University of Stuttgart
Institute of
Information Security

Actively Secure Polynomial Evaluation from Shared Polynomial Encodings

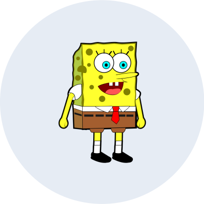
Pascal Reisert, Marc Rivinius
Toomas Krips, **Sebastian Hasler**, and Ralf Küsters

MPC Setup

1

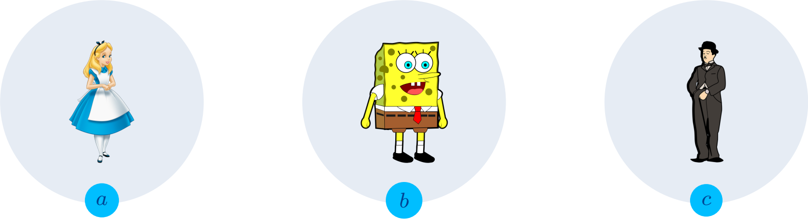
MPC Setup

- Multiple parties want to compute a function f on secret inputs $a, b, c \in \mathbb{F}_q$.

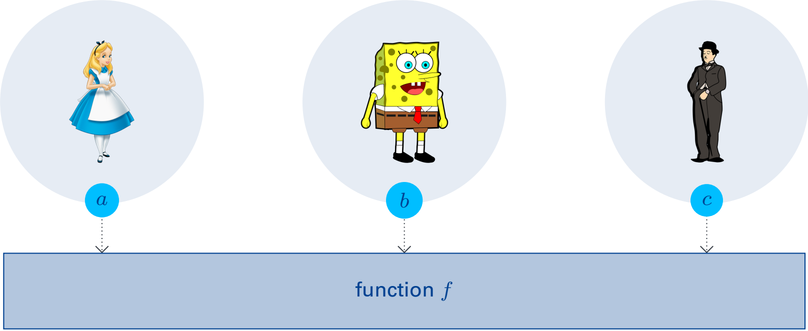

MPC Setup

- Multiple parties want to compute a function f on secret inputs $a, b, c \in \mathbb{F}_q$.



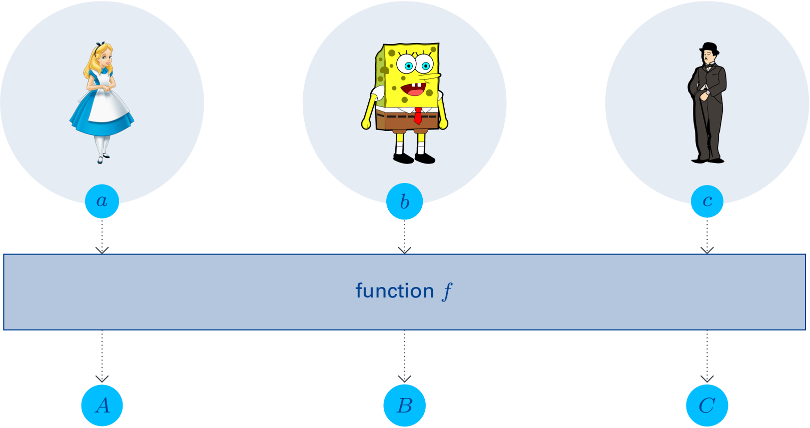
MPC Setup

- Multiple parties want to compute a function f on secret inputs $a, b, c \in \mathbb{F}_q$.



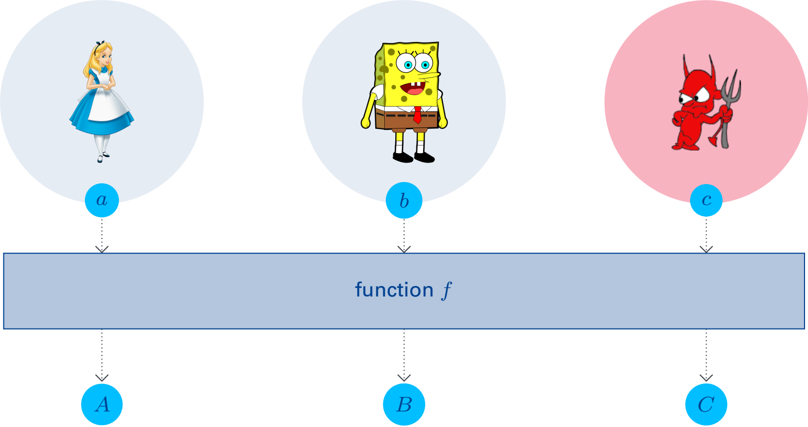
MPC Setup

- Multiple parties want to compute a function f on secret inputs $a, b, c \in \mathbb{F}_q$.



MPC Setup

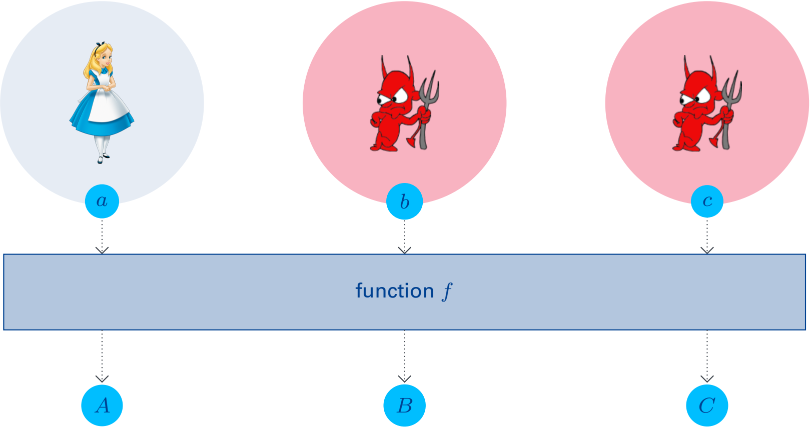
- Multiple parties want to compute a function f on secret inputs $a, b, c \in \mathbb{F}_q$.



- Up to $n-1$ actively malicious parties.

MPC Setup

- Multiple parties want to compute a function f on secret inputs $a, b, c \in \mathbb{F}_q$.



- Up to $n-1$ actively malicious parties.

SPDZ and Generalizations

- State of the art maliciously secure protocol to compute arithmetic functions:

  SPDZ [Dam+12] and its improvements, e.g., Overdrive [KPR18]

SPDZ and Generalizations

- State of the art maliciously secure protocol to compute arithmetic functions:

  SPDZ [Dam+12] and its improvements, e.g., Overdrive [KPR18]

- *Additions* and *multiplications with public values* is non-interactive

SPDZ and Generalizations

- State of the art maliciously secure protocol to compute arithmetic functions:

  SPDZ [Dam+12] and its improvements, e.g., Overdrive [KPR18]

- *Additions* and *multiplications with public values* is non-interactive

- *Multiplications* need interaction and preprocessed data, classically **Beaver triples**.

SPDZ and Generalizations

- State of the art maliciously secure protocol to compute arithmetic functions:

  SPDZ [Dam+12] and its improvements, e.g., Overdrive [KPR18]

- *Additions* and *multiplications with public values* is non-interactive

- *Multiplications* need interaction and preprocessed data, classically **Beaver triples.**

- *Classical Approach:*

  - Only addition & multiplication gates

  - One communication round and sufficiently many Beaver triples for each layer of multiplications

SPDZ and Generalizations

- State of the art maliciously secure protocol to compute arithmetic functions:

  SPDZ [Dam+12] and its improvements, e.g., Overdrive [KPR18]

- *Additions* and *multiplications with public values* is non-interactive

- *Multiplications* need interaction and preprocessed data, classically **Beaver triples**.

- *Classical Approach:*

  - Only addition & multiplication gates

  - One communication round and sufficiently many Beaver triples for each layer of multiplications

$\Rightarrow$ *Our Approach:* Replace Beaver triples by a new form of structured randomness to reduce communication down to **one** (amortized) round.

New Forms of Structured Randomness

- Optimized structured randomness has already been introduced in recent years
- E.g., to optimize typical operations in privacy-preserving ML:

New Forms of Structured Randomness

- Optimized structured randomness has already been introduced in recent years
- E.g., to optimize typical operations in privacy-preserving ML:
  - *Matrix triples* for matrix multiplications, e.g., in [Che+20; Rei+23]

New Forms of Structured Randomness

- Optimized structured randomness has already been introduced in recent years
- E.g., to optimize typical operations in privacy-preserving ML:
  - *Matrix triples* for matrix multiplications, e.g., in [Che+20; Rei+23]
  - *Convolution triples* for Tensor convolutions, e.g., in [Che+20; Riv+23]

New Forms of Structured Randomness

- Optimized structured randomness has already been introduced in recent years
- E.g., to optimize typical operations in privacy-preserving ML:
  - *Matrix triples* for matrix multiplications, e.g., in [Che+20; Rei+23]
  - *Convolution triples* for Tensor convolutions, e.g., in [Che+20; Riv+23]
- Another typical operation is polynomial evaluation. Unfortunately, for *multivariate polynomials*, known one-round solutions [CWB18; Cou19] come with exponential size of the structured randomness

  $\Rightarrow$ inefficient for large polynomial degrees

New Forms of Structured Randomness

- Optimized structured randomness has already been introduced in recent years
- E.g., to optimize typical operations in privacy-preserving ML:
  - *Matrix triples* for matrix multiplications, e.g., in [Che+20; Rei+23]
  - *Convolution triples* for Tensor convolutions, e.g., in [Che+20; Riv+23]
- Another typical operation is polynomial evaluation. Unfortunately, for *multivariate polynomials*, known one-round solutions [CWB18; Cou19] come with exponential size of the structured randomness

  $\Rightarrow$ inefficient for large polynomial degrees
- We address this problem:

  New form of (moderately-sized) structured randomness (called **polytuples**) to evaluate *multivariate polynomials* and *comparisons* in one (amortized) round

Randomized Encodings

2

Randomized Encodings

- We use randomized encodings that exist for more than 20 years [IK00], but are not explicitly used in SPDZ-like protocols yet.

Randomized Encodings

- We use randomized encodings that exist for more than 20 years [IK00], but are not explicitly used in SPDZ-like protocols yet.

- *Definition.* Let $X, Y, \hat{Y}, A$ be finite sets and let $f : X \to Y$. A function $\hat{f} : X \times A \to \hat{Y}$ is called *randomized encoding* of f if the following holds:
  - **Correctness.** There exists a reconstruction algorithm $\mathrm{Rec} : \hat{Y} \to Y$ such that

$$\begin{tikzcd} X \times A \arrow[r, "\hat{f}"] \arrow[d, "\mathrm{pr}_1"'] & \hat{Y} \arrow[d, "\mathrm{Rec}"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

    commutes, where $\mathrm{pr}_1 : X \times A \to X, (x, a) \mapsto x$ is the projection.
  - **Privacy.** There exists a simulator Sim such that $\mathrm{Sim}(f(x))$ and $\hat{f}(x, a)$ are identically distributed for all $x \in X$ if a is sampled uniformly from A.

Randomized Encodings

Example. For $f(x_0, x_1) = x_0 x_1$ take

$$\hat{f} = (y_0, y_1, y_2) = (x_0 - a_0, x_1 - a_1, a_1(x_0 - a_0) + a_0(x_1 - a_1) + a_0 a_1)$$

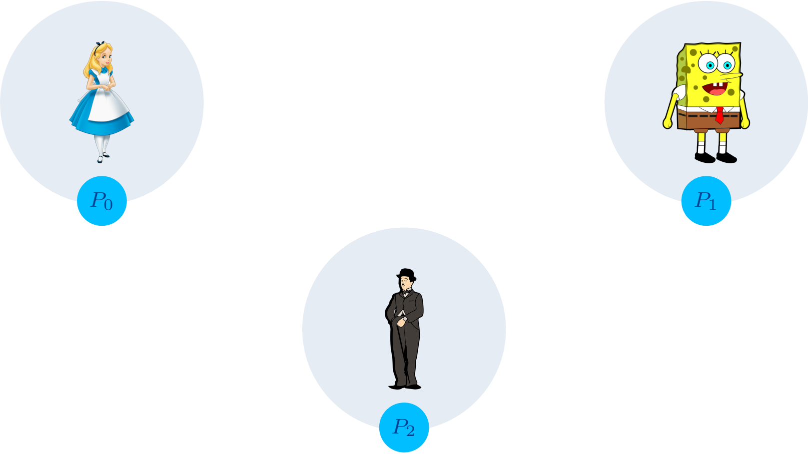
for $a_0, a_1 \in A$ and reconstruct by

$$\mathrm{Rec}(y_0, y_1, y_2) = y_0 y_1 + y_2 = x_0 x_1$$

.

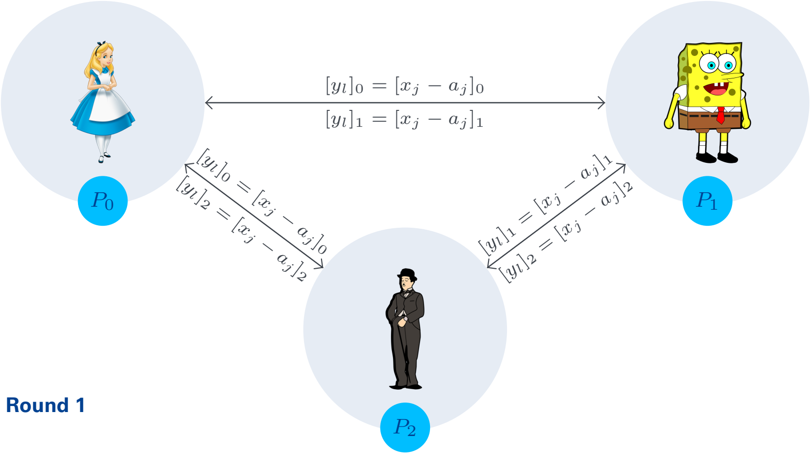
Randomized Encodings as MPC Protocols

- For $f(x_0, \ldots, x_{m-1})$: Compute randomized encoding $\hat{f} = (y_l)_{0 \leq l < k}$ on shares of inputs $[x_j]$ and randomness $[a_t]$.



Randomized Encodings as MPC Protocols

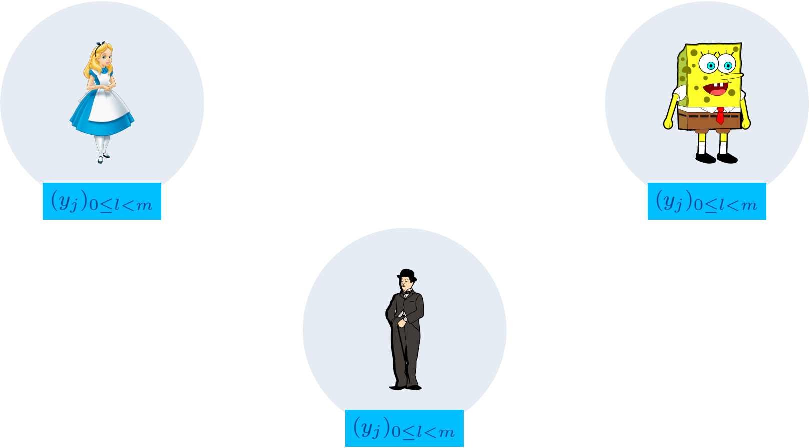
- The parties exchange all shares of the masked input values $y_l = x_l - a_l$, $0 \leq l < m$, parallely and each party reconstructs the y_l locally.



Round 1

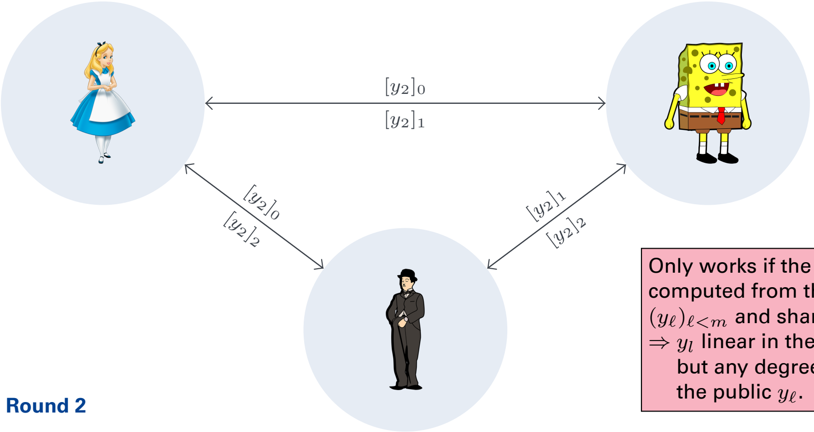
Randomized Encodings as MPC Protocols

- The parties exchange all shares of the masked input values $y_l = x_l - a_l$, $0 \leq l < m$, parallely and each party reconstructs the y_l locally.



Randomized Encodings as MPC Protocols

- The parties compute shares of the $y_l, l \geq m$ locally, e.g.,
  $[y_2] = [a_1](x_0 - a_0) + [a_0](x_1 - a_1) + [a_0 a_1]$ for $m = 2$ and $f(x_0, x_1) = x_0 x_1$.
  They exchange the shares and reconstruct with $\mathrm{Rec}(y_0, y_1, y_2) = y_0 y_1 + y_2$.



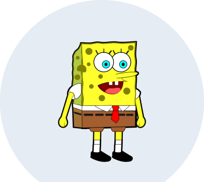
Only works if the $[y_l]$ can be computed from the public $(y_\ell)_{\ell<m}$ and shares locally.
$\Rightarrow$ y_l linear in the shares, but any degree in the public y_ℓ.

Round 2

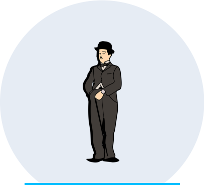
Randomized Encodings as MPC Protocols

- The parties compute shares of the $y_l, l \geq m$ locally, e.g.,
  $[y_2] = [a_1](x_0 - a_0) + [a_0](x_1 - a_1) + [a_0 a_1]$ for $m = 2$ and $f(x_0, x_1) = x_0 x_1$.
  They exchange the shares and reconstruct with $\mathrm{Rec}(y_0, y_1, y_2) = y_0 y_1 + y_2$.

$\mathrm{Rec}(y_0, y_1, y_2)$



$\mathrm{Rec}(y_0, y_1, y_2)$



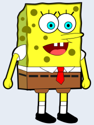
$\mathrm{Rec}(y_0, y_1, y_2)$

Only works if the $[y_l]$ can be computed from the public $(y_\ell)_{\ell<m}$ and shares locally.
$\Rightarrow$ y_l linear in the shares, but any degree in the public y_ℓ.

Randomized Encodings as MPC Protocols

- The parties compute shares of the $y_l, l \geq m$ locally, e.g., $[y_2] = [a_1](x_0 - a_0) + [a_0](x_1 - a_1) + [a_0 a_1]$ for $m = 2$ and $f(x_0, x_1) = x_0 x_1$. They exchange the shares and reconstruct with $\mathrm{Rec}(y_0, y_1, y_2) = y_0 y_1 + y_2$.

$f(x_0, x_1)$



$f(x_0, x_1)$

$f(x_0, x_1)$

Only works if the $[y_l]$ can be computed from the public $(y_\ell)_{\ell<m}$ and shares locally.
$\Rightarrow$ y_l linear in the shares, but any degree in the public y_ℓ.

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

$$x_0 \cdots x_3 = \boxed{y_0} \cdot \boxed{y_1} + \boxed{y_2}$$

$$\boxed{x_0x_1 - a_{01}} \quad \boxed{x_2x_3 - a_{23}} \quad \boxed{x_0x_1a_{23} - a_{01,23}} + \boxed{x_2x_3a_{01} - a_{23,01}} + \boxed{a_{01,23} + a_{23,01} - a_{01}a_{23}}$$

We get 4 polynomials of degree 2 in the x_i.

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + y_2$$

$$\boxed{x_0 x_1 - a_{01}} = y'_0 \cdot y'_1 + y'_2 - a_{01}$$

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + y_2$$

$$\boxed{x_0 x_1 - a_{01}} = \boxed{y'_0} \cdot \boxed{y'_1} + \boxed{y'_2 - a_{01}}$$

$$\boxed{x_0 - a_0} \qquad \boxed{x_1 - a_1} \qquad \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{a_{0,1} + a_{1,0} - a_0 a_1 - a_{01}}$$

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + y_2$$

$$\boxed{x_0 x_1 - a_{01}} = \boxed{y'_0} \cdot \boxed{y'_1} + \boxed{y'_2 - a_{01}}$$

$$\boxed{x_0 - a_0} \quad \boxed{x_1 - a_1} \quad \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{a_{0,1} + a_{1,0} - a_0 a_1 - a_{01}}$$

- Similarly we get 4 polynomials of degree 1 in the x_i for all of the degree-2 terms.

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + y_2$$

$$\boxed{x_0 x_1 - a_{01}} = \boxed{y'_0} \cdot \boxed{y'_1} + \boxed{y'_2 - a_{01}}$$

$$\boxed{x_0 - a_0} \qquad \boxed{x_1 - a_1} \qquad \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{a_{0,1} + a_{1,0} - a_0 a_1 - a_{01}}$$

- Similarly we get 4 polynomials of degree 1 in the x_i for all of the degree-2 terms.
- The approach extends to degree $m = 2^n$, i.e., we split $x_0 \cdots x_{m-1}$ in 4 degree $\frac{m}{2}$ terms and then each of these terms again in 4 $\frac{m}{4}$ terms, and so on.

Results on Randomized Encodings

- Randomized Encodings can be concatenated and composed [AIK06], e.g.,

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + y_2$$

$$\boxed{x_0x_1 - a_{01}} = \boxed{y'_0} \cdot \boxed{y'_1} + \boxed{y'_2 - a_{01}}$$

$$\boxed{x_0 - a_0} \qquad \boxed{x_1 - a_1} \qquad \boxed{x_0a_1 - a_{0,1}} + \boxed{x_1a_0 - a_{1,0}} + \boxed{a_{0,1} + a_{1,0} - a_0a_1 - a_{01}}$$

- Similarly we get 4 polynomials of degree 1 in the x_i for all of the degree-2 terms.
- The approach extends to degree $m = 2^n$, i.e., we split $x_0 \cdots x_{m-1}$ in 4 degree $\frac{m}{2}$ terms and then each of these terms again in 4 $\frac{m}{4}$ terms, and so on.

$\Rightarrow$ Output size of the encoding is in $\mathcal{O}(4^n) = \mathcal{O}(m^2)$ (cf. [Cra+03] for a similar result).

$\Rightarrow$ All terms are of total degree at most 3 (counting inputs and randomness to the degree), which is the theoretical minimum established in [Cra+03].

Our Randomized Encodings

3

Our New Randomized Encodings

- We can improve over the abovementioned generic approach with standard composition and concatenation:

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + y_2$$

$$\boxed{x_0 - a_0}\ \boxed{x_1 - a_1}\ \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{r_1}$$

Our New Randomized Encodings

- We can improve over the abovementioned generic approach with standard composition and concatenation:

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + \boxed{y_2}$$

$$\boxed{x_2 x_3 a_{01} - a_{23,01}} + \boxed{x_0 x_1 a_{23} - a_{01,23}} + \boxed{r_0}$$

$$\boxed{x_0 - a_0}\ \boxed{x_1 - a_1}\ \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{r_1}$$

Our New Randomized Encodings

- We can improve over the abovementioned generic approach with standard composition and concatenation:

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + \boxed{y_2}$$

$$\boxed{x_2 x_3 a_{01} - a_{23,01}} + \boxed{x_0 x_1 a_{23} - a_{01,23}} + \boxed{r_0}$$

$$\boxed{x_0 - a_0}\ \boxed{x_1 - a_1}\ \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{r_1}\ \boxed{x_0 - a_0}\ \boxed{x_1 a_{23} - a_{1,23}}\ \boxed{x_0 a_{1,23} - a_{0,123}} + \boxed{x_1 a_0 a_{23} - a_{1,0,23}} + \boxed{r_2}$$

Our New Randomized Encodings

- We can improve over the abovementioned generic approach with standard composition and concatenation:

$$x_0 \cdots x_3 = \boxed{y_0} \cdot y_1 + \boxed{y_2}$$

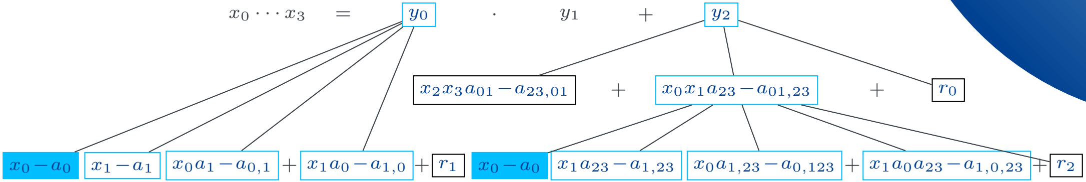
$$\boxed{x_2 x_3 a_{01} - a_{23,01}} + \boxed{x_0 x_1 a_{23} - a_{01,23}} + \boxed{r_0}$$

$$\boxed{x_0 - a_0}\ \boxed{x_1 - a_1}\ \boxed{x_0 a_1 - a_{0,1}} + \boxed{x_1 a_0 - a_{1,0}} + \boxed{r_1}\ \boxed{x_0 - a_0}\ \boxed{x_1 a_{23} - a_{1,23}}\ \boxed{x_0 a_{1,23} - a_{0,123}} + \boxed{x_1 a_0 a_{23} - a_{1,0,23}} + \boxed{r_2}$$

- If polynomials can be used in different reconstructions, we only need to output them **once**! Then the output size is reduced.

Our New Randomized Encodings

- We can improve over the abovementioned generic approach with standard composition and concatenation:



- If polynomials can be used in different reconstructions, we only need to output them **once**! Then the output size is reduced.

- Note: generally for two randomized encodings $(y_l)_{l\geq 0}$ of f and $(y'_l)_{l\geq 0}$ of g with $y_0 = y'_0$, the reduced concatenation $(y_0, y_l, y'_l)_{l\geq 1}$ is **not a secure** randomized encoding of $f \times g$.

Our New Randomized Encodings

- Fortunately, for the multiple terms in our construction, we could show that the reduced concatenation provides a **secure randomized encoding**.

Our New Randomized Encodings

- Fortunately, for the multiple terms in our construction, we could show that the reduced concatenation provides a **secure randomized encoding**.

- We are then able to compute a product $x_0 \cdots x_{m-1}$ with a randomized encoding of output size $\mathcal{O}(m \log(m))$.

Our New Randomized Encodings

- We then extend our approach to general monomials of the form $x_0^{d_0} \cdots x_{m-1}^{d_{m-1}}$ and further to arbitrary polynomials.

Our New Randomized Encodings

- We then extend our approach to general monomials of the form $x_0^{d_0} \cdots x_{m-1}^{d_{m-1}}$ and further to arbitrary polynomials.

- We further extend the approach to other tree structures, where we can multiply **any number** of inputs in one multiplication node (instead of 2 in the example above), e.g.:

$$x_0 \cdots x_8 = (x_0 x_1 x_2 - a_{012})(x_3 x_4 x_5 - a_{345})(x_6 x_7 x_8 - a_{678}) + (\text{degree } \leq 3 \text{ in the } x_j)$$

Our New Randomized Encodings

- We then extend our approach to general monomials of the form $x_0^{d_0} \cdots x_{m-1}^{d_{m-1}}$ and further to arbitrary polynomials.

- We further extend the approach to other tree structures, where we can multiply **any number** of inputs in one multiplication node (instead of 2 in the example above), e.g.:

$$x_0 \cdots x_8 = (x_0x_1x_2 - a_{012})(x_3x_4x_5 - a_{345})(x_6x_7x_8 - a_{678}) + (\text{degree } \leq 3 \text{ in the } x_j)$$

- We present a recursive formula to compute the exact output size and randomness size for all resulting randomized encodings in our paper.

Comparison and Benchmarks

4

Our Resulting MPC protocols

- We compute the structured randomness of the randomized encoding as an authenticated tuple. We call this **polytuple**.

Our Resulting MPC protocols

- We compute the structured randomness of the randomized encoding as an authenticated tuple. We call this **polytuple**.
- The offline phase is linear in the tuple size.

Our Resulting MPC protocols

- We compute the structured randomness of the randomized encoding as an authenticated tuple. We call this **polytuple**.
- The offline phase is linear in the tuple size.
- In the online phase, the number of elements sent equals the output size of the randomized encoding.

Our Resulting MPC protocols

- We compute the structured randomness of the randomized encoding as an authenticated tuple. We call this **polytuple**.
- The offline phase is linear in the tuple size.
- In the online phase, the number of elements sent equals the output size of the randomized encoding.
- Our protocol needs one round to exchange the masks and one opening round (which in the malicious setup includes a MAC check).

Our Resulting MPC protocols

- We compute the structured randomness of the randomized encoding as an authenticated tuple. We call this **polytuple**.
- The offline phase is linear in the tuple size.
- In the online phase, the number of elements sent equals the output size of the randomized encoding.
- Our protocol needs one round to exchange the masks and one opening round (which in the malicious setup includes a MAC check).
- Our protocol can be used in a multi-round fashion where $f_1 \circ \cdots \circ f_n$ is evaluated in n rounds plus one opening round.

Theoretical Comparison

- We get the following comparison for the computation of $x_1^{d/m} \cdots x_{m-1}^{d/m}$ of degree d with $d/m \in \mathbb{N}$ per party:

Approach	Rounds	Bandwidth	Tuple Size
Beaver Triples e.g. for $d = m = 16$	$\lceil \log d \rceil$ 4	$2(m-1)\lceil \log \frac{d}{m} \rceil$ 30	$3(m-1)\lceil \log d/m \rceil$ 45
Tuples from [CWB18] e.g. for $d = m = 16$	1 1	m 16	$(\frac{d}{m} + 1)^m - 1$ 65535
Example of a Polytuple e.g. for $d = m = 16$	1 1	$\mathcal{O}(m \log(m))$ 41	$\mathcal{O}(d(\log m)^2)$ 149

- The round count does not include the opening round (which is needed for all compared protocols).

Theoretical Comparison

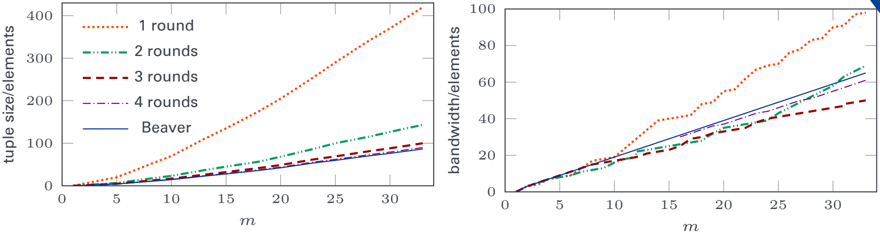
- We get the following comparison for the computation of $x_1^{d/m} \cdots x_{m-1}^{d/m}$ of degree d with $d/m \in \mathbb{N}$ per party:

Approach	Rounds	Bandwidth	Tuple Size
Beaver Triples e.g. for $d = m = 16$	$\lceil \log d \rceil$ 4	$2(m-1)\lceil \log \frac{d}{m} \rceil$ 30	$3(m-1)\lceil \log d/m \rceil$ 45
Tuples from [CWB18] e.g. for $d = m = 16$	1 1	m 16	$(\frac{d}{m} + 1)^m - 1$ 65535
Example of a Polytuple e.g. for $d = m = 16$	1 1	$\mathcal{O}(m \log(m))$ 41	$\mathcal{O}(d(\log m)^2)$ 149

- The round count does not include the opening round (which is needed for all compared protocols).
- The table includes only one example of a polytuple; depending on the setup, we can get different trade-offs.

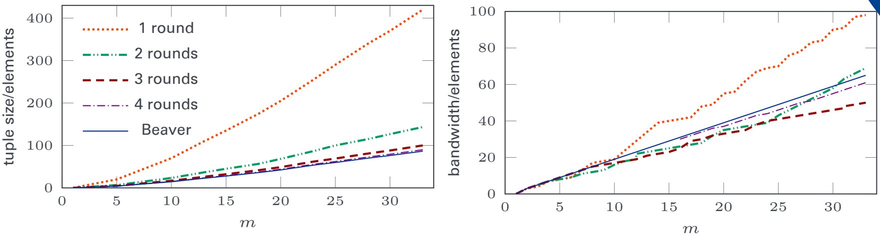
Multi-Round Use and Flexibility

- For example, we can use more communication rounds:



Multi-Round Use and Flexibility

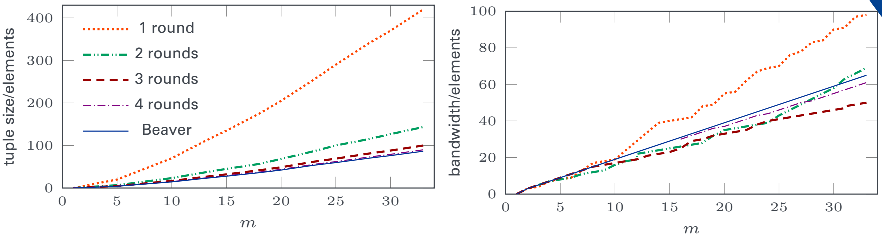
- For example, we can use more communication rounds:



- We can also use different variants of our randomized encoding to trade reduced bandwidth against larger tuple size (while keeping the same round complexity).

Multi-Round Use and Flexibility

- For example, we can use more communication rounds:



- We can also use different variants of our randomized encoding to trade reduced bandwidth against larger tuple size (while keeping the same round complexity).

⇒ Our protocols can be adapted to different setups, i.e., to bandwidth rate restrictions, network delay, or local computational power.

Implementation

- We have implemented our protocols as an extension of MP-SPDZ [Kel20], the state-of-the-art implementation of SPDZ-like protocols.

Implementation

- We have implemented our protocols as an extension of MP-SPDZ [Kel20], the state-of-the-art implementation of SPDZ-like protocols.
- Our implementation has been submitted as an artifact to Asiacrypt 2024 and has been accepted. It will be published with the paper.

Evaluation of Polynomials

- Evaluation of a Gaussian function in 32 variables with different network delays.

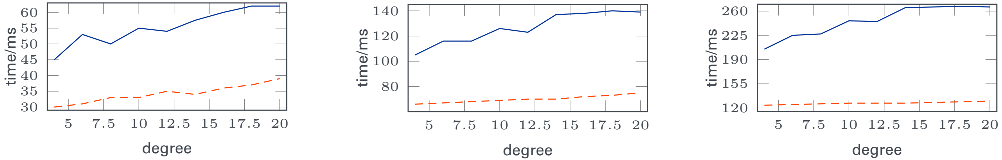


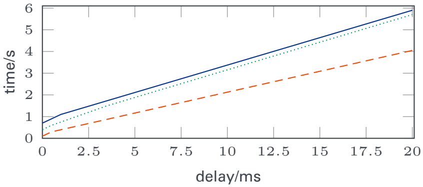
Figure: Benchmarks for Gaussian with 32 variables with 2ms (left), 5ms (middle), 10ms (right) delay; (blue: default MP-SPDZ implementation, orange/dashed: ours).

Comparisons and Rankings

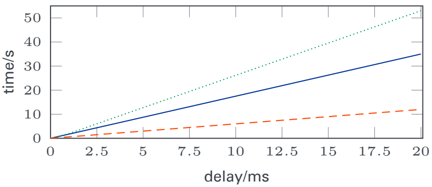
- We can also employ polytuples to (bit-wise) compare values (similar to [Dam+06]).

Comparisons and Rankings

- We can also employ polytuples to (bit-wise) compare values (similar to [Dam+06]).
- For sorting 40 items we then get:



(a) Using pairwise inequality tests.



(b) Using inequality and equality tests.

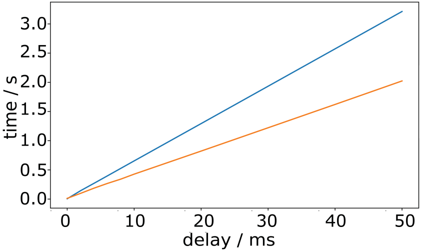
Figure: Benchmarks for sorting (blue: default MP-SPDZ implementation, orange/dashed: ours, green/dotted: MP-SPDZ with edabits [Esc+20]).

Application to Machine Learning Networks

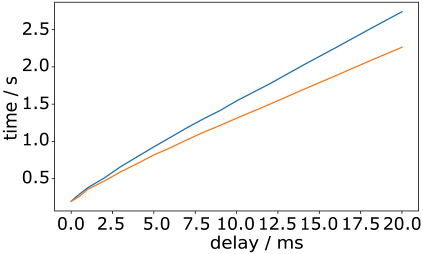
- We can use the same approach to compute comparison operations within machine learning networks.

Application to Machine Learning Networks

- We can use the same approach to compute comparison operations within machine learning networks.
- For example, if we only compute the Argmax layer (left) with our new protocol, we get for a toy ML network (right):



(a) ArgMax Layer, unlimited rate.



(b) Network A [MZ17].

Figure: Benchmarks for an ArgMax layer and the evaluation of a sample neural network included in MP-SPDZ [Kel20] as network A (cf. [Ria+18]; blue: default MP-SPDZ, orange: ours) both without bandwidth restriction.

Summary

5

Summary

- We introduce a new family of randomized encodings for the evaluation of multivariate polynomials and new correlated randomness (polytuples).

Summary

- We introduce a new family of randomized encodings for the evaluation of multivariate polynomials and new correlated randomness (polytuples).
- Our randomized encodings have the smallest known output size for arbitrary monomials.

Summary

- We introduce a new family of randomized encodings for the evaluation of multivariate polynomials and new correlated randomness (polytuples).
- Our randomized encodings have the smallest known output size for arbitrary monomials.
- We integrate the randomized encodings into a dishonest majority actively secure MPC protocol. Our approach evaluates a multivariate polynomial in just one round of online communication plus one opening round.

Summary

- We introduce a new family of randomized encodings for the evaluation of multivariate polynomials and new correlated randomness (polytuples).
- Our randomized encodings have the smallest known output size for arbitrary monomials.
- We integrate the randomized encodings into a dishonest majority actively secure MPC protocol. Our approach evaluates a multivariate polynomial in just one round of online communication plus one opening round.
- Our tuple size is significantly lower than for existing single-round approaches, and also multi-round computations yield improvements (e.g., lower bandwidth and round complexity than Beaver multiplication).

Summary

- We introduce a new family of randomized encodings for the evaluation of multivariate polynomials and new correlated randomness (polytuples).
- Our randomized encodings have the smallest known output size for arbitrary monomials.
- We integrate the randomized encodings into a dishonest majority actively secure MPC protocol. Our approach evaluates a multivariate polynomial in just one round of online communication plus one opening round.
- Our tuple size is significantly lower than for existing single-round approaches, and also multi-round computations yield improvements (e.g., lower bandwidth and round complexity than Beaver multiplication).
- We evaluated the performance of our protocols for sample applications (evaluation of polynomials, comparisons of secret-shared values, simple machine learning algorithms) and show that polytuples speed up these computations compared to Beaver multiplication.

Universität Stuttgart

Thank you!

Institute of Information Security
University of Stuttgart
Germany

Web sec.uni-stuttgart.de
Phone +49 711 685 88208

References

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. “Cryptography in NC^0”. In: *SIAM Journal on Computing* 36.4 (2006), pp. 845–888.

[CWB18] Hyunghoon Cho, David Wu, and Bonnie Berger. “Secure genome-wide association analysis using multiparty computation”. In: *Nat. Biotechnol.* 36.6 (2018), pp. 547–551.

[Che+20] Hao Chen, Miran Kim, Ilya P. Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. “Maliciously Secure Matrix Multiplication with Applications to Private Deep Learning”. In: *ASIACRYPT 2020*. Implementation: `https://github.com/snwagh/ponytail-public/`. Springer, 2020, pp. 31–59.

[Cou19] Geoffroy Couteau. “A note on the communication complexity of multiparty computation in the correlated randomness model”. In: *EUROCRYPT*. Springer. 2019, pp. 473–503.

[Cra+03] Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyial Kushilevitz. “Efficient Multi-party Computation over Rings”. In: *Advances in Cryptology — EUROCRYPT 2003*. Ed. by Eli Biham. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 596–613.

[Dam+06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. “Unconditionally Secure Constant-Rounds Multi-party Computation for Equality, Comparison, Bits and Exponentiation”. In: *TCC 2006*. Springer, 2006, pp. 285–304.

[Dam+12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. “Multiparty Computation from Somewhat Homomorphic Encryption”. In: *CRYPTO 2012*. Springer, 2012, pp. 643–662.

[Esc+20] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. “Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits”. In: *CRYPTO 2020*. Springer, 2020, pp. 823–852.

[IK00] Y. Ishai and E. Kushilevitz. “Randomizing polynomials: A new representation with applications to round-efficient secure computation”. In: *FOCS*. 2000, pp. 294–304.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. “Overdrive: Making SPDZ Great Again”. In: *EUROCRYPT 2018*. Springer, 2018, pp. 158–189.

[Kel20] Marcel Keller. “MP-SPDZ: A Versatile Framework for Multi-Party Computation”. In: *CCS ’20: 2020 ACM, Virtual Event*. ACM, 2020, pp. 1575–1590.

[MZ17] Payman Mohassel and Yupeng Zhang. “SecureML: A System for Scalable Privacy-Preserving Machine Learning”. In: *SP 2017*. IEEE Computer Society, 2017, pp. 19–38.

[Rei+23] Pascal Reisert, Marc Rivinius, Toomas Krips, and Ralf Küsters. “Overdrive LowGear 2.0: Reduced-Bandwidth MPC without Sacrifice”. In: *ACM ASIA Conference on Computer and Communications Security (ASIA CCS ’23), July 10–14, 2023, Melbourne, VIC, Australia*. 2023. URL: `https://ia.cr/2023/462`.

[Ria+18] M. Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M. Songhori, Thomas Schneider, and Farinaz Koushanfar. “Chameleon: A Hybrid Secure Computation Framework for Machine Learning Applications”. In: *AsiaCCS 2018*. ACM, 2018, pp. 707–721.

[Riv+23] Marc Rivinius, Pascal Reisert, Sebastian Hasler, and Ralf Küsters. “Convolutions in Overdrive: Maliciously Secure Convolutions for MPC”. In: *Privacy Enhancing Technologies Symposium (PETS 2023)*. 2023. URL: `https://ia.cr/2023/359`.

Acknowledgments

- This research was supported by the CRYPTECS project founded by the German Federal Ministry of Education and Research under Grant Agreement No. 16KIS1441 and by the French National Research Agency under Grant Agreement No. ANR-20-CYAL-0006 and by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart. Additionally, this research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Fundation) – 411720488. Furthermore, Toomas Krips was partly supported by the Estonian Research Council, ETAG, through grant PRG 946.