Multi-Party Function Evaluation with Perfectly Private Audit Trail

Édouard Cuvelier & Olivier Pereira

Université catholique de Louvain
ICTEAM – Crypto Group
1348 Louvain-la-Neuve – Belgium
Privacy vs Verifiability – Two Extremes

Public Auctions

Verifiability 100%
Privacy 0%

Sealed Bids Auctions

Verifiability 0%
Privacy 100%
Privacy vs Verifiability – Two Extremes

Public Auctions
- Verifiability 100%
- Privacy 0%

Sealed Bids Auctions
- Verifiability 0%
- Privacy 100%

How to conciliate Privacy and Verifiability?
Objectives

- **Generic** - Evaluate any computable functions in a multi-party setting
- **Privacy** - Parties only trust a third party for privacy
- **Verifiability** - Guarantee correctness of the result
- **Efficiency** - Run in reasonable execution-time & memory-size on standard laptop
Outline

1. Motivations
2. Protocol description
3. Three test applications
4. Conclusion
Motivations

A direct solution is the use of “Classic” Secure Multi-Party Computation...
"Classic" Secure Multi-Party Computation

Client 1
input : $x_1$

Client 2
input : $x_2$

Client 3
input : $x_3$

$f(x_1, x_2, x_3)$
Motivations I

A direct solution is the use of “Classic” Secure Multi-Party Computation...

Interesting features:

- No need of a trusted third party
- Allows to evaluate any arithmetic or boolean function [VIFF, Fairplay, Sharemind, TASTY]
- Existing implementations more and more efficient [SPDZ (Damgård et al. 13), BeDOZa (Bendlin et al. 10), TinyOT (Nielsen et al. 12)]
Motivations II

In practice, it raises issues:

- Go from 3 clients to 3333 clients?
- Online infrastructure
- Clients need to agree on the algorithm to compute the function
- Still not efficient enough to solve complex functions (NP-hard problems)
Protocol Description

\[\text{Com}(x)\] is a **commitment** on the value \(x\) (e.g. \(\text{Com}(x) = g^x h^r\)).

- \(\text{Com}(x)\) is perfectly private (information theory)
- \(\text{Com}(x)\) is computationally binding
**Protocol Description**

- $f(x_1, \ldots, x_n)$ and proof

$Com(x)$ is a **commitment** on the value $x$ (e.g. $Com(x) = g^x h^r$).

- $Com(x)$ is perfectly private (information theory)
- $Com(x)$ is computationally binding
Protocol Description

Com(x) is a **commitment** on the value x (e.g. \( Com(x) = g^x h^r \)).

- \( Com(x) \) is perfectly private (information theory)
- \( Com(x) \) is computationally binding
Advantages of the model I

- No communications between the clients
Advantages of the model II

- No communications between the clients

- The Worker can use his own sophisticated algorithms without compromising his intellectual property when the verification is not the algorithm itself
Advantages of the model II

- No communications between the clients

- The Worker can use his own sophisticated algorithms without compromising his intellectual property when the verification is not the algorithm itself

- Gain in complexity when the proof is simpler to compute than the function itself
A word on Encryption-Commitment

*Commitment Consistent Encryption* (CCEnc)

Proposed at Esorics 13 (Cuvelier, Pereira & Peters)

\[ \text{CCEnc} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{DerivCom}, \text{Open}, \text{Verify}) \]

! Ensure consistency between the commitment and the encryption !
**Efficient implementation over Elliptic Curve I**

\( G_1, G_2, G_T \) different groups of same prime order \( q \)

A bilinear map \( e : G_1 \times G_2 \rightarrow G_T \)

<table>
<thead>
<tr>
<th>( G_1 )</th>
<th>( G_2 )</th>
<th>( G_T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g )</td>
<td>( h )</td>
<td>( e(h, g) )</td>
</tr>
<tr>
<td>( g^a )</td>
<td>( h )</td>
<td>( e(g^a, h) = e(g, h)^a )</td>
</tr>
<tr>
<td>( g )</td>
<td>( h^b )</td>
<td>( e(g, h^b) = e(g, h)^b )</td>
</tr>
</tbody>
</table>

In our case: \( G_1 = E(\mathbb{F}_p) \), \( G_2 \subset E'(\mathbb{F}_{p^2}) \) and \( G_T \subset \mathbb{F}_{p^{12}} \) where \( E \) is a BN-curve, \( E' \) the twisted curve \( \sim E \)
Efficient implementation over Elliptic Curve II

small \( m \in \mathbb{Z}_q \)
additively homomorphic encryption & commitment

<table>
<thead>
<tr>
<th>( G_1 )</th>
<th>( G_2 )</th>
<th>( G_T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( g, g_1 )</td>
<td>( h, h_1 = h^{x_1} )</td>
<td></td>
</tr>
</tbody>
</table>
Efficient implementation over Elliptic Curve II

small \( m \in \mathbb{Z}_q \)
additively homomorphic encryption & commitment

\[
\begin{array}{ccc}
\mathbb{G}_1 & \mathbb{G}_2 & \mathbb{G}_T \\
g, g_1 & h, h_1 = h^{x_1} & \\
\end{array}
\]

\[d = g^r g_1^m\]
Efficient implementation over Elliptic Curve II

small \( m \in \mathbb{Z}_q \)
additively homomorphic encryption & commitment

\[
\begin{array}{c|c|c}
\mathbb{G}_1 & \mathbb{G}_2 & \mathbb{G}_T \\
g, g_1 & h, h_1 = h^{x_1} & \\
d = g^r g_1^m & c_1 = h^s & c_2 = h^r h_1^s \\
\end{array}
\]
Efficient implementation over Elliptic Curve II

small $m \in \mathbb{Z}_q$
additively homomorphic encryption & commitment

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g, g_1$</td>
<td>$h, h_1 = h^{x_1}$</td>
<td>$d = g^r g_1^m$</td>
</tr>
<tr>
<td>$d = g^r g_1^m$</td>
<td>$c_1 = h^s$</td>
<td>$c_2 = h^r h_1^s$</td>
</tr>
<tr>
<td></td>
<td>$Open_{sk}(c)$ :</td>
<td>$\text{Dec}_{sk}(c) : DLog$ of $e(g, c_1^{x_1} / c_2) \cdot e(d, h)$ $= e(g, h_1)^m$</td>
</tr>
<tr>
<td></td>
<td>$a = c_2 / c_1^{x_1}$</td>
<td>$\text{Verif}_{pk}(d, m, a) : e(g, a) \overset{?}{=} e(d / g_1^m, h)$</td>
</tr>
</tbody>
</table>
A word on the proof

The **Proof** of correctness is an aggregation of proofs on intermediate assumptions

- performed on the commitment space
- the proofs are Zero-Knowledge Proofs of Knowledge (ZKPK) that are rendered Non-Interactive
- ZKPK needed for multiplication and for range proof
- efficient in our elliptic curves based setting
A word on the proof - multiplication proof

From Damgård & Fujisaki 02:

\[ Com_1 = g^{r_1} g^{x_1}, \quad Com_2 = g^{r_2} g^{x_2}, \quad Com_3 = g^{r_3} g^{x_3} \]

we prove in NIZK that \( x_3 = x_1 x_2 \)

1. Prove the knowledge of the openings of \( Com_1, Com_2, Com_3 \)
2. Prove that \( Com_3 \) commits on the same value as \( Com_2 \) using base \( Com_1 \)

- online verification
- offline verification by using a precomputed multiplicative triplet [SPDZ]
**A word on the proof - range proof**

\[ \text{Com}(x) = g^r g_1^x \]

we prove in NIZK that \( x \in [0, L] \), \( L \leq 2^{16} \)

- needed for branching operators (\(<\) )
- based on signature-pairing (Camenish et al. 08)
  - amortized cost for small \( L \)
  - trusted setup
  - precomputation
- based binary decomposition \( L = 2^k + 1 \) and ZKPK\(_{0,1}\)
  - cost linear in \( k \)
A word on the proof - complexity

M : 1 scalar multiplication over EC
M₀ : 1 scalar multiplication over EC with precomputation ≈ 1/5M
A : 1 addition over EC
U : 1 integer in \( \mathbb{Z}_q \)

<table>
<thead>
<tr>
<th></th>
<th>Computation</th>
<th>Verification</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commitment</td>
<td>2M₀ + 1A</td>
<td>2M₀ + 1A</td>
<td>2U</td>
</tr>
<tr>
<td>ZKPK₀,₁</td>
<td>4M₀ + 2A</td>
<td>2M + 3M₀ + 3A</td>
<td>4U</td>
</tr>
<tr>
<td>ZKPKₖ₉Log</td>
<td>4M₀ + 2A</td>
<td>2M + 4M₀ + 4A</td>
<td>4U</td>
</tr>
<tr>
<td>ZKPKₖ₉consist</td>
<td>8M₀ + 3A</td>
<td>8M₀ + 3A</td>
<td>4U</td>
</tr>
<tr>
<td>ZKPKₖ₉mul</td>
<td>6M₀ + 3A</td>
<td>4M + 5M₀ + 6A</td>
<td>6U</td>
</tr>
<tr>
<td>ZKPKₖ₉ₙ₉range(2ₖ+1)</td>
<td>6kM₀ + 3kA</td>
<td>(3k − 1)M + 3kM₀ + (4k − 1)A</td>
<td>6kU</td>
</tr>
</tbody>
</table>
1st application: Auctions

The Proof consists of \( n - 1 \) NIZK Range Proofs in the commitment space \( O(n \log n) \).
1st application: Auctions

Clients → Worker

Optimal sorting \( O(n \log n) \)

\[ x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots \rightarrow x_n \]

\[ x_3 \rightarrow x_7 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{10} \]

Com\((x_1)\) Com\((x_2)\) Com\((x_3)\) \(\cdots\) Com\((x_n)\)

The Proof consists of \( n - 1 \) NIZK Range Proofs in the commitment space \( O(n) \).
1st application: Auctions

The Proof consists of \( n - 1 \) NIZK Range Proofs in the commitment space \( O(n \log n) \).
1st application: Auctions

The Proof consists of $n - 1$ NIZK Range Proofs in the commitment space $O(n)$.
2nd application: Linear System Solving

\[ S := \begin{cases} 
  a_{1,1}z_1 + \cdots + a_{1,n}z_n = b_1 \\
  \vdots \\
  a_{m,1}z_1 + \cdots + a_{m,n}z_n = b_m 
\end{cases} \iff AZ = B \]

\[ a_{i,j} = \text{Com}(x_{i,j}) \]

\[ X \in M^{m \times n}, \ A \in C^{m \times n}, \ B \in C^{m \times 1} \]

The unique solution, if one exists, is \( Z_s = X^{-1}B \ \mathcal{O}(m^3n^3) \)

The Worker publishes \( Z_s \) with the list of openings of \( AZ_s \)

The Clients verify that \( AZ_s \) open to \( B \ \mathcal{O}(mn) \)
3rd application: Shortest Path in a Graph I

Weighted graph with $E$ edges and $V$ vertices

Each Client owns one or several edges
The Clients commit on their private input and publish it. They send $\text{Enc}(x_i)$ to the Worker.

The Worker computes the shortest path using Bellman-Ford’s algorithm $O(EV)$.

The Proof of correctness is the evaluation of the algorithm on the commitments $O(EV)$.
Applications results

<table>
<thead>
<tr>
<th>test application</th>
<th>algorithm complexity</th>
<th>verif. proof complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auctions with $n$ bids</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Solving Linear System of Equations with $m$</td>
<td>$O(m^3 n^3)$</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>equations, $n$ variables</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest Path in a Graph ($E$ edges, $V$</td>
<td>$O(EV)$</td>
<td>$O(EV)$</td>
</tr>
<tr>
<td>vertices)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Worker</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auctions</td>
<td>$n = 10/100/1000$</td>
<td>$2.71e10^{-1}/2.74/27.5$</td>
</tr>
<tr>
<td>Linear System Solving</td>
<td>$n = 16/256/4096$</td>
<td>$1.43e10^{-1}/2.3/36.8$</td>
</tr>
<tr>
<td>Shortest Path</td>
<td>$E = V = 4/16/64$</td>
<td>$5.15e10^{-1}/6.87/106.58$</td>
</tr>
</tbody>
</table>

Timings in seconds that include encryption/decryption and computation/verification of the proofs.
Conclusion

A protocol that evaluates any multi-party functions

- with strong guarantee on the correctness of the result
- with perfect privacy given the trust assumption on the worker
- efficiently

We offer:

- the possibility to use optimized algorithms without compromising the intellectual property when the verification proof does not depend on the algorithm
- a gain in complexity for the client compared with the complexity of the whole algorithm (in any case the maximum bound)
- no need for a (big) network infrastructure
Future applications

The technique is especially efficient for applications where the solution is easier to verify than to compute.

In particular, **NP-hard problems** such as

- the map coloring problem
- finding an Hamiltonian cycle in a graph
- the knapsack problem