# Publicly Verifiable Covert Computation with Staged Judgement


**Abstract**

Here we provide a formal proof of security for our PVC protocol.


*Simulation for corrupted $P_2$.* We construct a simulator $\mathcal{S}$ for a corrupted $P_2$ and prove that the simulator produces a transcript indistinguishable from an adversary running the real protocol. Let $\mathcal{A}$ be a PPT malicious adversary corrupting $P_2$. We construct a simulator $\mathcal{S}$ as follows:

1. $\mathcal{S}$ acts like $P_1$ in Step 1 to Step 3.
2. In Step 4a, $\mathcal{S}$ receives $\mathcal{A}$'s inputs to $\mathcal{F}^{\Pi}_{\text{SignedOT}}$ and proceeds as follows. If $\mathcal{A}$'s input is $\mathsf{abort}_2$, then $\mathcal{S}$ sends $\mathsf{abort}_2$ to the trusted party, outputs whatever $\mathcal{A}$ outputs. Otherwise, $\mathcal{S}$ receives $(\mathbf{t}^i, \mathbf{v}^i)$ from $\mathcal{A}$, and $\mathbf{u}^i$ as well. If the checks in Step 4c pass, $\mathcal{S}$ extracts $\mathbf{r}'$ by computing $\mathbf{u}^i \oplus \mathbf{t}^i \oplus \mathbf{v}^i$. Otherwise, $\mathcal{S}$ sends $\mathsf{abort}_2$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs.
3. $\mathcal{S}$ parses $\mathbf{r}' = \mathbf{x_2} \| \boldsymbol{\phi}$ and sends $x_2$ to the trusted party, receiving back output $y_2$.
4. $\mathcal{S}$ acts as $P_1$ in Step 4d.
5. $\mathcal{S}$ chooses $\rho \leftarrow [\lambda]$. For $j \in [\lambda] \backslash \{\rho\}$, $\mathcal{S}$ acts like $P_1$ in Step 5. For $j \in \rho$, $\mathcal{S}$ computes $GC_\rho \leftarrow \mathcal{S}_{GC}(1^\kappa, y_2, \phi(C))$, where $\mathcal{S}_{GC}$ is a garbled circuit simulator, taking as input the security parameter $1^\kappa$, an output bitstring $\mathbf{y}$, and circuit leakage $\phi(C)$. $\mathcal{S}$ then acts like $P_1$ in Step 6 to 7.
6. $\mathcal{S}$ receives $\mathcal{A}$'s input $\gamma$ to $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$ and proceeds as follows:
   (a) If $\mathcal{A}$'s input is $\mathsf{abort}_2$, then $\mathcal{S}$ sends $\mathsf{abort}_2$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs.
   (b) If the input is a choice bit $\gamma$, $\mathcal{S}$ does the following. If $\gamma \neq \rho$, $\mathcal{S}$ rewinds to Step 5 above, unless $\mathcal{S}$ has rewound $\kappa\lambda$ times, in which case it halts. otherwise, $\mathcal{S}$ inputs $(\{\boldsymbol{\delta}_i, \mathbf{seed}_i\}_{i \in [\lambda] \backslash j}, \mathbf{k}^j_{\omega_i, x_1[i]_{i \in [\ell]}})$ as the $j$th input to $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$ , and then proceeds as an honest $P_1$ would.
   (c) $\mathcal{S}$ acts like $P_1$ for the rest of the protocol, and outputs whatever $\mathcal{A}$ outputs.

The proof that $\mathcal{S}$ correctly simulates a malicious $P_2$ is similar to the proof by Aumann *et al.* Aumann and Lindell (2007) and Kolesnikov *et al.* Kolesnikov and Malozemoff (2015). We just give a sketch here due to page limit.

*Non-halting detection accuracy.* We claim that $\Pi_{\mathsf{PVC}}$ is non-halting detection accurate in the sense that for an honest party $P_1$ and a fail-stop party $P_2$, the probability that $P_1$ outputs $\mathsf{corrupted}_1$ is negligible. Note that the output of the honest $P_1$ is $\mathsf{corrupted}_1$ only if it has received invalid circuits, invalid wire labels, or detected selective OT attack. However, $P_1$ must cheat actively in order that the above will happen. Due to the security of $\mathcal{F}^{\Pi}_{\text{SignedOT}}$ and $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$, the adversary corrupting $P_1$ cannot know the challenges of $P_2$. Therefore, the adversary cannot abort as a consequence of being detected cheating.

*Simulatability with $\epsilon$-deterrent for corrupted* $P_1$. Let $\mathcal{A}$ be a PPT covert adversary corrupting $P_1$. We construct a simulator $\mathcal{S}$ as follows:

1. $\mathcal{S}$ acts as $P_2$ up through Step 3;
2. In Step 4, $\mathcal{S}$ receives $\mathcal{A}$'s inputs to $\mathcal{F}^{\Pi}_{\text{SignedOT}}$ and proceeds as follows:
   (a) If $\mathcal{A}$ inputs $\mathsf{abort}_1$ in any iteration, $\mathcal{S}$ sends $\mathsf{abort}_1$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs.
   (b) Otherwise, $\mathcal{S}$ obtains $\mathbf{s}$, and acts as $P_1$ in Step 4a to 4c.
   (c) $\mathcal{S}$ receives $\mathbf{y}^i_{\omega_{\ell+j},0}$, $\mathbf{y}^i_{\omega_{\ell+j},1}$ and the signatures of them from $\mathcal{A}$. If any signature is invalid, $\mathcal{S}$ sends $\mathsf{abort}_1$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs. Otherwise, $\mathcal{S}$ obtains $\mathbf{k}^i_{\omega_{\ell+j},0}$ and $\mathbf{k}^i_{\omega_{\ell+j},1}$ by computing

$$\mathbf{k}^i_{\omega_{\ell+j},r_j} = \mathbf{y}^i_{\omega_{\ell+j},r_j} \oplus H(j, \mathbf{t}_j),$$

$$\mathbf{k}^i_{\omega_{\ell+j},1-r_j} = \mathbf{y}^i_{\omega_{\ell+j},1-r_j} \oplus H(j, \mathbf{t}_j \oplus s).$$

3. $\mathcal{S}$ acts as $P_2$ through Step 7.
4. In Step 8, $\mathcal{S}$ receives $\mathcal{A}$'s input to $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$ and proceeds as follows:
   (a) If $\mathcal{A}$'s input is $\mathsf{abort}_1$, then $\mathcal{S}$ sends to $\mathsf{abort}_1$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs.
   (b) Otherwise, $\mathcal{S}$ parses the input as $\lambda$ tuples, where the $j$th tuple is constructed as $(\{\boldsymbol{\delta}_i, \mathbf{seed}_i\}_{i\in[\lambda]\setminus\{\gamma\}}, \mathbf{k}^j_{\omega_i,x_1[i]_{i\in[n]}})$.
5. For $\gamma \in [\lambda]$, $\mathcal{S}$ sends $\gamma$ to $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$, receiving back

$$\left((\gamma, (\{\boldsymbol{\delta}_i, \mathbf{seed}_i\}_{i\in[\lambda]\setminus\{\gamma\}}, \mathbf{k}^j_{\omega_i,x_1[i]_{i\in[\ell]}}), \sigma\right).$$

   If $\sigma$ is not a valid signature, $\mathcal{S}$ aborts as an honest $P_2$ would, outputting whatever $\mathcal{A}$ outputs. Otherwise, $\mathcal{S}$ rewinds to before it sent $\gamma$ to $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$ and receives back the appropriate output. A detailed description of this process is shown in Kolesnikov and Malozemoff (2015).
6. $\mathcal{S}$ acts as $P_2$ in Steps 9 through 13.
7. $\mathcal{S}$ uses the circuit openings retrieved during the rewinding to open the circuit $GC_\gamma$ and extract $\mathcal{A}$'s input $\mathbf{x}'_1$. $\mathcal{S}$ then sends $\mathbf{x}'_1$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs.

The proof that $\mathcal{S}$ correctly simulates a covert $P_1$ follows closely to the proof by Aumann *et al.* Aumann and Lindell (2007) and Kolesnikov *et al.* Kolesnikov and Malozemoff (2015), and thus we do not repeat it here.

*Accountability.* We need to show that for every PPT adversary $\mathcal{A}$ corrupting party $P_1$ the following holds:

If $\text{OUTPUT}(\text{EXEC}_{\pi,\mathcal{A}(z),1}(x_1, x_2; 1^n)) = \mathsf{corrupted}_1$, then

$$\Pr[\mathsf{PreJudge}(Cert) = id_1] > 1 - \mu(n),$$

where *Cert* is the output certificate of $P_2$ invoking the Blame algorithm in the execution. This follows from the description of the protocol $\pi$ and the Blame, PreJudge, Appeal, Judge algorithms. There are two cases to consider.

- An adversarial $P_1$ deviates from protocol description and does not invoke the Appeal algorithm after receiving *Cert* from $P_2$. The adversary who constructs one faulty circuit must decide before the oblivious transfer in Step 8 if it wished to abort, or if it wishes to proceed. Note that due to the security of the oblivious transfer, $P_1$ cannot know what value $\gamma$ party $P_2$ inputs to $\binom{\lambda}{1}$-$\mathcal{F}^{\Pi}_{\text{SignedOT}}$ , and so cannot avoid being detected. Once the honest party outputs the certificate, it contains all the necessary information that caused the party to decide on the corruption. The verification algorithm PreJudge performs exactly the same check as the honest party, and so accountability holds.

- An adversarial $P_1$ deviates from protocol description and *invokes* the Appeal algorithm after receiving *Cert* from $P_2$. Denote the output certificate of the Appeal algorithm as $Cert^*$. In this case, the Judge algorithm is invoked with inputs *Cert* and $Cert^*$. Note that the adversary cannot provide a valid certificate in this case, and the Judge algorithm will give the same output with the PreJudge algorithm. Therefore, accountability also holds in the case where the Appeal algorithm is invoked by an adversarial $P_1$.

*Defamation-Free.* We need to show that for every PPT adversary $\mathcal{A}$ controlling $i^* \in \{1, 2\}$ and interacting with the honest party, there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large $x_1, x_2, z \in (\{0, 1\}^*)^3$:

$$\Pr[Cert^* \leftarrow \mathcal{A}; \mathsf{PreJudge}(Cert^*) = \mathsf{id}_1 \wedge \mathsf{Judge}(Cert^*, Cert) = \mathsf{id}_1] < \mu(n)$$

or

$$\Pr[Cert^* \leftarrow \mathcal{A}; \mathsf{PreJudge}(Cert) = \mathsf{id}_1 \wedge \mathsf{Judge}(Cert, Cert^*) = \mathsf{id}_2] < \mu(n)$$

There are two cases to consider.

- The first case is that an adversarial $P_2$ succeeded in defaming an honest $P_1$ in the PreJudge algorithm. This could only happen when key = SelectiveOTattack. In this case, an honest $P_1$ can obtain some evidence of $P_2$ cheating by comparing some information about $P_2$'s corrupted input matrix during signed-OT phase in *Cert* with the original correct input recovered from personal input and $P_2$'s one bit private input included in *Cert*. Therefore, by invoking the Appeal algorithm, $P_1$ can generate a valid certificate and prove that $P_2$ has cheated in the PreJudge algorithm.

- The second case is that an honest $P_2$ obtains a valid certificate in $\Pi_{\mathsf{PVC}}$, submits it to the PreJudge algorithm, and successfully caught an adversarial $P_1$. The adversary corrupting $P_1$ could invoke the Appeal algorithm when key = SelectiveOTattack, and generates a certificate $Cert^*$. This certificate, however, will not pass the checks in the Judge algorithm, unless the adversary can forge a signature. That is, if the adversary produces a certificate that passes the verification, it must have forged one of the messages.

This completes our proof. ■

## References


Aumann, Y., Lindell, Y., 2007. Security against covert adversaries: Efficient protocols for realistic adversaries, in: Theory of Cryptography Conference, Springer. pp. 137–156.

Kolesnikov, V., Malozemoff, A.J., 2015. Public verifiability in the covert model (almost) for free, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer. pp. 210–235.