2-party Secure Computation

Malicious Adversaries

Bar-Ilan Winter School, Feb 2015
abhi shelat
Brief Survey
...and nothing else
### The age of optimism

<table>
<thead>
<tr>
<th>Decade</th>
<th>Invention</th>
<th>80s</th>
<th>90s</th>
<th>00s</th>
<th>10s</th>
<th>20s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PKE</td>
<td>PKE</td>
<td>PKE</td>
<td></td>
<td></td>
<td>SFE</td>
</tr>
<tr>
<td>Invented</td>
<td>SFE</td>
<td></td>
<td></td>
<td>Feasible</td>
<td>Practical</td>
<td>Ubiquit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SFE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MNPS04
MNPS08
KS06, K08

Fairplay
Honest but curious

4k gates,
600 gates/sec
<table>
<thead>
<tr>
<th>MNPS04, MNPS08</th>
<th><strong>Fairplay</strong></th>
<th>4k gates, 600 gates/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>KS06, K08</td>
<td>Honest but curious</td>
<td></td>
</tr>
<tr>
<td>LP04, LP07, LPS08</td>
<td><strong>Cut-and-choose</strong></td>
<td>1k gates, 4 gates/sec</td>
</tr>
<tr>
<td></td>
<td>Malicious adv</td>
<td></td>
</tr>
</tbody>
</table>
PSSW09

AES circuit
Malicious adv

40k gates,
35 gates/sec
(2^{-40} security)

Alice
x

f

Bob
y

AES_x(y)
<table>
<thead>
<tr>
<th>MNPS04</th>
<th>Fairplay</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Honest but curious</td>
</tr>
<tr>
<td></td>
<td>4k gates, 600 gates/sec</td>
</tr>
<tr>
<td>MNPS08</td>
<td>Cut-and-choose</td>
</tr>
<tr>
<td></td>
<td>Malicious adv</td>
</tr>
<tr>
<td></td>
<td>1k gates, 4 gates/sec</td>
</tr>
<tr>
<td>KS06, K08</td>
<td>Hybrid, C&amp;C+ZK</td>
</tr>
<tr>
<td></td>
<td>Malicious adv</td>
</tr>
<tr>
<td>LP04, LP07, LPS08</td>
<td>Yao + ZK</td>
</tr>
<tr>
<td></td>
<td>Malicious adv</td>
</tr>
<tr>
<td>PSSW09</td>
<td>Lego+</td>
</tr>
<tr>
<td></td>
<td>Malicious adv</td>
</tr>
<tr>
<td></td>
<td>Protocol</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>MNPS04, MNPS08</td>
<td><strong>Fairplay</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>KS06, K08</td>
<td></td>
</tr>
<tr>
<td>LP04, LP07, LPS08</td>
<td><strong>Cut-and-choose</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PSSW09</td>
<td></td>
</tr>
<tr>
<td>LP10</td>
<td><strong>Hybrid, C&amp;C+ZK</strong></td>
</tr>
<tr>
<td>JS07 JKS08</td>
<td><strong>Yao + ZK</strong></td>
</tr>
<tr>
<td>NO09</td>
<td><strong>Lego+</strong></td>
</tr>
<tr>
<td>IPS08,09, LOP11</td>
<td><strong>Better BB Cut-and-choose</strong></td>
</tr>
<tr>
<td>HL08, HL08b</td>
<td><strong>Tamper proof model</strong></td>
</tr>
<tr>
<td>Scheme</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>PSSW09</td>
<td>AES circuit</td>
</tr>
<tr>
<td>SS11</td>
<td>Hybrid CC+ZK</td>
</tr>
<tr>
<td>NNOB11</td>
<td>GMW + OT Ext</td>
</tr>
<tr>
<td>DPSZ11</td>
<td>GMW + Beaver</td>
</tr>
</tbody>
</table>

Amortized

- 10ms/gate ~ 100 g/s
- 5000x (2^{-40} security) 3s/block
- 10 gates/sec
- 20k gates/sec
- 130 gates/sec
- 35 gates/sec

5000x
Bottleneck became the Compiler
2011

HEKM11     Pipeline + Circuit Lib   40k gates
            Honest but curious      12k gates/sec

Bottleneck became the Compiler

JKS08     200x200 edit distance  660s
HEKM11 Pipeline + Circuit Lib Honest but curious 40k gates 12k gates/sec

Bottleneck became the Compiler
JKS08 200x200 edit distance 660s

HEKM11 1.2B nonxor gates 96k g/s 2k x 10k edit distance
2012

KSS12

1.2B nonxor gates
2k x 10k edit distance
6B gates  4K x 4K edit distance
260m gate   RSA-256
330m gate   2k x 2s Edit

96k g/s
86k gates/sec
125k gates/sec
123k gates/sec
MNPS04  
MNPS08  
KS06, K08  
LP04, LP07, LPS08

Fairplay  
Honest but curious

4k gates,  
600 gates/sec

Cut-and-choose  
Malicious adv

1k gates,  
4 gates/sec

PSSW09

AES circuit  
Malicious adv

40k gates,  
35 gates/sec

LP11

Hybrid, C&C+ZK  
Malicious adv

Hybrid C&C+ZK  
Malicious adv

40k gates,  
130 gates/sec

SS11

KSS12

Hybrid CC+ZK, Parallel  
Malicious adv

6B gates,  
130k gates/sec

SS13

CC, Parallel  
Malicious adv

B gates,  
1M gates/sec
More Garbled Circuits work

K08  
Output Auth

KS08  
Free XOR-trick

CKKZ12  
Using circular 2-corr RHF

HEKM11  
Pipeline + Circuit Lib

HS13  
Less Memory, Parallel

FN12  
GPU system

HMSG13  
40k- 1.2B gates

Honest but curious

12k-96k gates/sec

35M gates/sec
<table>
<thead>
<tr>
<th>Reference</th>
<th>Scheme</th>
<th>Malicious adv</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS07 JKS08</td>
<td>Yao + ZK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NO09</td>
<td>Lego+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPS08,09, LOP11</td>
<td>Better BB Cut-and-choose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NNOB11</td>
<td>GMW + OT Ext</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPSZ11</td>
<td>GMW + Beaver + SHE</td>
<td>100k ops</td>
<td>10ms/&quot;op&quot; ~ 100 ops/s</td>
</tr>
<tr>
<td>DKLPS12</td>
<td>GMW + OT Ext</td>
<td>500 ops/s</td>
<td></td>
</tr>
<tr>
<td>SZ13</td>
<td>GMW + OT Ext</td>
<td>Fast</td>
<td>??</td>
</tr>
</tbody>
</table>
Advanced Techniques

Cut & choose

Lindell13
Huang-Evans-Katz13

Amortization: C&C + LEGO

Huang-Katz-Kolesnikov-Kumaresan-Malozemoff14
Lindell-Riva14

Garbling

Zahur-Evans-Rosulek14
Malkin-Pastro-shelat15

Algorithmic

Venkatasubramanian-shelat15
ORAM Secure Computation

Gordon-Katz-Kolesnikov-Krell-Malkin12

Keller-Scholl14
- 4 accesses/second to oblivious array of size one million
- Dijkstra’s algorithm:
  - $2^{11}$ vertices and $2^{12}$ edges in 10 hours
  - $2^{18}$ vertices and $2^{19}$ edges in 14 months
  (estimated from running a fully functional program)

Wang-Huang-Chan-shelat-Shi14

SCORAM: 4m gates/ORAM op

Wang-Chan-Shi14

CORAM: 500k gates/ORAM op

BenSasson-Chiesa-Tromer-Virza14

TinyRAM
Question: which secure computation techniques are preferable?
overhead

2-party Secure computation

Plain → HBC → Malicious
overhead

2-party Secure computation

Parallelizability is KEY
Basic Protocols
Garbled circuits

Y82

Garbled gates + Composition + Key Mgmt

Oblivious Transfer

0 1 c
Honest-but-curious

\[ f(x,y) \]

OT 1st msg

OT 2nd msg

Garbled circuit, keys for x

2 round!
1. Incrementally construct maliciously-secure protocol
Definition

\[ \forall A \exists S \forall (x_1, x_2), z \]

\[ \text{IDEAL}_{f, S(z), I}(x_1, x_2, k) \approx_c \text{REAL}_{f, A(z), I}(x_1, x_2, k) \]
IDEAL

S, l

TTP

f(x, y)

out: f(x, y)

TTP

f(x, y)

out: f(x, y)
IDEAL

\[ f(x, y') \]

\[ f(x, y') \]

\[ \text{out: } f(x, y) \]

\[ \text{out: } ? \]
\[ \text{REAL} \quad A, l \]

\[ x \rightarrow f(x, y) \rightarrow y \]

\[ \text{out:} f(x, y) \quad \text{out:} ? \]
Definition

\[ \forall A \exists S \forall (x_1, x_2), z \quad \text{IDEAL}_{f,S(z),I}(x_1, x_2, k) \approx_c \text{REAL}_{f,A(z),I}(x_1, x_2, k) \]
1. Incrementally construct maliciously-secure protocol

2. Optimize
What can go wrong?

OT 1st msg

OT 2nd msg

Garbled circuit

sending a bad circuit
Prove circuit is good

GMW, Jarecki-Shmatikov07

\[
\bigwedge_{g \in G} \text{CorrectGarble}_g \land \bigwedge_{w \in W} \text{GoodKeys}_w \land \bigwedge_{w \in W_S} \text{CorrectInput}_w \\
\land \bigwedge_{w \in W_R} \text{ZKS}_w \land \bigwedge_{w \in W_O} \text{CorrectOutput}_w
\]

where

\[
\text{GoodKeys}_w = \text{ZKNotEq}(C_0^w, C_1^w)
\]

\[
\text{CorrectInput}_w = (\text{ZKDL}(g, C_0^w / x_{b_w}) \land \text{ZKDL}(g, C_b)) \lor
\text{ZKDL}(g, C_1^w / x_{b_w}) \land \text{ZKDL}(g, C_b / \alpha)), \text{where } C_b \text{ is the sCS commitment inside } \text{Com}_{cids_i} \text{ if } w \text{ is the } i^{th} \text{ input wire of } S
\]

\[
\text{CorrectOutput}_w = \text{ZKPlainEq2}(E_0^w, C_0^w, 0) \land \text{ZKPlainEq2}(E_1^w, C_1^w, 1)
\]

\[
\text{CorrectGarble}_g = \text{CorrectShuffle}(0, 0) \lor \text{CorrectShuffle}(0, 1) \lor \\
\text{CorrectShuffle}(1, 0) \lor \text{CorrectShuffle}(1, 1)
\]

\[
\text{CorrectShuffle}(\alpha, \beta) = \text{CorrectCipher}(0, 0, \alpha, \beta) \land \text{CorrectCipher}(0, 1, \alpha, \beta) \land \\
\text{CorrectCipher}(1, 0, \alpha, \beta) \land \text{CorrectCipher}(1, 1, \alpha, \beta)
\]

\[
\text{CorrectCipher}(\sigma_A, \sigma_B, \alpha, \beta) = \text{ZKPlainEq}(F^{(1)}_{\alpha\beta}, C^{A}_{\alpha \oplus \sigma_A}; D_{\alpha \beta}) \land \\
\text{ZKPlainEq}(F^{(2)}_{\alpha\beta}, C^{B}_{\beta \oplus \sigma_B}; (C^{g}_{\sigma_B(\alpha \oplus \sigma_A, \beta \oplus \sigma_B)} / D_{\alpha \beta}))
\]
CorrectGarble_g = CorrectShuffle(0, 0) ∨ CorrectShuffle(0, 1) ∨ CorrectShuffle(1, 0) ∨ CorrectShuffle(1, 1)

CorrectShuffle(α, β) = CorrectCipher(0, 0, α, β) ∧ CorrectCipher(0, 1, α, β) ∧ CorrectCipher(1, 0, α, β) ∧ CorrectCipher(1, 1, α, β)

CorrectCipher(σ_A, σ_B, α, β) = ZKPlainEq(F^{(1)}_{αβ}, C_{α⊕σ_A}^A, D_{αβ}) ∧ 
ZKPlainEq(F^{(2)}_{αβ}, C_{β⊕σ_B}^B, (C_{g(α⊕σ_A, β⊕σ_B)}^{C}/D_{αβ}))

32-clause Sigma-protocol PER gate
Given \( \text{com}(K^0_x), \text{com}(K^1_x), \text{com}(K^0_y), \text{com}(K^1_y), \text{com}(K^0_w), \) and \( \text{com}(K^1_w) \), \( P_2 \) needs \( P_1 \) to prove that the AND gate \((\delta, T_4, T_5, \sigma, T_\sigma)\) is correctly computed. More specifically,

(a) \( P_1 \) sends \( \text{com}(\delta; r) \) to \( P_2 \), and \( P_1 \) proves that \( \text{com}(\delta; r) = q^\delta b^r \).

(b) For every \((b_0, b_1) \in \{0, 1\}^2\), let \( i = 2 \cdot b_0 + b_1 \), \( P_1 \) sends \( \text{com}(T_i) \) to \( P_2 \) and proves that
\[
\left( \text{com}(K^b_0) \text{com}(K^b_1) \text{com}(\delta) = \text{com}(K^b_0 + K^b_1 + \delta) \right) \land \\
\left( \text{com}(T_i) = \text{com}(K^b_0 + K^b_1 + \delta) \right).
\]
Moreover, \( P_1 \) proves that \( T_i \in \mathbb{Z}_N^* \) for \( i = 0, 1, 2, 3 \).

(c) Let \( \text{Mask}(b_0, b_1) \) denote the case that \( (K^0_x)_N = b_0 \) and \( (K^0_y)_N = b_1 \). \( P_1 \) proves to \( P_2 \) that
\[
\text{mask}(0, 0) \lor \text{mask}(0, 1) \lor \text{mask}(1, 0) \lor \text{mask}(1, 1).
\]

In particular, for case \( \text{mask}(b_0, b_1) \), let
\[
\begin{align*}
a_0 &= 2 \cdot b_0 + b_1 \\
a_1 &= 2 \cdot b_0 + (1 - b_1)
\end{align*}
\text{and}
\begin{align*}
a_2 &= 2 \cdot (1 - b_0) + b_1 \\
a_3 &= 2 \cdot (1 - b_0) + (1 - b_1).
\end{align*}
\]

It is defined that
\[
\text{mask}(b_0, b_1) = (P(a_0) = T_0) \land (P(a_1) = T_1) \land (P(a_2) = T_2) \land (Q(a_3) = T_3),
\]
where \( P(x) \) is the Lagrange polynomial coincides at points \((-1, K^0_w)\), \((4, T_4)\), \((5, T_5)\), and \((\sigma, T_\sigma)\); and \( Q(x) \) is the Lagrange polynomial coincides at points \((-1, K^1_w)\), \((4, T_4)\), \((5, T_5)\), and \((\sigma, T_\sigma)\).
Open problem to optimize so as to outperform C&C
Cut & Choose
First Idea: Cut & Choose

OT 1st msg

OT 2nd msg

Send $k$ fresh garbled circuits
First Idea: Cut & Choose

OT 1st msg

OT 2nd msg

Send $k$ fresh garbled circuits

“open” challenge set of $t$ circuits

random coins for challenge
Garbler sends $k$ circuits to Evaluator. Evaluator selects $t$ to test. Evaluator verifies that all $t$ circuits are valid.

G asks E for random coins used to garble.
What does this cut&choose test accomplish?
Balls & Bins

k circuits in total

Evaluator picks $c$ circuits to corrupt.
Garbler picks $t$ circuits to test.

$\binom{k - c}{t}$

$\binom{k}{t}$
Given that evaluator checks \( t \),
Pr that garbler succeeds in passing test:

\[
\frac{\binom{k-c}{t}}{\binom{k}{t}}
\]

setting \( t=k/2 \)
Given that evaluator checks $t$,
Pr that garbler succeeds in passing test:

\[
\frac{\binom{k-c}{t}}{\binom{k}{t}} = \frac{(k/2)(k/2 - 1) \cdots (k/2 - c)}{k(k - 1)(k - 2) \cdots (k - c)}
\]

setting $t = k/2$
Given that evaluator checks $t$, 
Pr that garbler succeeds in passing test:

$$\frac{\binom{k-c}{t}}{\binom{k}{t}} = \frac{(k/2)(k/2-1) \cdots (k/2-c)}{k(k-1)(k-2) \cdots (k-c)} < 2^{-c}$$

setting $t=k/2$
NEGL probability that test passes if $O(k)$ circuits are bad
\[ 2^{-c} > \frac{{k-c \choose t}}{{k \choose t}} = \frac{(k/2)(k/2 - 1) \cdots (k/2 - c)}{k(k - 1)(k - 2) \cdots (k - c)} \geq \left(\frac{1}{2} - \frac{c}{k}\right)^c \]

setting \( t = k/2 \)

\[ \frac{k/2 - c}{k - c} \geq \frac{k/2 - c}{k} \geq \left(\frac{1}{2} - \frac{c}{k}\right) \]

Noticeable probability that \( O(1) \) circuits are corrupted
What do we do with the remaining circuits?
First idea:
Abort if outputs are not all the same.
First idea:
Abort if outputs are not all the same.

If \( y_1 = 0 \), output \( f(x,y) \)
else output \( f(x,y)+1 \)
∀A ∃S ∀(x_1, x_2), z

\text{IDEAL}_{f,S(z),I}(x_1, x_2, k) \approx_c \text{REAL}_{f,A(z),I}(x_1, x_2, k)

If \ y_1=0, \ output \ f(x,y)
else \ output \ f(x,y)+1
Comment

In practice, all circuits must have same # of gates & same wiring.

Cheating restricted to changing gates.

Hard to analyze.
Second idea:

Eval all remaining circuits, take *majority* output.
Third idea:

Eval all remaining circuits, exploit cheating later.

state-of-the-art [L13]
Send $k$ fresh garbled circuits

challenge set of $t$ circuits

random coins for challenge

majority of Eval
Problem: Garblers’ inputs

In basic protocol, garblers’ input wires sent in this step.
OT

Send $k$ fresh garbled circuits

challenge set of $t$ circuits

challenge response

majority of Eval

Can’t send garblers’ inputs in this step anymore!
Can’t send garblers’ inputs in this step anymore!

Send here instead.

G keys

Send k fresh garbled circuits

challenge set of t circuits

challenge response

majority of Eval
Problem: Input Consistency

OT

Send $k$ fresh garbled circuits

challenge set of $t$ circuits

challenge response $K^1_{in}, K^2_{in}, \ldots, K^l_{in}$

majority of Eval

$l = k - t$ circuits
Needs all input keys.
Problem: Input Consistency

OT

Send $k$ fresh garbled circuits

challenge set of $t$ circuits

challenge response $K^1_{in}, K^2_{in}, \ldots, K^t_{in}$

majority of Eval

What if keys do not correspond to same input?

$l = k - t$ circuits

Needs all input keys.
Input Consistency Attack

\[ f(x,y) = \langle x,y \rangle \]

(inner product)

[Gen] \[x\] \[y\] [Eval]

[Mohassel-Franklin06, Kiraz-Schoenmakers06, Lindell-Pinkas07]
Input Consistency Attack

[y_1] [y_2] [y_3] [y_4]

<x,y> (inner product)

x 0001 y

x 0010 y

x 0100 y

x 1000 y

[Mohassel-Franklin06, Kiraz-Schoenmakers06, Lindell-Pinkas07]
Input Consistency Attack

Majority($y_1, y_2, y_3, y_4$)

[Mohassel-Franklin06, Kiraz-Schoenmakers06, Lindell-Pinkas07]
∀A ∃S ∀(x_1, x_2), z

IDEAL_{f,S(z),I}(x_1, x_2, k) \cong_c REAL_{f,A(z),I}(x_1, x_2, k)
Input Consistency Attack

\[ \langle x, y \rangle \]

(inner product)

\[ \begin{align*}
  y_1 & : \quad 0001 \quad y \\
  y_2 & : \quad 0010 \quad y \\
  y_3 & : \quad 0100 \quad y \\
  y_4 & : \quad 1000 \quad y
\end{align*} \]

Majority\((y_1, y_2, y_3, y_4)\)  Bad!

[Mohassel-Franklin06, Kiraz-Schoenmakers06, Lindell-Pinkas07]
How to handle inconsistent inputs?

$K^1_{in}, K^2_{in},...,K^l_{in}$

Prove consistency
OT

Send $k$ fresh garbled circuits

challenge set of $t$ circuits

challenge resp $K^1_{in}, \ldots, K^t_{in}$

majority of Eval
<table>
<thead>
<tr>
<th></th>
<th>Input Consistency</th>
<th>2-Outputs</th>
<th>OT +</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Theta(k^2 n)$</td>
<td>$\Theta(k^2 n)$</td>
<td>OWF</td>
</tr>
<tr>
<td><strong>LP07</strong></td>
<td>$\Theta(k^2 n)$</td>
<td>$\Theta(k n)$</td>
<td>DLOG</td>
</tr>
<tr>
<td>“blackbox”</td>
<td>$\Theta(k n)$</td>
<td>$\Theta(k n)$</td>
<td>DLOG</td>
</tr>
<tr>
<td><strong>Kiraz08</strong></td>
<td>$\Theta(k n)$</td>
<td>$\Theta(k n)$</td>
<td>DLOG</td>
</tr>
<tr>
<td><strong>LP11</strong></td>
<td>$\Theta(k n)$</td>
<td>$\Theta(k n)$</td>
<td>DLOG</td>
</tr>
<tr>
<td><strong>SS11</strong></td>
<td>$\Theta(k n)$</td>
<td>$\Theta(k n)$</td>
<td>DLOG</td>
</tr>
<tr>
<td><strong>KSS12</strong></td>
<td>$\Theta(k n)$</td>
<td>$\Theta(k n)$</td>
<td>DLOG</td>
</tr>
<tr>
<td><strong>SS13</strong></td>
<td>$\Theta(k n)$</td>
<td>$\Theta(k n)$</td>
<td>OWF</td>
</tr>
</tbody>
</table>
Problem: Malicious OT

Use Malic-secure OT here. Is that enough?

Send $k$ fresh garbled circuits

challenge set of $t$ circuits

challenge resp $K_1^{\text{in}}, \ldots, K_l^{\text{in}}$

majority of Eval
Input OT
“Key management”

keys for $y_i=0$
$$\{w_{i,0}(j)\}_{j \in [k]}$$
keys for $y_i=0$

0 1 c
Oblivious Transfer

input \{0,1\}
Oblivious Transfer

What are the possible outcomes?

Selective Failure attack
Oblivious Transfer

bogus keys

keys for $y_i = 0$

$\{w_{i,0}^{(j)}\}_{j \in [k]}

$\{0\}_{j \in [k]}

What are the possible outcomes?

Input $y_i = 0$\hspace{1cm}OK

Input $y_i = 1$\hspace{1cm}FAIL: Cannot Eval

Selective Failure attack
Selective Failure Solutions

Encode inputs

Prove consistency
Send k fresh garbled circuits

challenge set of t circuits

challenge resp $K^1_{in}, \ldots, K^l_{in}$

majority of Eval

OT $\pi'_1$

$\pi_1 \pi'_2$
Committing OT

Com(Alice’s inputs)

Coin Flipping

Open Circuits, send Eval
Key problems for Malicious Security

Circuit Consistency

Input Consistency

Selective Failure

Output Authentication
(2-output case)
Circuit

Consistency
Given that evaluator checks $t$, 
Pr that garbler succeeds in passing test:

\[
\frac{\binom{k-c}{t}}{\binom{k}{t}}
\]
k=10. Suppose evaluator checks 1. Garbler can choose how many to corrupt.
$k=10$. Suppose evaluator checks 1.

Garbler can choose how many to corrupt.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>2/5</td>
<td>3/10</td>
<td>1/5</td>
<td>1/10</td>
<td>0</td>
</tr>
</tbody>
</table>
$k=10$. Suppose evaluator checks 2.

Garbler can choose how many to corrupt.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
k=10. Suppose evaluator checks 2.

Garbler can choose how many to corrupt.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>2/9</td>
<td>2/15</td>
<td>1/15</td>
<td>1/45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># eval checks</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>2/5</td>
<td>3/10</td>
<td>1/5</td>
<td>1/10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>2/9</td>
<td>2/15</td>
<td>1/15</td>
<td>1/45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/6</td>
<td>1/12</td>
<td>1/30</td>
<td>1/120</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1/6</td>
<td>1/14</td>
<td>1/42</td>
<td>1/210</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1/12</td>
<td>1/42</td>
<td>1/252</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2/15</td>
<td>1/30</td>
<td>1/210</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1/15</td>
<td>1/120</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1/5</td>
<td>1/45</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1/10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pr garbler succeeds in corrupting a majority of evaluated circuits
If eval checks $t$ circuits, garbler should corrupt

$$\left\lfloor \frac{(k - t) + 1}{2} \right\rfloor$$

majority of evaluated should be corrupt, no more
Evaluator should thus check $t^*$ circuits to minimize

$$\min_t \left[ k - \left\lfloor \frac{(k-t)+1}{2} \right\rfloor \right] \frac{t^k}{\binom{k}{t}}$$
s copies of the circuit can yield

\[ 2 - 0.32s \]

if \( t^* \sim 3/5s \)
Optimal for single choice of t.

But Eval can randomize choice of t.
Value of game

If Garbler wins, payoffs are (1,-1)
If Garbler looses, payoffs are (-1,1)

Both parties can run probabilistic strategies.

Game is zero-sum.

\[
\text{min payoff that Evaluator can force} = \text{max payoff that Garbler can achieve}
\]
We want to solve

\[
\min_{e_1, \ldots, e_k} \max_{x_1, \ldots, x_k} \prod e_t x_c \left( \frac{\binom{k-c}{t}}{\binom{k}{t}} \right)
\]

\(e_i\) : Pr that evaluator checks \(i\)

\(x_j\) : Pr that garbler corrupts \(j\)
Linear Program

Variables $x_i$: Pr that garbler corrupts $i$ circuits

$(x_1, x_2, \ldots, x_n)$

Table for Eval checking 1 circuit:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>2/5</td>
<td>3/10</td>
<td>1/5</td>
<td>1/10</td>
<td>0</td>
</tr>
</tbody>
</table>

$v_1 = (x_1, x_2, \ldots, x_n) \cdot (0, 0, 0, 0, 0, \frac{1}{2}, \frac{2}{5}, \frac{3}{10}, \frac{1}{5}, \frac{1}{10}, 0)$

Expected payoff if Eval check 1 circuit.
\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 1/2 & 2/5 & 3/10 & 1/5 & 1/10 \\
0 & 0 & 0 & 1/3 & 2/9 & 2/15 & 1/15 & 1/45 & 0 \\
0 & 0 & 0 & 1/6 & 1/12 & 1/30 & 1/120 & 0 & 0 \\
0 & 0 & 1/6 & 1/14 & 1/42 & 1/210 & 0 & 0 & 0 \\
0 & 0 & 1/12 & 1/42 & 1/252 & 0 & 0 & 0 & 0 \\
0 & 2/15 & 1/30 & 1/210 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/15 & 1/120 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/5 & 1/45 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
x_7 \\
x_8 \\
x_9 \\
\end{bmatrix}
\]

Evaluator chooses min row

To express as LP, add variable v.
maximize \( v \)

subject to

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{2}{5} & \frac{3}{10} & \frac{1}{5} & \frac{1}{10} & -1 \\
0 & 0 & 0 & 0 & \frac{1}{3} & \frac{2}{9} & \frac{2}{15} & \frac{1}{15} & \frac{1}{45} & 0 & -1 \\
0 & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{12} & \frac{1}{30} & \frac{1}{120} & 0 & 0 & -1 \\
0 & 0 & \frac{1}{6} & \frac{1}{14} & \frac{1}{42} & \frac{1}{210} & 0 & 0 & 0 & -1 \\
0 & 0 & \frac{1}{12} & \frac{1}{42} & \frac{1}{252} & 0 & 0 & 0 & 0 & -1 \\
0 & \frac{2}{15} & \frac{1}{30} & \frac{1}{210} & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & \frac{1}{15} & \frac{1}{120} & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\frac{1}{5} & \frac{1}{45} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\frac{1}{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
x_7 \\
x_8 \\
x_9 \\
v
\end{bmatrix}
\leq \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}

0 \leq x_i \leq 1

\sum x_i = 1
H-representation
begin
20 11 rational
0 0 0 0 0 1/2 2/5 3/10 1/5 1/10 -1
0 0 0 0 1/3 2/9 2/15 1/15 1/45 0 -1
0 0 0 1/6 1/12 1/30 1/120 0 0 -1
0 0 0 1/6 1/14 1/42 1/210 0 0 0 -1
0 0 0 1/15 1/120 0 0 0 0 0 -1
0 0 0 1/15 1/210 0 0 0 0 0 -1
0 0 0 1/5 1/45 0 0 0 0 0 0 -1
0 0 0 1/10 0 0 0 0 0 0 0 -1
1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0
-1 1 1 1 1 1 1 1 1 1 0
0 1 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 1 0 0
end
maximize
0 0 0 0 0 0 0 0 0 0 1

* cdd+: Double Description Method in C++:Version 0.77(August 19, 2003)
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch
* Compiled for Rational Exact Arithmetic with GMP
*cdd LP Result
*cdd input file : 10.ine (20 x 11)
*LP solver: Dual Simplex
*LP status: a dual pair (x, y) of optimal solutions found.
*maximization is chosen.
*Objective function is
1 X[10]
*LP status: a dual pair (x, y) of optimal solutions found.

begin
primal_solution
1 :  60/247
2 :  575/1729
3 :  440/1729
4 :  30/247
5 :  12/247
6 :  0
7 :  0
8 :  0
9 :  0
10 :  6/247
dual_solution
19 :  23/1235
20 :  53/2470
17 :  23/2470
18 :  147/9880
1 :  7/247
3 :  27/247
5 :  63/247
7 :  90/247
9 :  60/247
10 :  6/247
optimal_value : 6/247
571/162 ~ .02429
6/247 ~ .02429
end

*number of pivot operations = 5
*Computation starts at Sun Feb 15 06:50:05 2015
* terminates at Sun Feb 15 06:50:05 2015
*Total processor time = 0 seconds
* = 0h 0m 0s
<table>
<thead>
<tr>
<th>60/247</th>
<th>575/1729</th>
<th>440/1729</th>
<th>30/247</th>
<th>12/247</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/247</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27/247</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1/6</td>
<td>1/12</td>
</tr>
<tr>
<td>63/247</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1/12</td>
<td>1/42</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2/15</td>
<td>1/30</td>
<td>1/210</td>
</tr>
<tr>
<td>90/247</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1/15</td>
<td>1/120</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1/5</td>
<td>1/45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60/247</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1/10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution for k=10
<table>
<thead>
<tr>
<th>k</th>
<th>primal solution</th>
<th>dual solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10645508192981161500/2005554759628164776001</td>
<td>10645508192981161500/2005554759628164776001</td>
</tr>
<tr>
<td>2</td>
<td>311397613586611188434870/96407051729532588078236807</td>
<td>2-16.2</td>
</tr>
<tr>
<td>3</td>
<td>1160462119873878916970070/96407051729532588078236807</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1612440944034884924757/5074055354185925688328253</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32525554436935813832401/5074055354185925688328253</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>95452384789288218166605/922555518942895579696046</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>139873469989158782768035/10148110708371851376656506</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>786108086596520648697555/5074055354185925688328253</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>68900733695195588092725/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>57701938641754468976460/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>42391924836751780371900/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2753737126573696865100/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>15916990323524614014600/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8228586854967489164700/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>382047354566824762900/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>159838383839272731270/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>604090345871550037500/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>206458340712361920000/461277759471447789848023</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2765067631119900000/2005554759628164776001</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1013857923141063000/2005554759628164776001</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>10645508192981161500/822277745144754755816041</td>
<td></td>
</tr>
</tbody>
</table>
$k=117 \cdot 0000000000000034624553 \quad 2^{-41.3}$

versus

$k=125$ in SS11