faster malicious 2pc with online/offline dual execution

Mike Rosulek @ Oregon State University

collaborators:

Vladimir Kolesnikov @ Alcatel-Lucent
Payman Mohassel @ Yahoo!
Peter Rindal @ Oregon State University
Ben Riva @ Bar-Ilan University
two-party secure computation

\[ f(x, y) \]
two-party secure computation

\[ f(x; y) \]
two-party secure computation

\[ f(x, y) \]
two-party secure computation

- Security against malicious adversaries

\[ f(x, y) \]
yao’s 2pc protocol

$\text{garble } f(\cdot, y)$
yao’s 2pc protocol

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

\[ \text{garbled } x \]
yao’s 2pc protocol

\[
garble f(x, y) \quad \text{garbled circuit} \quad \text{garbled } x
\]

Full security against malicious receiver
Malicious sender can construct bad garbled circuit
yao’s 2pc protocol

garbled $f(x, y)$

garbled circuit

garble $f(\cdot, y)$

$x$ OT $x$ garbled $x$

Full security against malicious receiver

Malicious sender can construct bad garbled circuit
yao’s 2pc protocol

$\text{Yao}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\text{\[f(x, y)\]}$
yao’s 2pc protocol

- Full security against malicious receiver
- Malicious sender can construct bad garbled circuit
dual execution protocol [MohasselFranklin06]
Define a common garbled encoding: $[Z]_{A,B} \overset{\text{def}}{=} [Z]_A \oplus [Z]_B$
Define a common garbled encoding: \( [Z]_{A,B} \overset{\text{def}}{=} [Z]_A \oplus [Z]_B \)

Malicious Bob learns whether \( g(x) = f(x, y) \) (only 1 bit)
Define a **common** garbled encoding: \( [Z]_{A,B} \overset{\text{def}}{=} [Z]_A \oplus [Z]_B \)

- Malicious Bob learns whether \( g(x) = f(x, y) \) (only 1 bit)
- Malicious Bob can’t predict \( [Z]_{A,B} \) for for \( z \neq f(x, y) \)
  \[ \Rightarrow \text{can’t make Alice accept incorrect output!} \]
reducing leakage [KolesnikovMohasselRivaR15]
reducing leakage [KolesnikovMohasselRivaR15]

Main idea:
- Run $\kappa$ copies of Yao’s protocol in each direction
reducing leakage \[\text{[KolesnikovMohasselRivaR15]}\]

Main idea:
- Run $\kappa$ copies of Yao’s protocol in each direction
- Cut and choose: check each garbled circuit with probability $1/2$. 
Main idea:

- Run $\kappa$ copies of Yao’s protocol in each direction
- Cut and choose: check each garbled circuit with probability $1/2$.
- Garbled circuits in same direction have same output encoding
Main idea:

- Run $\kappa$ copies of Yao’s protocol in each direction
- Cut and choose: check each garbled circuit with probability 1/2.
- Garbled circuits in same direction have same output encoding
- What to do when Alice gets disagreeing outputs?
Honest parties can compute common \( [Z^*] A, B \) def \( = [Z^*] B \oplus [Z^*] A \).
reconciliation technique

\[
\begin{align*}
&\left[ Z_1 \right]_B, \left[ Z_2 \right]_B, \ldots \\
&S_A = \left\{ \left[ Z_i \right]_{A,B} \right\}_i
\end{align*}
\]

- Honest parties can compute common \( \left[ Z^* \right]_{A,B} \stackrel{\text{def}}{=} \left[ Z^* \right]_B \oplus \left[ Z^* \right]_A \)
- If disagreeing outputs, compute set of candidates
reconciliation technique

\[
\begin{align*}
\mathbf{z}_1 & \in B, \mathbf{z}_2 \in B, \ldots \\
S_A &= \left\{ \mathbf{z}_i \right\}_{A,B} \\
S_A \cap S_B & \xrightarrow{\text{PSI}} \\
S_B & \xrightarrow{\text{PSI output identifies the “correct” } z_i}
\end{align*}
\]

- Honest parties can compute common \( \mathbf{z}^* \) \( A,B \) \( \overset{\text{def}}{=} \mathbf{z}^* \) \( B \bigoplus \mathbf{z}^* \) \( A \)
- If disagreeing outputs, compute **set of candidates**
- Do **private set intersection** on the sets!
protocol summary

\[ x \rightarrow Yao \rightarrow y \]

\[ J_1^{z_1}K_a; J_1^{z_2}K_b; \cdots \]

\[ S_A = \{ J_1^{z_1}K_a; B \} \]

\[ S_B = \{ J_1^{z_2}K_a; B \} \]

\[ S_A \setminus S_B \]

\[ \kappa \text{ instances of Yao in each direction, check random subset} \]
protocol summary

\[ \left[ z_1 \right]_B, \left[ z_2 \right]_B, \ldots \]

\[ \left[ z'_1 \right]_A, \left[ z'_2 \right]_A, \ldots \]

\[ S_A = \left\{ \left[ z_i \right]_{A,B} \right\}_i \]

\[ S_B = \left\{ \left[ z'_i \right]_{A,B} \right\}_i \]

- \( \kappa \) instances of Yao in each direction, check random subset
- Compute set of reconciliation values
protocol summary

$S_A = \{ [z_i]_{A,B} \}_{i}$

$S_B = \{ [z_i']_{A,B} \}_{i}$

$S_A \cap S_B$

- $\kappa$ instances of Yao in each direction, check random subset
- Compute set of reconciliation values
- Private set intersection to identify correct output
protocol analysis

$$S_A = \left\{ [z_i]_{A,B} \right\}_i$$

$$S_B = \left\{ [z'_i]_{A,B} \right\}_i$$
protocol analysis: corrupt Bob

$S_A = \{ [z_i]_{A,B} \}_{i}$
protocol analysis: corrupt Bob

\[
\begin{align*}
\mathcal{S}_A &= \left\{ [z_i]_{A,B} \right\}_i \\
\mathcal{S}_B &= \left\{ [z'_i]_{A,B} \right\}_i
\end{align*}
\]

\[
\begin{align*}
[z^*_1]_B, [z^*_2]_B, \ldots
\end{align*}
\]

Bob's only "useful" PSI input is \([z^*_i]_{A,B}\)
protocol analysis: corrupt Bob

Bob’s only “useful” PSI input is $[Z^*]_{A,B}$

One of Bob’s garbled circuits is correct
protocol analysis: corrupt Bob

Bob’s only “useful” PSI input is $[Z^*]_{A,B}$

One of Bob’s garbled circuits correct $\Rightarrow$ PSI output leaks nothing!
protocol analysis: corrupt Bob

Bob’s only “useful” PSI input is $[Z^*]_{A,B}$

One of Bob’s garbled circuits correct $\Rightarrow$ PSI output leaks nothing!

None of Bob’s garbled circuits correct $\Rightarrow$
protocol analysis: corrupt Bob

Bob’s only “useful” PSI input is $[Z^*]_{A,B}$

One of Bob’s garbled circuits **correct** $\implies$ PSI output leaks nothing!

None of Bob’s garbled circuits correct $\implies$ PSI output leaks just 1 bit
"dual-ex+PSI" summary

\[ \kappa \text{ garbled circuits in each direction (can be done simultaneously)} \]

Adversary cannot violate output correctness

Adversary learns a single bit with probability \( 1/2^\kappa \); only when:

- All opened circuits are correct
- All evaluated circuits are incorrect
Online/offline, multi-execution setting

- Reducing # of garbled circuits

Adapting “dual-execution+PSI” protocol to online/offline setting:
[RindalR16]

- Ensuring input consistency
- Lightweight private set intersection

Implementation, performance

- Comparison to [LindellRiva15] and info-theoretic protocols
online/offline setting

Want to do 2PC of same circuit \( N \) times?

\[ \text{[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]} \]
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

generate a lot of garbled circuits
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

open and check some fraction of them
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

pick a random “bucket” of available circuits and evaluate them
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

pick a random “bucket” of available circuits and evaluate them
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

pick a random “bucket” of available circuits and evaluate them
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

pick a random “bucket” of available circuits and evaluate them
online/offline setting

Want to do 2PC of same circuit $N$ times?

[HuangKatzKolesnikovKumaresanMalozemoff14,LindellRiva14]

- for security $1/2^\kappa$, need $O(\kappa / \log N)$ circuits per execution
- example: $N = 1024$, $\kappa = 40 \implies 4$ circuits per execution
online/offline dual-ex [RindalR16]
online/offline dual-ex [RindalR16]
online/offline dual-ex [RindalR16]

open & check some fraction

offline phase

online phase
online/offline dual-ex [RindalR16]
online/offline dual-ex [RindalR16]
challenge #1: input consistency

How to ensure same inputs in Alice/Bob circuits?

[KolesnikovMohasselRivaR15,shelatShen13] technique incompatible with offline circuit garbling
pre-computing OTs [Beaver95]

[for simplicity: 1 input bit from Alice]
pre-computing OTs [Beaver95]

\[ r, k_r \quad \text{R-OT} \quad k_0, k_1 \]

offline phase

online phase
pre-computing OTs\cite{Beaver95}

\begin{align*}
r, k_r & \quad \text{R-OT} \quad k_0, k_1 \\
\text{offline phase} & \quad \text{online phase} \\
d & = r \oplus x \\
k_0 \oplus [d]_B; \quad k_1 \oplus [\overline{d}]_B
\end{align*}
consistency b/w A & B circuits

\[ r, k_r \quad \text{R-OT} \quad k_0, k_1 \]

offline phase

online phase

\[ d = r \oplus x \]

\[ [X]_B \quad k_0 \oplus [d]_B; \quad k_1 \oplus [\overline{d}]_B \]
consistency b/w A & B circuits

\[
\begin{align*}
\text{offline phase} \quad d &= r \oplus x \\
\text{online phase} \\
\end{align*}
\]
consistency b/w A & B circuits

\[ \begin{align*}
\text{offline phase} & \quad \text{online phase} \\
\mathbf{r}, \mathbf{k}_r & \quad \text{R-OT} \quad \mathbf{k}_0, \mathbf{k}_1 \\
C_0 &= \text{Com}([\mathbf{r}]_A); \quad C_1 = \text{Com}([\bar{\mathbf{r}}]_A) \\
d &= \mathbf{r} \oplus \mathbf{x} \\
k_0 \oplus [\mathbf{d}]_B; \quad k_1 \oplus [\bar{\mathbf{d}}]_B & \quad \text{? } [\mathbf{x}]_A ?
\end{align*} \]
consistency b/w A & B circuits

\[
\begin{align*}
C_0 &= \text{Com}(\llbracket r \rrbracket_A); \\
C_1 &= \text{Com}(\llbracket \bar{r} \rrbracket_A)
\end{align*}
\]

For offline phase and online phase:

\[
\begin{align*}
d &= r \oplus x; \\
\text{Open}(C_d)
\end{align*}
\]

\[
\begin{align*}
k_0 \oplus \llbracket d \rrbracket_B; \\
k_1 \oplus \llbracket \bar{d} \rrbracket_B
\end{align*}
\]

\[
\begin{align*}
[\cdot]_A, x \\
[\cdot]_B
\end{align*}
\]

\[
[\cdot]_A, x \\
[\cdot]_B
\]

\[
[\cdot]_A, x \\
[\cdot]_B
\]

✓
consistency b/w A & B circuits

\[ \begin{align*}
C_0 &= \text{Com}([r]_A); \\
C_1 &= \text{Com}([\overline{r}]_A)
\end{align*} \]

Can check consistency of commitments in cut-and-choose:

- Alice can show \( k_r \) to prove what \( r \) she got from OT
- at least one pair of A/B circuits with consistency
input consistency

Within each bucket,

- Alice uses same input $x$ on all Bob-circuits (easy)
- At least one Alice-circuit where Alice uses $x$ (except prob $1/2^\kappa$)

Suffices for security!

Zero online cost for input consistency!
challenge #2: psi

How to efficiently instantiate PSI?
closer look at PSI

\[ S_A = \left\{ [z_i]_{A,B} \right\}_i \]

Bob's only "useful" PSI input is \( [z^*]_{A,B} \)
closer look at PSI

\[
S_A = \{ [z_i]_{A,B} \}_{i=1}^n
\]

\[
\begin{bmatrix} z_1 \end{bmatrix}_B, \begin{bmatrix} z_2 \end{bmatrix}_B, \ldots
\]

\[
[ z^* ]_A
\]

Bob’s only “useful” PSI input is \([z]_{A,B}\).

\[ \star \]

Simulator knows \([z]_{A,B}\); rest of \(S_A\) independent of Adv’s view.
closer look at PSI

\[ [z_1]_B, [z_2]_B, \ldots \]

\[ z^* = \text{correct output} \]

\[ [z^*]_A \]

\[ S_A = \left\{ [z_i]_{A,B} \right\}_i \]

\[ S_A \cap S_B \]

- Simulator knows \([z^*]_{A,B}\); rest of \(S_A\) independent of Adv’s view

- Simulator **does not need to extract** Adv’s input \(S_B\)!

  \ldots it suffices to check whether \([z^*]_{A,B} \in S_B\)
instantiating psi

[KolesnikovMohasselRivaR15]:
  ▶ Suggest using fully malicious PSI subprotocol

[RindalR16]:
  ▶ PSI protocol with “non-extracting security” suffices
  ▶ Implementation uses semi-honest PSI protocol of [PinkasSchneiderZohner14]
  ▶ Very cheap, based on pre-processed OTs (no public-key operations)
comparison to [LindellRiva15]

[LindellRiva14/15]: online/offline, malicious security, based on “traditional cut and choose” [Lindell13]:

Two phases:
1. $B$ circuits computing $f(x, y)$
comparison to [LindellRiva15]

[LindellRiva14/15]: online/offline, malicious security, based on “traditional cut and choose” [Lindell13]:

Two phases:

1. $B$ circuits computing $f(x, y)$
2. $\sim 3B$ circuits computing:
   
   "if Bob can prove Alice cheated in phase 1, then reveal $x$ to Bob"
protocol comparison

[KolesnikovMohasselRiva15,Rindal16]:

[Yao]:

[PSI] ▶

B primary circuits in each direction (evaluated simultaneously!)

PSI computation scales only with B

[LindellRiva14/15]:

Yao

Yao

Yao

Yao

Yao

Yao

Yao

Yao

Yao

Yao

Yao

B primary circuits ▶

Aux computation scales as $3^B$ input
protocol comparison

[KolesnikovMohasselRivaR15,RindalR16]:

- $B$ primary circuits in each direction (evaluated simultaneously!)

[LindellRiva14/15]:

- $B$ primary circuits

$\Rightarrow \quad $ $B$ primary circuits in each direction (evaluated simultaneously!)
protocol comparison

[KolesnikovMohasselRivaR15,RindalR16]:

- \( B \) primary circuits in each direction (evaluated simultaneously!)
- PSI computation scales only with \( B \)

[LindellRiva14/15]:

- \( B \) primary circuits
- Aux computation scales as \( 3B \cdot l_{input} \)
a closer look at $\kappa$

$\kappa_C$: Computational security parameter (e.g., 128)

$\kappa_s$: Statistical security parameter: security properties violated with probability $1/2^{\kappa_s}$ (e.g., 40)

“Traditional cut-and-choose” [LindellRiva14/15]:

- When cut-and-choose fails, adversary can completely break privacy & correctness

$\Rightarrow$ # of garbled circuits scales with $\kappa_s$
a closer look at $\kappa$

$\kappa_c$: Computational security parameter (e.g., 128)

$\kappa_s$: Statistical security parameter: security properties violated with probability $1/2^{\kappa_s}$ (e.g., 40)

$\kappa_b$: Bit-leaking parameter: privacy violated by a single bit (other security maintained) with probability $1/2^{\kappa_b}$

"Traditional cut-and-choose" [LindellRiva14/15]:

- When cut-and-choose fails, adversary can completely break privacy & correctness

$\Rightarrow$ # of garbled circuits scales with $\kappa_s$
a closer look at $\kappa$

$\kappa_C$: Computational security parameter (e.g., 128)

$\kappa_s$: Statistical security parameter: security properties violated with probability $1/2^{\kappa_s}$ (e.g., 40)

$\kappa_b$: Bit-leaking parameter: privacy violated by a single bit (other security maintained) with probability $1/2^{\kappa_b}$

“Traditional cut-and-choose” [LindellRiva14/15]:

- When cut-and-choose fails, adversary can completely break privacy & correctness

$\Rightarrow$ # of garbled circuits scales with $\kappa_s$

Dual-execution approach [KolesnikovMohasselRivaR15,RindalR16]:

- When cut-and-choose fails, adversary learns only a bit

$\Rightarrow$ # of garbled circuits scales with $\kappa_b \leq \kappa_s$
some parameter possibilities

[RindalR16] with $\kappa_s = \kappa_b = 40$:

- same security as [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits
some parameter possibilities

[RindalR16] with $\kappa_s = \kappa_b = 40$:

- same security as [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits

[RindalR16] with $\kappa_s = 40$, $\kappa_b = 30$:

- same security as [LindellRiva] with $\kappa_s = 40$, except slightly higher probability of leaking single bit
- fewer garbled circuits (25% savings for $N = 1024$, 40% for $N = 512$)
some parameter possibilities

[RindalR16] with $\kappa_s = \kappa_b = 40$:

- same security as [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits

[RindalR16] with $\kappa_s = 40, \kappa_b = 30$:

- same security as [LindellRiva] with $\kappa_s = 40$, except slightly higher probability of leaking single bit
- fewer garbled circuits (25% savings for $N = 1024$, 40% for $N = 512$)

[RindalR16] with $\kappa_s = 80, \kappa_b = 40$:

- strictly stronger security than [LindellRiva] with $\kappa_s = 40$
- same # of garbled circuits (only PSI cost increases)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AES circuit</td>
<td>5.1ms</td>
<td>1.3ms</td>
<td>74ms</td>
<td>7ms</td>
<td>high?</td>
<td>6ms</td>
</tr>
<tr>
<td>SHA256 circuit</td>
<td>48ms</td>
<td>8.4ms</td>
<td>206ms</td>
<td>33ms</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Implementation

<table>
<thead>
<tr>
<th></th>
<th>[RindalR16] offline</th>
<th>online</th>
<th>[LindellRiva] offline</th>
<th>online</th>
<th>[DamgårdZakarias15] offline</th>
<th>online</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES circuit</td>
<td>5.1ms</td>
<td>1.3ms</td>
<td>74ms</td>
<td>7ms</td>
<td>high?</td>
<td>6ms</td>
</tr>
<tr>
<td>SHA256 circuit</td>
<td>48ms</td>
<td>8.4ms</td>
<td>206ms</td>
<td>33ms</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Amortized cost over $N = 1024$ executions:

- same hardware, LAN connection (Amazon c4.8xlarge = 36 core, 64GB RAM)
- same security ($\kappa_s = \kappa_b = 40$)
### Implementation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AES circuit</td>
<td>5.1ms</td>
<td>74ms</td>
<td>high?</td>
<td>1.3ms</td>
<td>7ms</td>
<td>6ms</td>
</tr>
<tr>
<td>SHA256 circuit</td>
<td>48ms</td>
<td>206ms</td>
<td>-</td>
<td>8.4ms</td>
<td>33ms</td>
<td>-</td>
</tr>
</tbody>
</table>

Amortized cost over \( N = 1024 \) executions:

- same hardware, LAN connection
  (Amazon c4.8xlarge = 36 core, 64GB RAM)
- same security \( \kappa_s = \kappa_b = 40 \)

Maximum **throughput**: 0.26ms / AES block (3800+ Hz)

- [DamgårdZakarias15] reports 0.4ms
summary

Online-offline dual execution:

- Fastest 2PC with malicious security to date: 1.3ms AES
- Some protocol advantages over “classic” cut-and-choose

Future work:
summary

Online-offline dual execution:

- Fastest 2PC with malicious security to date: 1.3ms AES
- Some protocol advantages over “classic” cut-and-choose

Future work:

<table>
<thead>
<tr>
<th></th>
<th>garbled circ</th>
<th>info-theoretic</th>
</tr>
</thead>
<tbody>
<tr>
<td>online latency</td>
<td>low ✓ ✓</td>
<td>low ✓</td>
</tr>
<tr>
<td>online throughput</td>
<td>high ✓</td>
<td>high ✓ ✓ ✓</td>
</tr>
<tr>
<td>constant rounds?</td>
<td>yes ✓</td>
<td>no ✗</td>
</tr>
<tr>
<td>offline time</td>
<td>low ✓</td>
<td>(very) high ✗</td>
</tr>
<tr>
<td>function-indep</td>
<td>no ✗</td>
<td>yes ✓</td>
</tr>
<tr>
<td>pre-processing?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
summary

Online-offline dual execution:

- Fastest 2PC with malicious security to date: 1.3ms AES
- Some protocol advantages over “classic” cut-and-choose

Future work: Combine GC with info-theoretic? (at least for 2-party)

<table>
<thead>
<tr>
<th></th>
<th>garbled circ</th>
<th>info-theoretic</th>
<th>hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>online latency</td>
<td>low ✓ ✓</td>
<td>low ✓</td>
<td>low ✓</td>
</tr>
<tr>
<td>online throughput</td>
<td>high ✓</td>
<td>high ✓ ✓</td>
<td>high ✓ ✓</td>
</tr>
<tr>
<td>constant rounds?</td>
<td>yes ✓</td>
<td>no ✗</td>
<td>no ✗</td>
</tr>
<tr>
<td>offline time</td>
<td>low ✓</td>
<td>(very) high ✗</td>
<td>low ✓</td>
</tr>
<tr>
<td>function-indep</td>
<td>no ✗</td>
<td>yes ✓</td>
<td>yes ✓</td>
</tr>
<tr>
<td>pre-processing?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
the end; thanks.

Richer Efficiency/Security Tradeoffs in 2PC
Vladimir Kolesnikov, Payman Mohassel, Ben Riva & Mike Rosulek
ia.cr/2015/055

Faster Malicious 2-party Secure Computation with Online/Offline Dual Execution
Peter Rindal & Mike Rosulek
in submission