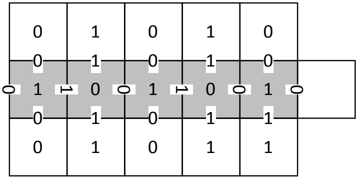


Addition with tiles

adding program

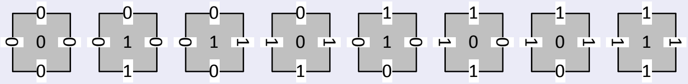


Add the rest of the bits, one at a time: $10 + 11 = 21 (= 10101_2)$

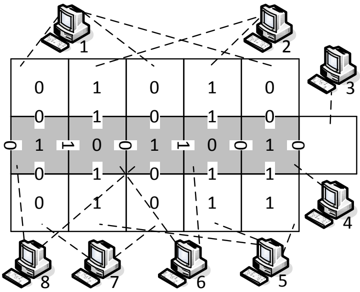


Addition with tiles

adding program

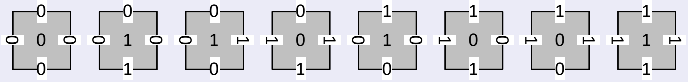


Suppose computers deployed tiles

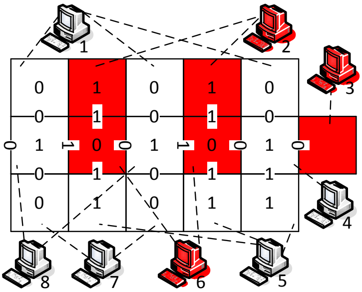


Addition with tiles

adding program

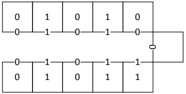


Even if some were compromised, they couldn't learn private data

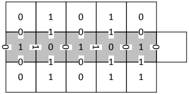


3-SAT with tiles [Winfree 1998]

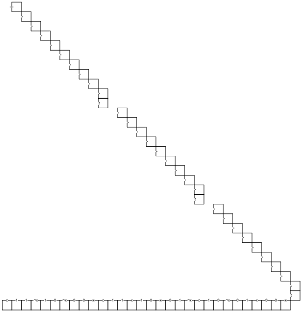
Addition [TCS'07]



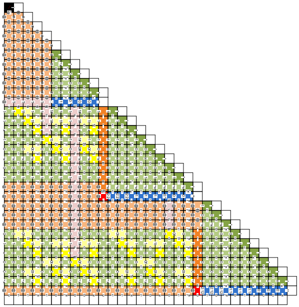
→



3-SAT [Nat.Comp.'12]

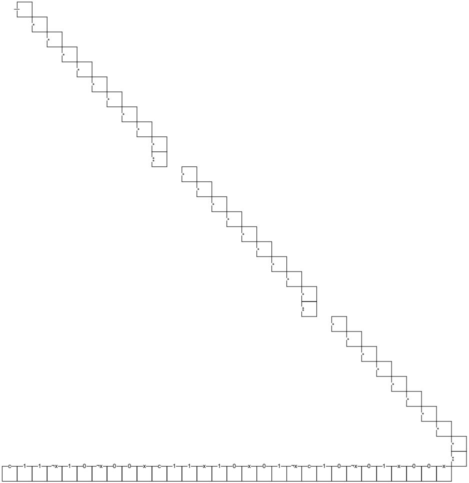


→



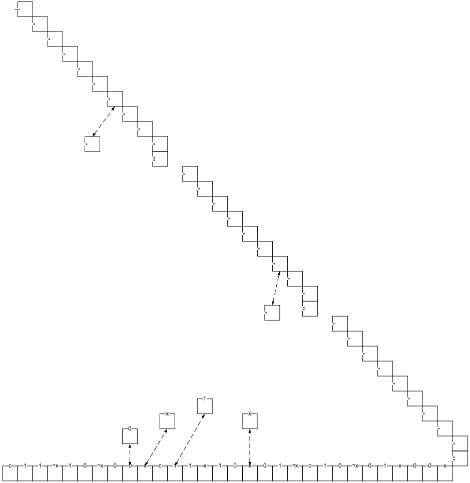
sTile intuition: computers simulate tiles

1. set up the 3-SAT seed



sTile intuition: computers simulate tiles

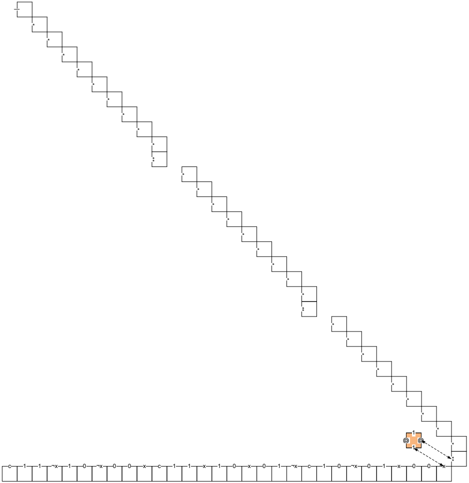
2. the seed self-replicates



discovery algorithm

sTile intuition: computers simulate tiles

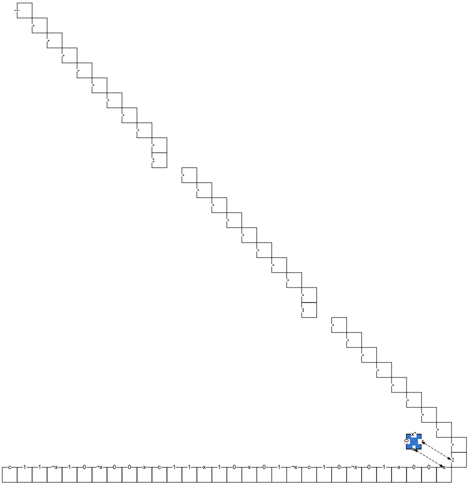
3 tiles recruit neighbors



secure multi-party computation [Yao 1986]

sTile intuition: computers simulate tiles

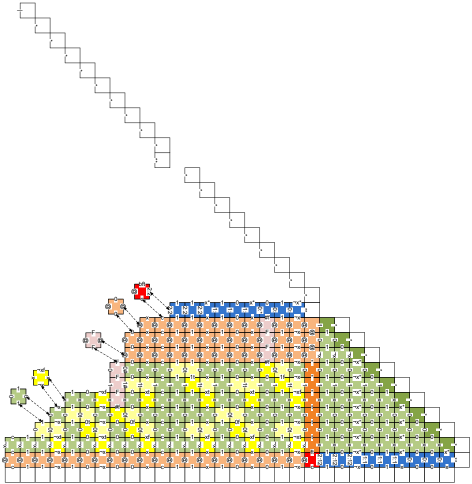
3 tiles recruit neighbors



secure multi-party computation [Yao 1986]

sTile intuition: computers simulate tiles

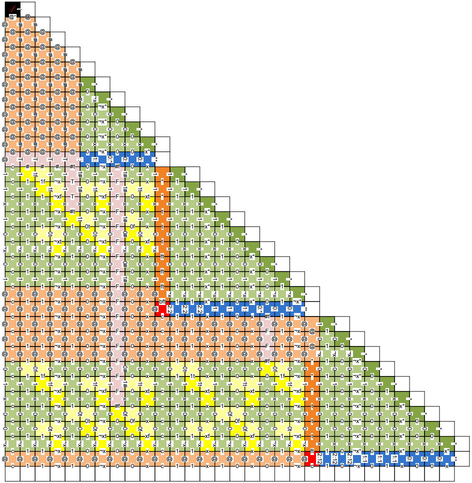
3. tiles recruit neighbors



secure multi-party computation [Yao 1986]

sTile intuition: computers simulate tiles

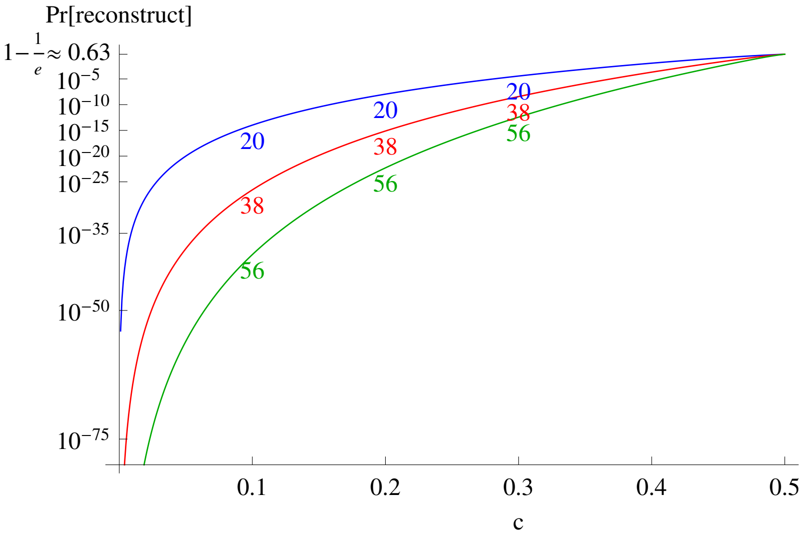
4. report solution to the client



Evaluation plan

- Formally prove privacy
- Empirically demonstrate robustness to network delay
- Empirically demonstrate scalability

Probability of reconstructing a 20-, 38-, and 56-bit input



sTile provides highly-probable privacy

Threat model:

A Byzantine fraction of the cloud attempts to reconstruct private data.

sTile guarantee:

$$\boxed{P_{\text{compromise}}(c, n, s) = 1 - (1 - c^n)^s}$$

c — compromised fraction n — bits in input s — number of seeds

TeraGrid example

Controlling $\frac{1}{8}$ of TeraGrid's 100,000 machines yields a probability of 10^{-10} of data compromise of a 17-variable formula.

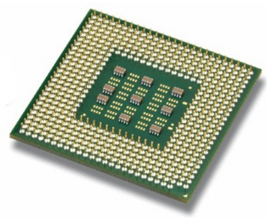
Experimental Setup

- Mahjong: sTile implementation framework
  - Java, 3K LoC, builds on Prism-MW [Malek et al. 2005]
  - Input: NP-c problem instance P
  - Output: Distributed software system to solve P
  - Download: http://www.cs.umass.edu/~brun/Mahjong

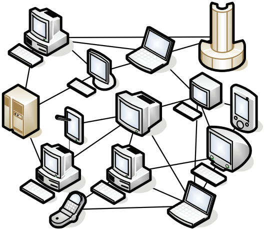
Experimental Setup

- Mahjong: sTile implementation framework
    - Java, 3K LoC, builds on Prism-MW [Malek et al. 2005]
    - Input: NP-c problem instance P
    - Output: Distributed software system to solve P
    - Download: http://www.cs.umass.edu/~brun/Mahjong
- Networks
    - 11-node private cluster (P4 1.5GHz, 512MiB, WinXP/2000)
    - 186-node USC HPCC cluster [High Performance Computing and Communications] (P4 Xeon 3GHz, Linux)
    - 100-node PlanetLab [Peterson et al. 2003] (global, varying speeds and resources)

Network Delay

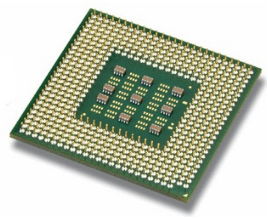


vs.

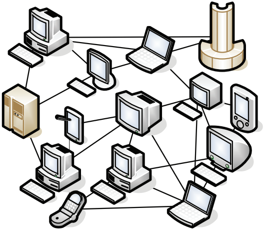


Communication is ∼100–1000 times faster in a CPU than on a network.

Network Delay



vs.



Communication is ∼100–1000 times faster in a CPU than on a network.

But latency is not throughput!

Robustness to Network Delay

Problem	# of Nodes	Network Delay	Execution Time
Mahjong			
$\mathfrak{A}$	11	Private Cluster	20.1 sec.
		HPCC	19.3 sec.
		PlanetLab	18.5 sec.
$\mathfrak{B}$	11	Private Cluster	41.6 min.
		HPCC	41.2 min.
		PlanetLab	43.9 min.
Simjong			
$\mathfrak{D}$	1,000,000	0ms	65 min.
		10ms	57 min.
		100ms	64 min.
		500ms	60 min.
		Gaussian	68 min.
		Distance-based	59 min.

Robustness to Network Delay

Problem	# of Nodes	Network Delay	Execution Time
Mahjong			
$\mathfrak{A}$	11	Private Cluster	20.1 sec.
		HPCC	19.3 sec.
		PlanetLab	18.5 sec.
$\mathfrak{D}$	1,000,000	0ms	65 min.
		10ms	57 min.
		100ms	64 min.
		500ms	60 min.
		Gaussian	68 min.
		Distance-based	59 min.

Network latency does not affect system throughput

Scalability: Speed $\propto$ Network Size

Network & Problem	# of Nodes	Execution Time	Speed-up Ratio
Private Cluster	5	43 sec.	
$\mathfrak{A}$	10	23 sec.	1.9
HPCC	93	220 min.	
$\mathfrak{C}$	186	116 min.	1.9
PlanetLab	50	9.2 min.	
$\mathfrak{B}$	100	4.8 min.	1.9
Simjong	125,000	8.7 hours	
	250,000	4.5 hours	1.9
$\mathfrak{D}$	500,000	2.1 hours	2.1
	1,000,000	64 min.	2.0

Scalability: Speed $\propto$ Network Size

Network & Problem	# of Nodes	Execution Time	Speed-up Ratio
Private Cluster	5	43 sec.	
$\mathfrak{A}$	10	23 sec.	1.9
Simjong	125,000	8.7 hours	
	250,000	4.5 hours	1.9
$\mathfrak{D}$	500,000	2.1 hours	2.1
	1,000,000	64 min.	2.0

System speed scales almost linearly with network size

Related Work

- Private computation in quantum computing through entanglement [Childs 2005]
- Homomorphic encryption for private computation [Gentry 2009]
- Plethora of non-private distributed computation work [BOINC 2009; Korpela et al. 1996; Larson et al. 2002; Rosetta@home; Dean and Ghemawat 2004; Chakravarti and Baumgartner 2004]
- . . . and fault-tolerant computation work [Sarmenta 2002; Bondavalli et al. 1993, 2002; Felber and Schiper 2001; Koren and Krishna 2007; Hwang and Kesselman 2003]
- . . . and private storage and access [Ateniese et al. 2006; Wang et al. 2011; Yang et al. 2011; Yu et al. 2010]

Contributions

sTile

- Distribution can result in privacy
- A bound on the cost of privacy

Contributions

sTile

- Distribution can result in privacy
- A bound on the cost of privacy

For more, see "Entrusting Private Computation and Data to Untrusted Networks" by Y. Brun and N. Medvidovic. In IEEE Transactions on Dependable and Secure Computing (TDSC), 10(4):225–238, 2013. `http://dx.doi.org/10.1109/TDSC.2013.13`

How do I compute a function using Byzantine machines?

How do I send you a message over a noisy channel?

Environment model

A pool of network nodes

- some nodes are Byzantine
- Byzantine node identity and rate are unknown
- nodes may join, leave, fail, and become reliable

Smart redundancy: maximize task reliability for a given resource cost

Applicable to problems with many independent subtasks that can be executed out of order.

Example

- MapReduce / Hadoop [Dean and Ghemawat 2004]
- Globus Grid Toolkit [Foster et al. 2001]
- BOINC [Korpela et al. 1996]

Applicable to problems with many independent subtasks that can be executed out of order.

Example

- MapReduce / Hadoop [Dean and Ghemawat 2004]
- Globus Grid Toolkit [Foster et al. 2001]
- BOINC [Korpela et al. 1996]

Crowdsourcing applications too

- reCAPTCHA [von Ahn et al. 2008]
- ESP Game [von Ahn and Dabbish 2004]
- FoldIt [Baker 2009]
- software verification [Schiller and Ernst 2010]
- AutoMan [Barowy et al. 2012]

Voting redundancy

Assume (for now) we know average node reliability

Voting redundancy

Assume (for now) we know average node reliability

node reliability: 0.7 desired system reliability: 0.97

Voting redundancy

Assume (for now) we know average node reliability

node reliability: 0.7 desired system reliability: 0.97

- If we ask 3 nodes, the system reliability will be: $1 - 0.3^3 - 3\left(0.3^2\right)0.7 \approx 0.84$

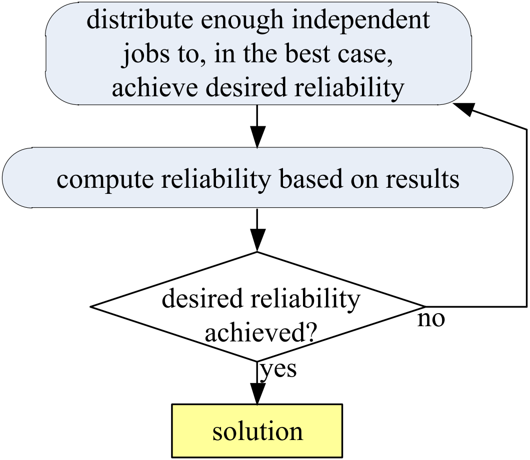
Voting redundancy

Assume (for now) we know average node reliability

node reliability: 0.7 desired system reliability: 0.97

- If we ask 3 nodes, the system reliability will be:
  $1 - 0.3^3 - 3\left(0.3^2\right)0.7 \approx 0.84$
- 19 nodes have to vote to get 0.97 reliability:
  $1 - \sum_{i=10}^{19} \binom{19}{i} 0.3^i 0.7^{19-i} \approx 0.97$

Smart redundancy



main idea: only deploy jobs if you need them

Smart redundancy example execution

answers		reliability
1	0	0.70

Smart redundancy example execution

answers			reliability
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84

Smart redundancy example execution

answers			reliability
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93

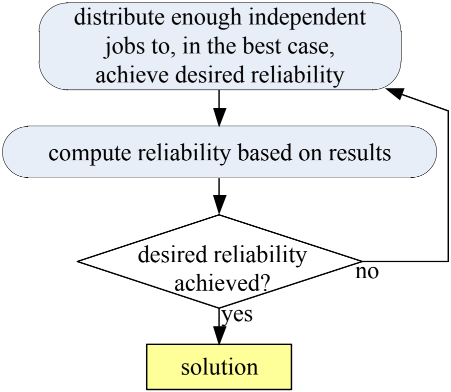
Smart redundancy example execution

answers			reliability
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84

Smart redundancy example execution

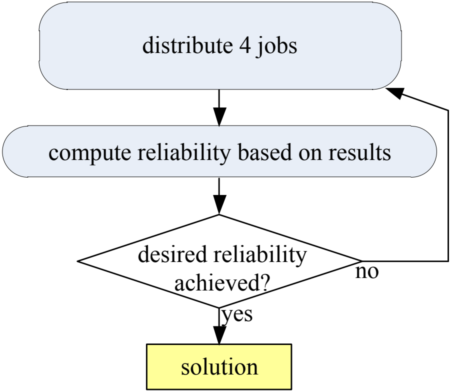
answers		reliability	
1	0		0.70
2	0	$\frac{\left(0.7^2\right)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{\left(0.7^3\right)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3\left(0.7^3\right)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{\left(0.7^4\right)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4\left(0.7^4\right)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5\left(0.7^5\right)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



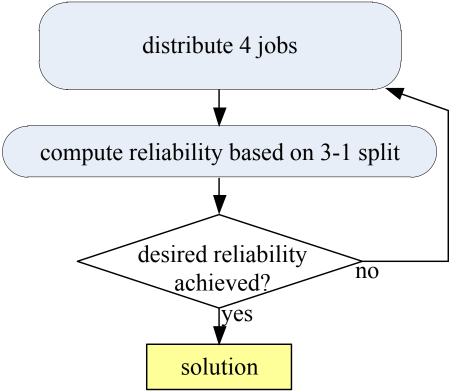
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



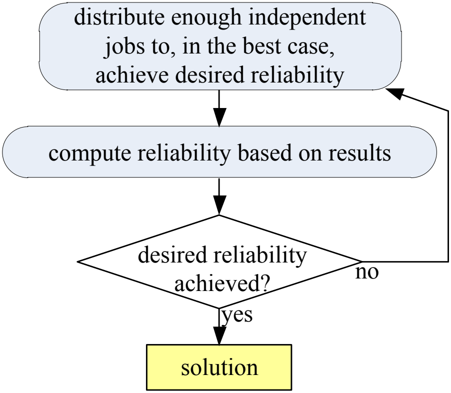
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



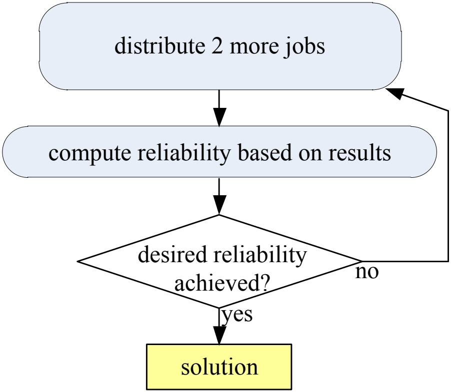
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



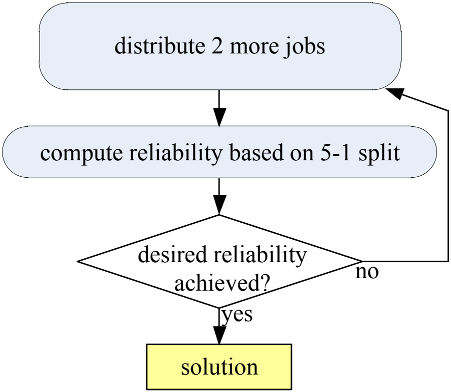
answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

Smart redundancy example execution



answers		reliability	
1	0		0.70
2	0	$\frac{(0.7^2)}{(0.7^2)+(0.3^2)}$	≈ 0.84
3	0	$\frac{(0.7^3)}{(0.7^3)+(0.3^3)}$	≈ 0.93
3	1	$\frac{3(0.7^3)0.3}{3(0.7^3)0.3+3(0.3^3)0.7}$	≈ 0.84
4	0	$\frac{(0.7^4)}{(0.7^4)+(0.3^4)}$	≈ 0.97
4	1	$\frac{4(0.7^4)0.3}{4(0.7^4)0.3+4(0.3^4)0.7}$	≈ 0.93
5	1	$\frac{5(0.7^5)0.3}{5(0.7^5)0.3+3(0.3^5)0.7}$	≈ 0.97

smart redundancy
(1) assumes best case and asks the minimum number of nodes
(2) asks more after learning how reality differs from best case.

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

$$\frac{\binom{1004}{504}\left(0.7^{504}\right)0.3^{500}}{\binom{1004}{504}\left(0.7^{504}\right)0.3^{500}+\binom{1004}{500}\left(0.3^{504}\right)0.7^{500}}$$

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

$$\frac{\cancel{\binom{1004}{504}}\left(0.7^{504}\right)0.3^{500}}{\cancel{\binom{1004}{504}}\left(0.7^{504}\right)0.3^{500}+\cancel{\binom{1004}{500}}\left(0.3^{504}\right)0.7^{500}}$$

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

$$\frac{\cancel{\binom{1004}{504}}(0.7^{504})\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}(0.7^{504})\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}(0.3^{\cancel{504}4})0.7^{500}}$$

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

$$\frac{\cancel{\binom{1004}{504}}\left(0.7^{\cancel{50}4}\right)\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}\left(0.7^{\cancel{50}4}\right)\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}\left(0.3^{\cancel{50}4}\right)\cancel{0.7^{500}}}$$

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

$$\frac{(0.7^4)}{(0.7^4)+(0.3^4)} = \frac{\cancel{\binom{1004}{504}}(0.7^{\cancel{50}4})\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}(0.7^{\cancel{50}4})\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}(0.3^{\cancel{50}4})\cancel{0.7^{500}}}$$

How many jobs to distribute?

room 1

Flip a 70% / 30% coin 4 times
get 4 heads and 0 tails.

room 2

Flip a 70% / 30% coin 1004 times
get 504 heads and 500 tails.

$$\frac{(0.7^4)}{(0.7^4)+(0.3^4)} = \frac{\cancel{\binom{1004}{504}}(0.7^{\cancel{504}4})\cancel{0.3^{500}}}{\cancel{\binom{1004}{504}}(0.7^{\cancel{504}4})\cancel{0.3^{500}}+\cancel{\binom{1004}{500}}(0.3^{\cancel{504}4})\cancel{0.7^{500}}}$$

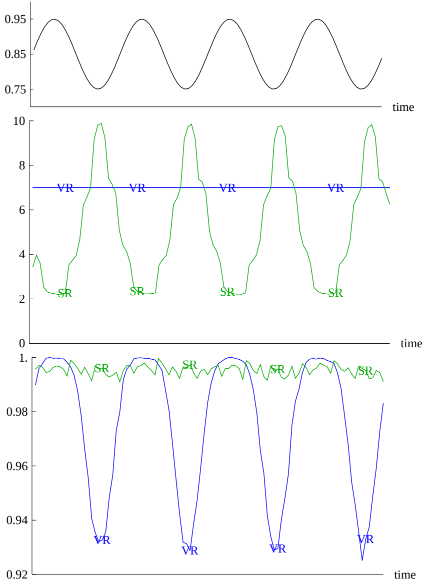
Bayes theorem implies that given an a-b split of answers, only the difference affects the reliability.

Inject redundancy only when it is needed

node reliability:

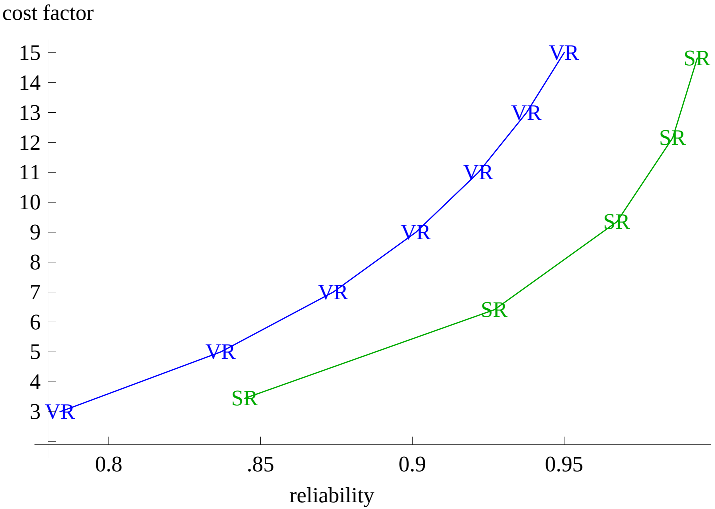
cost factor:

system reliability:



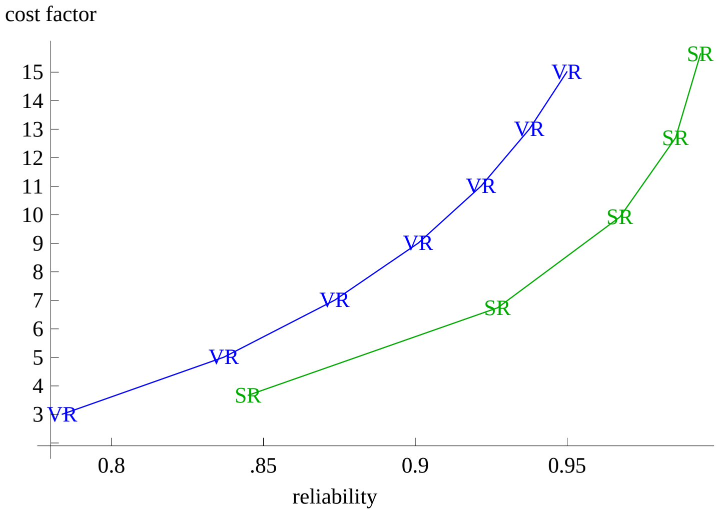
Smart always outperforms voting redundancy

Theoretical results



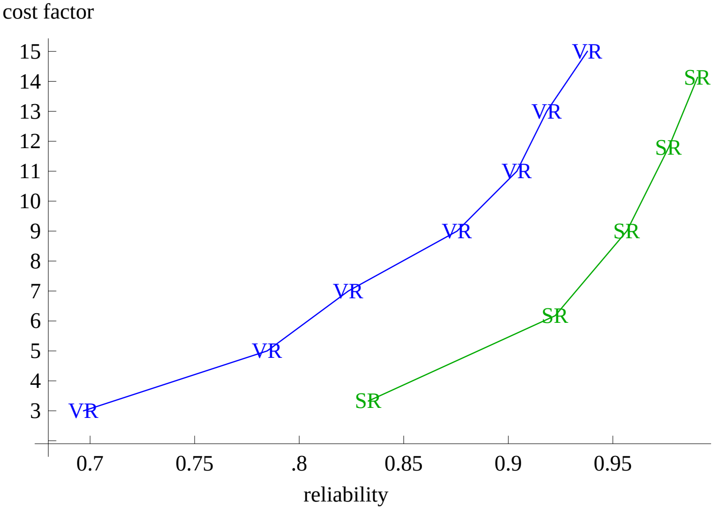
Simulation analysis confirms theoretical predictions

Simulated 1,000,000 task executions on 10,000 nodes using the XDEVS simulator [Edwards 2010]

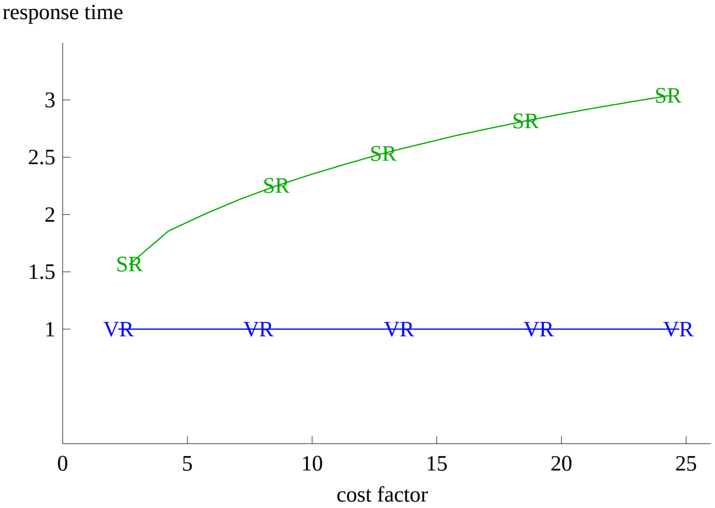


Empirical analysis confirms theoretical predictions

Deployed a SAT solver using BOINC [Anderson 2004]
on PlanetLab [Peterson et al. 2003]



Response time cost



Iterating increases individual task response time

Related work

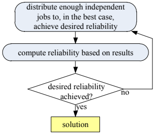
other redundancy techniques

- self-configuring optimistic programming [Bondavalli et al. 2002]
- credibility-based fault tolerance [Sarmenta 2002]
- checkpointing [Priya et al. 2007]
- crowdsourcing [Barowy et al. 2012]
- Byzantine faults in service-based computing (ZZ [Wood et al. 2011])

complementary

- primary backup [Budhiraja et al. 1993]
- active replication [Schneider 1990]
- developer-defined fault detection [Hwang and Kesselman 2003]

Contributions and Future Projects

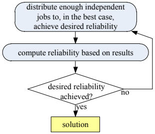


smart redundancy: using resources optimally to boost reliability

What's next?

- Channels with more bandwidth than 1 bit
- Using history to improve resource use (non-Byzantine)
- Crowdsourcing

Contributions and Future Projects



smart redundancy: using resources optimally to boost reliability

What's next?

- Channels with more bandwidth than 1 bit
- Using history to improve resource use (non-Byzantine)
- Crowdsourcing

For more, see "Smart redundancy for distributed computation" by Y. Brun et al. In the 31st International Conference on Distributed Computing Systems (ICDCS), 665-676, 2011. `http://dx.doi.org/10.1109/ICDCS.2011.25`

David P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID04)*, pages 4–10, Pittsburgh, PA, USA, 2004. ISBN 0-7695-2256-4. doi: http://dx.doi.org/10.1109/GRID.2004.14.

Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, February 2006. doi: 10.1145/1127345.1127346.

David Baker. Foldit. `http://fold.it`, 2009.

Daniel W. Barowy, Charlie Curtsinger, Emery D. Berger, and Andrew McGregor. AutoMan: A platform for integrating human-based and digital computation. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA12)*, pages 639–654, Tucson, AZ, USA, 2012. ISBN 978-1-4503-1561-6. doi: 10.1145/2384616.2384663.

BOINC. The Berkeley open infrastructure for network computing. `http://boinc.berkeley.edu`, 2009.

Andrea Bondavalli, Felicita Di Giandomenico, and Jie Xu. A cost-effective and flexible scheme for software fault tolerance. *Journal of Computer Systems Science and Engineering*, 8(4):234–244, 1993.

Andrea Bondavalli, Silvano Chiaradonna, Felicita Di Giandomenico, and Jie Xu. An adaptive approach to achieving hardware and software fault tolerance in a distributed computing environment. *Journal of Systems Architecture*, 47(9):763–781, 2002. ISSN 1383-7621. doi: 10.1016/S1383-7621(01)00029-7.

Yuriy Brun. Arithmetic computation in the tile assembly model: Addition and multiplication. *Theoretical Computer Science*, 378 (1):17–31, June 2007. ISSN 0304-3975. doi: 10.1016/j.tcs.2006.10.025.

Yuriy Brun. Efficient 3-SAT algorithms in the tile assembly model. *Natural Computing*, 11(2):209–229, 2012. doi: 10.1007/s11047-011-9299-0.

Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. The primary-backup approach. In *Distributed Systems*, pages 199–216. ACM Press/Addison-Wesley Publishing Co., 2 edition, 1993. ISBN 0-201-62427-3.

Arjav J. Chakravarti and Gerald Baumgartner. The organic grid: Self-organizing computation on a peer-to-peer network. In *Proceedings of the 1st International Conference on Autonomic Computing (ICAC04)*, pages 96–103, New York, NY, USA, 2004. ISBN 0-7695-2114-2.

Andrew M. Childs. Secure assisted quantum computation. *Quantum Information and Computation*, 5(456):456–466, 2005.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI04)*, San Francisco, CA, USA, December 2004.

George T. Edwards. *Automated Synthesis of Domain-Specific Model Interpreters*. PhD thesis, University of Southern California, Los Angeles, CA, USA, 2010.

Pascal Felber and Andre Schiper. Optimistic active replication. In *Proceedings of the 21st IEEE International Conference on Distributed Computing Systems (ICDCS01)*, pages 333–341, Phoenix, AZ, USA, 2001. IEEE Computer Society.

Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222, 2001. ISSN 1094-3420. doi: 10.1177/109434200101500302.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing (STOC09)*, pages 169–178, Bethesda, MD, USA, 2009. ISBN 978-1-60558-506-2. doi: 10.1145/1536414.1536440.

High Performance Computing and Communications. High performance computing and communications. `http://www.usc.edu/hpcc`.

Soonwook Hwang and Carl Kesselman. A flexible framework for fault tolerance in the grid. *Journal of Grid Computing*, 1(3): 251–272, September 2003. ISSN 1570-7873. doi: 10.1023/B:GRID.0000035187.54694.75.

Israel Koren and C. Mani Krishna. *Fault-Tolerant Systems*. Elsvier, Inc., 2007.

Eric Korpela, Dan Werthimer, David Anderson, Jeff Cobb, and Matt Lebofsky. SETI@home — massively distributed computing for SETI. *IEEE MultiMedia*, 3(1):78–83, 1996. ISSN 1070-986X.

Stefan M. Larson, Christopher D. Snow, Michael R. Shirts, and Vijay S. Pande. *Folding@Home and Genome@Home: Using Distributed Computing to Tackle Previously Intractable Problems in Computational Biology*. Horizon Press, 2002.

Sam Malek, Marija Mikic-Rakic, and Nenad Medvidovic. A style-aware architectural middleware for resource-constrained, distributed systems. *IEEE Transactions on Software Engineering*, 31(3):256–272, 2005. ISSN 0098-5589. doi: 10.1109/TSE.2005.29.

Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the Internet. *ACM SIGCOMM Computer Communication Review*, 33(1):59–64, 2003. ISSN 0146-4833. doi: http://doi.acm.org/10.1145/774763.774772.

S. Baghavathi Priya, M. Prakash, and K. K. Dhawan. Fault tolerance-genetic algorithm for grid task scheduling using check point. In *Proceedings of the 6th International Conference on Grid and Cooperative Computing (GCC07)*, pages 676–680, 2007. ISBN 0-7695-2871-6. doi: 10.1109/GCC.2007.67.

Rosetta@home. Rosetta@home. `http://boinc.bakerlab.org/rosetta`, 2007.

Luis F. G. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems. *Future Generation Computer Systems*, 18(4):561–572, 2002. ISSN 0167-739X. doi: 10.1016/S0167-739X(01)00077-2.

Todd W. Schiller and Michael D. Ernst. Rethinking the economics of software engineering. In *Workshop on the Future of Software Engineering Research*, Santa Fe, NM, USA, November 2010. doi: 10.1145/1882362.1882429.

Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990. ISSN 0360-0300. doi: 10.1145/98163.98167.

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI04)*, pages 319–326, Vienna, Austria, 2004. doi: 10.1145/985692.985733.

Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008. doi: 10.1126/science.1160379.

Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(5):847–859, May 2011. doi: 10.1109/TPDS.2010.183.

Erik Winfree. Simulations of computing by self-assembly of DNA. Technical Report CS-TR:1998:22, California Institute of Technology, Pasadena, CA, USA, 1998.

Timothy Wood, Rahul Singh, Arun Venkataramani, Prashant Shenoy, and Emmanuel Cecchet. ZZ and the art of practical BFT execution. In *Proceedings of the 6th European Conference on Computer Systems (EuroSys11)*, pages 123–138, Salzburg, Austria, 2011. ISBN 978-1-4503-0634-8. doi: 10.1145/1966445.1966457.

Zhenyu Yang, Shucheng Yu, Wenjing Lou, and Cong Liu. P^2: Privacy-preserving communication and precise reward architecture for V2G networks in smart grid. *IEEE Transactions on Smart Grid*, 2011.

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science (FOCS86)*, pages 162–167, Toronto, ON, Canada, October 1986.

Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th Conference on Information Communications (INFOCOM10)*, pages 534–542, San Diego, CA, USA, 2010.