We write μ_δ to denote a Bernoulli distribution, where outcome 1 occurs with probability δ and 0 with probability $1-\delta$ where $0 \leq \delta \leq 1$. We denote by μ_δ^k a probability distribution over k-tuples, where each bit of a k-tuple is drawn independently according to μ. Finally, we write $u \leftarrow \mu_\delta^k$ to denote that a k-tuple u is chosen according to μ_δ^k.

The protocol execution between two probabilistic algorithms A and B is denoted by $\langle A, B\rangle$. The output of A in such a protocol execution is denoted by $\langle A, B\rangle_A$, and of B by $\langle A, B\rangle_B$. Finally, let $\langle A, B\rangle_{\text{trans}}$ denote the transcript of communication between $\langle A, B\rangle_{\text{trans}}$.

Let $A(\delta) = (A_1, A_2)(\delta)$ denote a probabilistic two phase algorithm where in the first phase the probabilistic algorithm A_1 with randomness δ is used, and in the second phase the probabilistic algorithm A_2 with the same randomness δ.

Definition 1.1 (Dynamic weakly verifiable puzzle.) *A dynamic weakly verifiable puzzle (DWVP) is defined by a probabilistic algorithm P called a problem poser. A problem solver $S := (S_1, S_2)$ for P is a probabilistic two phase algorithm. We write $P(\pi)$ to denote the execution of P with the randomness fixed to $\pi \in \{0,1\}^n$, and $(S_1, S_2)(\rho)$ to denote the execution of both S_1 and S_2 with the randomness fixed to $\rho \in \{0,1\}^*$.*

In the first phase, the poser $P(\pi)$ and the solver $S_1(\rho)$ interact. As the result of the interaction $P(\pi)$ outputs a verification circuit Γ_V and a hint circuit Γ_H. The algorithm $S_1(\rho)$ produces no output. The circuit Γ_V takes as input $q \in Q$, an answer $y \in \{0,1\}^*$, and outputs a bit. We say an answer (q, y) is a correct solution if and only if $\Gamma_V(q, y) = 1$. The circuit Γ_H on input $q \in Q$ outputs a hint such that $\Gamma_V(q, \Gamma_H(q)) = 1$.

In the second phase, S_2 takes as input $x := \langle P(\pi), S_1(\rho)\rangle_{trans}$, and has oracle access to Γ_V and Γ_H. The execution of S_2 with the input x and the randomness fixed to ρ is denoted by $S_2(x, \rho)$. The queries of S_2 to Γ_V and Γ_H are called verification queries and hint queries respectively. The algorithm S_2 can ask at most h hint queries, v verification queries, and succeeds if and only if it makes a verification query (q, y) such that $\Gamma_V(q, y) = 1$, and it has not previously asked for a hint query on q.

Definition 1.2 (k-wise direct-product of DWVPs.) *Let $g : \{0,1\}^k \to \{0,1\}$ be a monotone function and $P^{(1)}$ a problem poser as in Definition 1.1. The k-wise direct product of $P^{(1)}$ is a DWVP defined by a probabilistic algorithm $P^{(g)}$. We write $P^{(g)}(\pi^{(k)})$ to denote the execution of $P^{(g)}$ with the randomness fixed to $\pi^{(k)} := (\pi_1, \ldots, \pi_k)$ where each $\pi_i \in \{0,1\}^n$. Let $(S_1, S_2)(\rho)$ be a solver for $P^{(g)}$ as in Definition 1.1. The algorithm $S_1(\rho)$ sequentially interacts in k rounds with $P^{(g)}(\pi^{(k)})$. In the i-th round $S_1(\rho)$ interacts with $P^{(1)}(\pi_i)$, and as the result $P^{(g)}(\pi^{(k)})$ generates circuits Γ_V^i, Γ_H^i. Finally, after k rounds $P^{(g)}(\pi^k)$ outputs a verification circuit*

$$\Gamma_V^{(g)}(q, y_1, \ldots, y_k) := g(\Gamma_V^1(q, y_1), \ldots, \Gamma_V^k(q, y_k))$$

and a hint circuit

$$\Gamma_H^{(k)}(q) := (\Gamma_H^1(q), \ldots, \Gamma_H^k(q)).$$

A verification query (q, y) of a solver S for which a hint query on this q has been asked before can not be a successful verification query. Therefore, without loss of generality, we make the assumption that S does not ask verification queries on q for which a hint query has been asked before. Furthermore, we assume that once S asked a successful verification query, it does not ask any further hint or verification queries.

Let C be a circuit that corresponds to a solver S as in Definition 1.1. Similarly as for a two phase algorithm, we write $C(\rho) := (C_1, C_2)(\rho)$ to denote that the randomness used by C is fixed to ρ, and $C(\rho)$ in the first phase uses $C_1(\rho)$ and in the second phase $C_2(\rho)$.

Experiment $Success^{P,C}(\pi, \rho)$

Oracle: A problem poser P, a solver circuit $C = (C_1, C_2)$.
Input: Bitstrings $\pi \in \{0,1\}^n$, $\rho \in \{0,1\}^*$.
Output: A bit $b \in \{0,1\}$.

Run $\langle P(\pi), C_1(\rho)\rangle$
  $(\Gamma_V, \Gamma_H) := \langle P(\pi), C_1(\rho)\rangle_P$
  $x := \langle P(\pi), C_1(\rho)\rangle_{\text{trans}}$

Run $C_2^{\Gamma_V,\Gamma_H}(x, \rho)$
  if $C_2^{\Gamma_V,\Gamma_H}$ asks a verification query (q, y) such that $\Gamma_V(q, y) = 1$ **then**
    return 1
return 0

We define *the success probability* of C in solving a puzzle defined by P as

$$\Pr_{\pi,\rho}[Success^{P,C^{(\cdot,\cdot)}}(\pi, \rho) = 1]. \tag{0.0.1}$$

Theorem 1.3 (Security amplification for a dynamic weakly verifiable puzzle.) *Let $P^{(1)}$ be a fixed problem poser as in Definition 1.1, and $P^{(g)}$ be a poser for the k-wise direct product of $P^{(1)}$. There exists a probabilistic algorithm Gen with oracle access to: a solver circuit C for $P^{(g)}$, a monotone function $g : \{0,1\}^k \to \{0,1\}$ and $P^{(1)}$. Additionally, Gen takes as input parameters ε, δ, the value n being the length of the input bitstring to $P^{(1)}$, the number of verification queries v and hint queries h asked by C, and outputs a solver circuit D for $P^{(1)}$ as in Definition 1.1 such that the following holds:*
If C is such that

$$\Pr_{\pi^{(k)},\rho}\left[Success^{P^{(g)},C}(\pi^{(k)}, \rho) = 1\right] \geq 8(h+v)\left(\Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \varepsilon\right)$$

then D satisfies almost surely

$$\Pr_{\pi,\rho}\left[Success^{P^{(1)},D}(\pi, \rho) = 1\right] \geq (\delta + \frac{\varepsilon}{6k}).$$

Additionally, D requires oracle access to g, $P^{(1)}$, C, and asks at most $\frac{6k}{\varepsilon}\log\left(\frac{6k}{\varepsilon}\right) h$ hint queries and one verification query. Finally, $Size(D) \leq Size(C) \cdot \frac{6k}{\varepsilon}$ and $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$.

Let $hash : Q \to \{0, 1, \ldots, 2(h+v) - 1\}$, then a set $P_{hash} \subseteq Q$, defined with respect to $hash$, is the set of preimages of 0 for $hash$. The idea is that P_{hash} contains $q \in Q$ on which C is not allowed to ask hint queries, and the first successful verification query (q, y) of C is such that $q \in P_{hash}$. Therefore, if C makes a verification query (q, y) such that $q \in P_{hash}$, then we know that no hint query is ever asked on this q. In the experiment *CanonicalSuccess* a circuit C succeeds if and only if it asks a successful verification query (q, y) such that $q \in P_{hash}$, and no hint query is asked on $q \in P_{hash}$.

In the experiment *CanonicalSuccess* we denote the i-th query of C by q_i if it is a hint query, and by (q_i, y_i) if it is a verification query.

Experiment $CanonicalSuccess^{P,C,hash}(\pi, \rho)$

Oracle: A problem poser P, a solver circuit $C = (C_1, C_2)$.
A function $hash : Q \to \{0, \dots, 2(h + v) - 1\}$.
Input: Bitstrings $\pi \in \{0,1\}^n$ and $\rho \in \{0,1\}^*$.
Output: A bit $b \in \{0,1\}$.

Run $\langle P(\pi), C_1(\rho)\rangle$
    $(\Gamma_V, \Gamma_H) := \langle P(\pi), C_1(\rho)\rangle_P$
    $x := \langle P(\pi), C_1(\rho)\rangle_{\text{trans}}$

Run $C_2^{\Gamma_V,\Gamma_H}(x, \rho)$
    Let (q_j, y_j) be the first verification query of C_2 such that $\Gamma_v(q_j, y_j) = 1$.
    If C_2 does not succeed let (q_j, y_j) be an arbitrary verification query.

If $(\forall i < j : hash(q_i) = 0)$ **and** $(hash(q_j) = 1)$ **and** $(\Gamma_V(q_j, y_j) = 1)$ **then**
    return 1
else
    return 0

We define *the canonical success probability* of a solver C for P with respect to a function $hash$ as

$$\Pr_{\pi,\rho}[CanonicalSuccess^{P,C,hash}(\pi, \rho) = 1]. \tag{0.0.2}$$

For fixed $hash$ and a problem poser P a *canonical success* of C for π, ρ is a situation where $CanonicalSuccess^{P,C,hash}(\pi, \rho) = 1$.

We show that if a solver circuit C for $P^{(g)}$ often succeeds in the experiment $Success$, then it is also often successful in the experiment $CanonicalSuccess$.

Lemma 1.4 ***(Success probability in solving a k-wise direct product of $P^{(1)}$ with respect to a function hash.)*** *For fixed $P^{(g)}$ let C be a solver for $P^{(g)}$ with success probability at least γ, asking at most h hint queries and v verification queries. There exists a probabilistic algorithm* ***FindHash*** *that takes as input: parameters γ, n, k, the number of verification queries v and hint queries h, and has oracle access to C and $P^{(g)}$. Furthermore,* ***FindHash*** *runs in time $O((h + v)^4/\gamma^4)$, and with high probability outputs a function hash $\in \mathcal{H}$ such that the canonical success probability of C with respect to hash is at least $\frac{\gamma}{8(h+v)}$.*

Proof. We fix $P^{(g)}$ and a solver C for $P^{(g)}$ in the whole proof of Lemma 1.4. Let $\mathcal{H}$ be a family of pairwise independent hash functions $Q \to \{0, 1, \dots, 2(h+v)-1\}$. For all $i, j \in \{1, \dots, (h+v)\}$ and $k, l \in \{0, 1, \dots, 2(h + v) - 1\}$ by the pairwise independence property of $\mathcal{H}$, we have

$$\forall q_i, q_j \in Q, q_i \neq q_j : \Pr_{hash \leftarrow \mathcal{H}}[hash(q_i) = k \mid hash(q_j) = l] = \Pr_{hash \leftarrow \mathcal{H}}[hash(q_i) = k] = \frac{1}{2(h + v)}. \tag{0.0.3}$$

Let $\pi^{(k)}, \rho$ be fixed. We choose $hash \leftarrow \mathcal{H}$ and consider the experiment $CanonicalSuccess^{P^{(g)},C,hash}(\pi^{(k)}, \rho)$. Let X be a random variable for the event $hash(q_j) = 0$ and for every query q_i asked before q_j

$hash(q_i) \neq 0$. We get

$$
\begin{aligned}
\Pr_{hash \leftarrow \mathcal{H}}[X=1] &= \Pr_{hash \leftarrow \mathcal{H}}[hash(q_j) = 0 \wedge (\forall i < j : hash(q_i) \neq 0)] \\
&= \Pr_{hash \leftarrow \mathcal{H}}[\forall i < j : hash(q_i) \neq 0 \mid hash(q_j) = 0] \Pr_{hash \leftarrow \mathcal{H}}[hash(q_j) = 0] \\
&\overset{(0.0.3)}{=} \frac{1}{2(h+v)} \left(1 - \Pr_{hash \leftarrow \mathcal{H}}[\exists i < j : hash(q_i) = 0 \mid hash(q_j) = 0] \right) \\
&\overset{(0.0.3)}{=} \frac{1}{2(h+v)} \left(1 - \Pr_{hash \leftarrow \mathcal{H}}[\exists i < j : hash(q_i) = 0] \right) \\
&\overset{(\text{u.b})}{\geq} \frac{1}{2(h+v)} \left(1 - \sum_{i<j} \Pr_{hash \leftarrow \mathcal{H}}[hash(q_i) = 0] \right) \\
&\overset{(0.0.3)}{\geq} \frac{1}{4(h+v)}.
\end{aligned}
$$

Let $\mathcal{P}_{Success}$ be the set of all $(\pi^{(k)}, \rho)$ for which $Success^{P^{(g)},C}(\pi^{(k)}, \rho)$. Furthermore, we denote the set of those $(\pi^{(k)}, \rho)$ for which $CanonicalSuccess^{P^{(g)},C(\cdot,\cdot),hash}(\pi^{(k)}, \rho) = 1$ by $\mathcal{P}_{Canonical}$. For fixed $(\pi^{(k)}, \rho)$ if C succeeds canonically, then also $Success^{P^{(g)},C}(\pi^{(k)}, \rho) = 1$. Hence, $\mathcal{P}_{Canonical} \subseteq \mathcal{P}_{Success}$, and we conclude

$$
\begin{aligned}
\Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi^{(k)}, \rho}} \left[CanonicalSuccess^{P^{(g)},C,hash}(\pi^{(k)}, \rho) = 1 \right] &= \mathop{\mathbb{E}}_{(\pi^{(k)}, \rho) \in \mathcal{P}_{Success}} \left[\Pr_{hash \leftarrow \mathcal{H}}[X = 1] \right] \\
&\geq \frac{\gamma}{4(h+v)}. \qquad (0.0.4)
\end{aligned}
$$

Algorithm: FindHash(h, v, γ, n, k)

Oracle: A problem poser $P^{(g)}$, a solver circuit C for $P^{(g)}$.
Input: Parameters h, v, γ, n, k
Output: A function $hash : Q \to \{0, 1, \ldots, 2(h+v) - 1\}$.

Let $\mathcal{H}$ be a family of pairwise independent hash functions $Q \to \{0, 1, \ldots, 2(h+v) - 1\}$
for $i = 1$ **to** $32(h+v)^2/\gamma^2$ **do:**
  $hash \overset{\$}{\leftarrow} \mathcal{H}$
  $count := 0$
  for $j := 1$ to $32(h+v)^2/\gamma^2$ **do:**
    $\pi^{(k)} \overset{\$}{\leftarrow} \{0,1\}^{kn}$
    $\rho \overset{\$}{\leftarrow} \{0,1\}^*$
    if $CanonicalSuccess^{P^{(g)},C,hash}(\pi^{(k)}, \rho) = 1$ **then**
      $count := count + 1$
  if $\frac{\gamma^2}{32(h+v)^2} count \geq \frac{\gamma}{6(h+v)}$ **then**
    return $hash$
return $\perp$

We show that **FindHash** chooses $hash$ such that the canonical success probability of C with respect to P_{hash} is at least $\frac{\gamma}{4(h+v)}$ almost surely. Let $\mathcal{H}_{Good}$ denote a family of functions $hash \in \mathcal{H}$ for which

$$
\Pr_{\pi^{(k)}, \rho} \left[CanonicalSuccess^{P^{(g)},C,hash}(\pi^{(k)}, \rho) = 1 \right] \geq \frac{\gamma}{4(h+v)}, \qquad (0.0.5)
$$

and $\mathcal{H}_{Bad}$ be the family of functions $hash \in \mathcal{H}$ such that

$$\Pr_{\pi^{(k)},\rho}\left[CanonicalSuccess^{P^{(g)},C^{(\cdot,\cdot)},hash}(\pi^{(k)},\rho)=1\right] \leq \frac{\gamma}{8(h+v)}. \tag{0.0.6}$$

Let $N := 32(h+v)^2/\gamma^2$ be the number of iterations of the inner loop of **FindHash**. For a fixed $hash$, we define binary random variables $X_1,\ldots,X_N$ such that

$$X_i = \begin{cases} 1 & \text{if in the } i\text{-th iteration of the inner loop } count \text{ is increased} \\ 0 & \text{otherwise.} \end{cases}$$

We show now that **FindHash** is unlikely to return $hash \in \mathcal{H}_{Bad}$. For $hash \in \mathcal{H}_{Bad}$ by (0.0.6) we have $\mathbb{E}_{\pi^{(k)},\rho}[X_i] \leq \frac{\gamma}{8(h+v)}$. Therefore, for any fixed $hash \in \mathcal{H}_{Bad}$ using the Chernoff bound we get

$$\Pr_{\pi^{(k)},\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \geq \frac{\gamma}{6(h+v)}\right] \leq \Pr_{\pi^{(k)},\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \geq (1+\frac{1}{3})\mathbb{E}[X_i]\right] \leq e^{-\frac{\gamma}{8(h+v)}N/27} = e^{-\frac{4}{27}\frac{(h+v)}{\gamma}}.$$

The probability that $hash \in \mathcal{H}_{Good}$, when picked, is not returned amounts

$$\Pr_{\pi^{(k)},\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \leq \frac{\gamma}{6(h+v)}\right] \leq \Pr_{\pi^{(k)},\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \leq (1-\frac{1}{3})\mathbb{E}[X_i]\right] \leq e^{-\frac{\gamma}{4(h+v)}N/18} = e^{-\frac{4}{9}\frac{(h+v)}{\gamma}}.$$

Finally, we show that **FindHash** picks in one of its iteration $hash \in \mathcal{H}_{Good}$ almost surely. Let Y_i be a binary random variable such that

$$Y_i = \begin{cases} 1 & \text{if in the } i\text{-th iteration of the outer loop } hash \in \mathcal{H}_{Good} \text{ is picked} \\ 0 & \text{otherwise.} \end{cases}$$

From equation (0.0.4) we know that $\Pr_{hash\leftarrow\mathcal{H}}[Y_i=1] = \mathbb{E}[Y_i] \geq \frac{\gamma}{4(h+v)}$, almost surely. Thus, we get

$$\Pr_{hash\leftarrow\mathcal{H}}\left[\sum_{i=1}^{K} Y_i = 0\right] \leq \left(1-\frac{\gamma}{4(h+v)}\right)^K \leq e^{-\frac{\gamma}{4(h+v)}K} = e^{-\frac{8(h+v)}{\gamma}}. \qquad \square$$

We define the following circuit $\widetilde{C}_2$:

Circuit $\widetilde{C}_2^{\Gamma_V^{(g)},\Gamma_H^{(k)},C_2,hash}(x,\rho)$

Oracle: $\Gamma_V^{(g)},\Gamma_H^{(k)},hash,C_2$
Input: A transcript x, a bitstring ρ.
Output: A tuple $(q,y_1,\ldots,y_k)$ or $\perp$.

Run $C_2^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x,\rho)$
  if C_2 asks a hint query on q **then**
    if $q \in P_{hash}$ **then**
      return $\perp$
    else
      answer the query using $\Gamma_H^{(k)}(q)$

  if C_2 asks a verification query $(q,y_1,\ldots,y_k)$ **then**
    if $q \in P_{hash}$ **then**
      ask a verification query $(q,y_1,\ldots,y_k)$
      return $(q,y_1,\ldots,y_k)$

else

answer the verification query with 0

return $\perp$

We define a new solver circuit $\widetilde{C} = (C_1, \widetilde{C}_2)$ that in the first phase uses C_1 and in the second phase $\widetilde{C}_2$. From a circuit C we can build a circuit $\widetilde{C}$ that asks at most one verification query $(q, y_1, \ldots, y_k)$ such that $q \in P_{hash}$, and every hint query on q is such that $q \notin P_{hash}$. Furthermore, we write $(q, y_1, \ldots, y_k) := \widetilde{C}_2(x, \rho)$ to denote the verification query $(q, y_1, \ldots, y_k)$ asked by $\widetilde{C}_2$. If $\widetilde{C}_2$ does not ask a verification query we write $\perp := \widetilde{C}_2(x, \rho)$.

Lemma 1.5 ***(Security amplification of a dynamic weakly verifiable puzzle with respect to P_{hash}.)*** *For fixed $P^{(1)}$ there exists an algorithm Gen, with oracle access to: $P^{(1)}$, a monotone function $g : \{0,1\}^{(k)} \to \{0,1\}$, a solver circuit C for $P^{(g)}$ and a function hash : $Q \to \{0, \ldots, 2(h+v)-1\}$. Additionally, Gen takes as input parameters ε, δ, n, the number of verification queries v and hint queries h asked by C, the number of puzzles to solve k, and outputs a solver circuit D for $P^{(1)}$ as in Definition 1.1 such that the following holds:*
If C is such that

$$\Pr_{\pi^{(k)},\rho}\left[CanonicalSuccess^{P^{(g)},C,hash}(\pi^{(k)}, \rho) = 1\right] \geq \Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \varepsilon,$$

then D satisfies almost surely

$$\Pr_{\pi,\rho}\left[CanonicalSuccess^{P^{(1)},D,hash}(\pi, \rho) = 1\right] \geq (\delta + \frac{\varepsilon}{6k}).$$

Additionally, D requires oracle access to g, $P^{(1)}$, C. Furthermore, D asks at most $\frac{6k}{\varepsilon}\log\left(\frac{6k}{\varepsilon}\right) h$ hint queries and at most one verification query. Finally, $Size(D) \leq Size(C)\frac{6k}{\varepsilon}$ and $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$.

Proof. First we define the following procedure that takes as input $b \in \{0,1\}$, and returns an estimate for $\Pr_{(u_2,\ldots,u_k)\leftarrow\mu_\delta^{k-1}}[g(b, u_2, \ldots, u_k) = 1]$.

EstimateFunctionProbability$^g(b, \varepsilon, \delta)$

Oracle: A function g.
Input: A bit $b \in \{0,1\}$, parameters k, ε
Output: An estimate $\widetilde{g} \in [0,1]$.

For $i := 1$ **to** $\frac{16k^2}{\varepsilon^2}\log(n)$ **do:**
$(u_2, \ldots, u_k) \leftarrow \mu_\delta^{(k-1)}$
$g_i := g(b, u_2, \ldots, u_k)$
return $\frac{\varepsilon^2}{16k^2\log(n)}\sum_{i=1}^{\frac{16k^2}{\varepsilon^2}\log(n)} g_i$

Lemma 1.6 (Estimate for the function g.) *The procedure **EstimateFunctionProbability**$^g(b)$ outputs an estimate $\widetilde{g}$ for the function $g : \{0,1\}^n \to \{0,1\}$ with the first bit fixed to $b \in \{0,1\}$ such that $|\widetilde{g} - \Pr_{(u_2,\ldots,u_k)\leftarrow\mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]| \leq \frac{\varepsilon}{4k}$ almost surely.*

Proof. We define a binary random variable K_i for the event $g_i = 1$. By Chernoff bound we get

$$\Pr\left[\left|\frac{\varepsilon^2 \log(n)}{16k^2} \sum_{i=1}^{\frac{16k^2}{\varepsilon^2}\log(n)} \widetilde{g}_i - \mathbb{E}[K_i]\right| \geq \frac{\varepsilon}{4k}\right] \leq 2 \cdot e^{-\log(n)/3}. \qquad \square$$

Next we define a procedure **EvalutePuzzles**$^{C,P^{(1)},hash}(\pi^{(k)}, \rho)$.

EvaluatePuzzles$^{P^{(1)},P^{(g)},C,hash}(\pi^{(k)}, \rho)$

Oracle: A circuit C, posers $P^{(1)}$, $P^{(g)}$, a function $hash$.
Input: Bitstrings $\pi^{(k)}$, ρ.
Output: A tuple $(c_1, \ldots, c_k)$.

Run $\langle P^{(g)}(\pi^{(k)}), C_1(\rho)\rangle$
  $(\Gamma_V^{(g)}, \Gamma_H^{(k)}) := \langle P(\pi^{(k)}), C_1(\rho)\rangle_{P^{(g)}}$
  $x := \langle P^{(g)}(\pi^{(k)}), C_1(\rho)\rangle_{\text{trans}}$

$(q, y_1, \ldots, y_k) := \widetilde{C}_2^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x, \rho)$
for $i := 1$ **to** k **do:**   //simulate k rounds of sequential interaction
  $(\Gamma_V^i, \Gamma_H^i) := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle_{P^{(1)}}$
for $i := 1$ **to** k **do:**
  $c_i := \Gamma_v^i(q, y_i)$
return $(c_1, \ldots, c_k)$

All puzzles used by the procedure are generated internally. Therefore, it is possible to answer all hint and verification queries without calls to hint and verification oracles. For fixed $\pi^{(k)}, \rho$ let $(\Gamma_V^{(g)}, \Gamma_H^{(k)}) := \langle P^{(g)}(\pi^{(k)}), C_1(\rho)\rangle_{P^{(g)}}$ and $x := \langle P^{(g)}(\pi^{(k)}), C_1(\rho)\rangle_{\text{trans}}$. Additionally, we denote by (Γ_V^i, Γ_H^i) the verification and hint circuits generated in the i-th round of the interaction between $P^{(g)}(\pi^{(k)})$ and $C_1(\rho)$. Finally, for $(q, y_1, \ldots, y_k) := \widetilde{C}_2(x^{(k)}, \rho)$ we denote the output of $\Gamma_V^i(q, y_i)$ by c_i. For $b \in \{0,1\}$ we define the surplus

$$S_{\pi^*,b} = \Pr_{\pi^{(k)},\rho}[g(b, c_2, \ldots, c_k) = 1 \mid \pi_1 = \pi^*] - \Pr_{(u_2,\ldots,u_k)\leftarrow\mu^{(k)}}[g(b, u_2, \ldots, u_k) = 1] \qquad (0.0.7)$$

The surplus $S_{\pi^*,b}$ tells us how good $\widetilde{C}$ performs when the bitstring π_1 is fixed to π^*, and the fact whether $\widetilde{C}$ succeeds in solving the first puzzle defined by $P^{(1)}(\pi_1)$ is neglected. Instead, the bit b is used as the input on the first position of the function g.

The procedure **EstimateSurplus** returns an estimate $\widetilde{S}_{\pi^*,b}$ for $S_{\pi^*,b}$.

EstimateSurplus$^{P^{(1)},C,hash}(\pi^*, b)$

Oracle: An algorithm $P^{(1)}$, a circuit C, a function $hash$, a function g.
Input: A bistring π^*, a bit b, an integer k.
Output: A circuit D.

$\widetilde{g}_b :=$ **EvaluteFunctionProbability**$^g(b, \varepsilon, \delta)$
For $i := 1$ to $\frac{16k^2}{\varepsilon^2}\log(n)$ **do:**
  $(\pi_2, \ldots, \pi_k) \xleftarrow{\$} \{0,1\}^{(k-1)n}$
  $\rho \xleftarrow{\$} \{0,1\}^*$

$(c_1, \ldots, c_k) :=$ **EvalutePuzzles**$^{P^{(1)},C,hash}(\pi^*, \pi_2, \ldots, \pi_k, \rho)$
$\widetilde{s}^i_{\pi^*,b} := g(b, c_2, \ldots, c_k)$
return $\frac{\varepsilon^2 \log(n)}{16k^2} \sum_{i=1}^{\frac{16k^2}{\varepsilon^2}\log(n)} \widetilde{s}^i_{\pi^*,b} - \widetilde{g}_b$

Lemma 1.7 *The estimate $\widetilde{S}_{\pi^*,b}$ returned by **EstimateSurplus** differs from $S_{\pi^*,b}$ by at most $\frac{\varepsilon}{2k}$ almost surely.*

Proof. We use union bound and similar argument as in Lemma 1.6 which yields that $\frac{\varepsilon^2 \log(n)}{16k^2} \sum_{i=1}^{\frac{16k^2}{\varepsilon^2}\log(n)} \widetilde{s}^i_{\pi^*,b}$ differs from $\mathbb{E}[g(b, c_2, \ldots, c_k)]$ by at most $\frac{\varepsilon}{4k}$ almost surely. Together, with Lemma 1.6 we conclude that the surplus estimate returned by **EstimateSurplus** differs from $S_{\pi^*,b}$ by at most $\frac{\varepsilon}{2k}$ almost surely. □

From Lemma 1.7 we conclude that if $\widetilde{S}_{\pi^*,b} \geq (1 - \frac{3}{4k})\varepsilon$, then $S_{\pi^*,b} \geq (1 - \frac{1}{k})\varepsilon$ almost surely.

Circuit $D = (D_1, D_2)(\sigma)$

Phase I $D_1^C(\sigma)$

Oracle: A circuit C.
Input: A bitstring $\sigma \in \{0,1\}^*$.

Interact with the problem poser $P^{(1)}$ using $C_1(\rho)$.
  Let x^* be the transcript of any internal simulations of C_1 and the interaction with the problem poser $P^{(1)}$.
  Let Γ^*_V, Γ^*_H be the verification and hint circuits output by the problem poser $P^{(1)}$.

Phase II $D_2^{P^{(1)},C}(x^*, \sigma)$

Oracle: $P^{(1)}$, C, $hash$, g, Γ^*_V, Γ^*_H.
Input: A transcript x^*, a bitstring $\sigma \in \{0,1\}^*$.
Output: A verification query (q, y^*).

for at most $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})$ iterations **do:**
  $\pi^{(k-1)} \leftarrow$ read $(k-1)\cdot n$ bits from σ
  for $i := 2$ **to** k **do:** // Finish remaining $k-1$ interactions.
    Simulate $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle$
      $x_i := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle_{\text{trans}}$
      $(\Gamma^i_V, \Gamma^i_H) := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle_{P^{(1)}}$
  $\Gamma^{(g)}_V := g(\Gamma^*_V, \Gamma^2_V, \ldots, \Gamma^k_V)$
  $\Gamma^{(k)}_H := (\Gamma^*_H, \Gamma^2_H, \ldots, \Gamma^k_H)$
  $(q, y^*, y_2, \ldots, y_k) := \widetilde{C}^{\Gamma^{(g)}_V, \Gamma^{(k)}_H, C, hash}((x^*, x_2, \ldots, x_k), \rho)$
  $(c^*, c_2, \ldots, c_k) := (\Gamma^*_V(q, y^*), \Gamma^2_V(q, y_2), \ldots, \Gamma^k_V(q, y_k))$
  if $g(1, c_2, \ldots, c_k) = 1 \wedge g(0, c_2, \ldots, c_k) = 0$ **then**
    Make a verification query (q, y^*)
    return (q, y^*)
return $\perp$

Algorithm $Gen^{C,P^{(1)},g,hash}(\varepsilon,\delta,n,v,h,k)$

Oracle: $P^{(1)}, C, g, hash$
Input: $\varepsilon, \delta, n, v, h, k$
Output: D

---

for $i := 1$ to $\frac{6k}{\varepsilon}\log(n)$ **do:**
  $\pi^* \xleftarrow{\$} \{0,1\}^n$
  $\widetilde{S}_{\pi^*,0} :=$ **EstimateSurplus**$^{P^{(1)},C,hash}(\pi^*, 0)$
  $\widetilde{S}_{\pi^*,1} :=$ **EstimateSurplus**$^{P^{(1)},C,hash}(\pi^*, 1)$
  if $\exists b \in \{0,1\} : \widetilde{S}_{\pi^*,b} \geq (1 - \frac{3}{4k})\varepsilon$ **then**
    Let C_1' be as C_1 except the first round of interaction between C_1 and $P^{(g)}$ which is simulated internally by using $P^{(1)}(\pi^*)$
    Let C_2' be as C_2 except the solution for the first puzzle which is discarded.
    $C' := (C_1', C_2')$
    $g'(b_2, \dots, b_k) := g(b, b_2, \dots, b_k)$
    return $Gen^{C',P^{(1)},g',hash}(\varepsilon, \delta, n, v, h, k-1)$
// all estimates are lower than $(1 - \frac{3}{4k})\varepsilon$
return D^C

For $k = 1$ the function $g : \{0,1\} \to \{0,1\}$ is either the identity or a constant function. If g is the identity function then the success probability of C in the random experiment $CanonicalSuccess$ is as least $\delta + \varepsilon$, and D simply uses the circuit $\widetilde{C}$. In case g is a constant function the statement is vacuously true.

In case Gen manages to find an estimate that satisfies $\widetilde{S}_{\pi^*,b} \geq (1 - \frac{3}{4k})\varepsilon$ we define a monotone function $g'(b_2, \dots, b_k) := g(b, b_2, \dots, b_k)$, and a circuit $\widetilde{C}' = (C_1', C_2')$, where C_1' first internally simulates the interaction between C_1 and $P^{(1)}(\pi^*)$, and then interacts with $P^{(g')}$. The circuit C_2' is defined as C_2 with the solution for the first puzzle discarded. The surplus estimate is greater than $1 - \frac{3}{4k}\varepsilon$. Therefore, the canonical success probability for the $(k-1)$-wise direct product of puzzles is at least $\Pr_{u \leftarrow \mu_\delta^{k-1}}[g'(u_1, \dots, u_{k-1})] + \varepsilon$. Hence, the circuit C' satisfies the conditions of Lemma 1.5 for $k-1$ puzzles and we recurse using g' and C'.

If all estimates are less than $(1 - \frac{3}{4k})\varepsilon$, then intuitively C does not perform much better on the remaining $k-1$ puzzles than an algorithm that solves each puzzle independently with probability δ. However, from the assumption we know that on all k puzzles $\widetilde{C}$ has higher success probability. Therefore, it is likely that the first puzzle is correctly solved with the probability higher than δ. We now show that this intuition is indeed correct. For a fixed π^* using (0.0.7), we get

$$\Pr_{u \leftarrow \mu_\delta^k}[g(1, u_2, \dots, u_k) = 1] - \Pr_{u \leftarrow \mu_\delta^k}[g(0, u_2, \dots, u_k) = 1] =$$
$$\Pr_{\pi^{(k)},\rho}[g(1, c_2, \dots, c_k) = 1 \mid \pi_1 = \pi^*] - \Pr_{\pi^{(k)},\rho}[g(0, c_2, \dots, c_k) = 1 \mid \pi_1 = \pi^*] - (S_{\pi^*,1} - S_{\pi^*,0}). \tag{0.0.8}$$

Let $\mathcal{G}_b := \{b_1, b_2, \dots, b_k : g(b, b_2, \dots, b_k) = 1\}$. From the monotonicity of g we know that $\mathcal{G}_0 \subseteq \mathcal{G}_1$. Using $\mathcal{G}_0 \subseteq \mathcal{G}_1$ and (0.0.8) we get:

$$\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0] = \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_1 \setminus \mathcal{G}_0 \mid \pi_1 = \pi^*] - (S_{\pi^*,1} - S_{\pi^*,0}). \tag{0.0.9}$$

From (0.0.9) fixing $\pi_1 = \pi^*$ we obtain

$$\Pr_{\rho}[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1] = \frac{\Pr_{\rho}[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_2]}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]} - \frac{\Pr_{\rho}[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1](S_{\pi^*,1}-S_{\pi^*,0})}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]} \tag{0.0.10}$$

We make use of the fact that the event $c\in\mathcal{G}_1\setminus\mathcal{G}_0$ implies $D(x^*,r)\neq\perp$, and write the first summand of (0.0.10) as

$$\Pr_{\rho}[CanonicalSuccess^{P(1),D,hash}(\pi^*,\rho)=1]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*] = \Pr_{\substack{\rho\\ x^*=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}}}[D_2(x^*,\rho)\neq\perp]\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*] \tag{0.0.11}$$

Now we consider two cases: if $\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\leq\frac{\varepsilon}{6k}$ then

$$\Pr_{\pi^{(k)}}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)}}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\leq\frac{\varepsilon}{6k}, \tag{0.0.12}$$

for $\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0]>\frac{\varepsilon}{6k}$ the circuit D_2 outputs $\perp$ if and only if it fails in all $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})$ iterations to find $\pi^{(k)}$ such that $g(1,c_2,\dots,c_k)=1\wedge g(0,c_2,\dots,c_k)=0$ (i.e. in none of the iterations $c\in\mathcal{G}_1\setminus\mathcal{G}_0$) which happens with probability

$$\Pr_{\substack{\rho\\ x^*:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}}}[D_2(x^*,\rho)=\perp]\leq(1-\frac{\varepsilon}{6k})^{\frac{6k}{\varepsilon}\log(\frac{\varepsilon}{6k})}\leq\frac{\varepsilon}{6k}. \tag{0.0.13}$$

We conclude that in both cases:

$$\begin{aligned}&\Pr_{\substack{\rho\\ x^*:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}}}[D_2(x^*,\rho)\neq\perp]\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\\ &\geq\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}.\end{aligned} \tag{0.0.14}$$

Therefore, we have

$$\begin{aligned}&\Pr_{\substack{\rho\\ x^*:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}}}[D_2(x^*,\rho)\neq\perp]\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\\ &\geq\Pr_{\pi^{(k)},\rho}[c_1=1\wedge c\in\mathcal{G}_0\setminus\mathcal{G}_1\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}\\ &=\Pr_{\pi^{(k)},\rho}[g(c_1,c_2,\dots,c_k)=1\wedge g(0,c_2,\dots,c_k)=0\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}\\ &=\Pr_{\pi^{(k)},\rho}[g(c_1,c_2,\dots,c_k)=1\mid\pi_1=\pi^*]-\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_0\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k},\end{aligned}$$

and finally by (0.0.7)

$$\begin{aligned}&\Pr_{\substack{\rho\\ x^*:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}}}[D_2(x^*,\rho)\neq\perp]\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\\ &=\Pr_{\pi^{(k)},\rho}[g(c_1,c_2,\dots,c_k)=1\mid\pi_1=\pi^*]-\Pr_{u\leftarrow\mu_\delta^{(k)}}[u\in\mathcal{G}_0]-S_{\pi^*,0}-\frac{\varepsilon}{6k}.\end{aligned} \tag{0.0.15}$$

Inserting this result into the equation (0.0.10) yields

$$\begin{aligned}
&\Pr_{\pi,\rho}[CanonicalSuccess^{P^{(1)},D,hash}] = \\
&\quad = \mathbb{E}_{\pi^*}\left[\frac{\Pr_{\pi^{(k)}}[g(c)=1 \mid \pi_1=\pi^*] - \Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_0] - \frac{\varepsilon}{6k}}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\right] \\
&\qquad - \mathbb{E}_{\pi^*}\left[\frac{S_{\pi^*,0} + \Pr_\rho[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1](S_{\pi^*,1}-S_{\pi^*,0})}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\right]
\end{aligned} \tag{0.0.16}$$

For the second summand we show that if we do not recurse, then almost surely majority of estimates is low. Let assume

$$\Pr_{\pi,\rho}\left[\left(S_{\pi,0}\le(1-\frac{1}{2k})\varepsilon\right)\wedge\left(S_{\pi,1}\le(1-\frac{1}{2k})\varepsilon\right)\right] < 1-\frac{\varepsilon}{6k}, \tag{0.0.17}$$

then the algorithm recurses almost surely. Therefore, under the assumption that *Gen* does not recurse, we have almost surely

$$\Pr_{\pi,\rho}\left[\left(S_{\pi,0}\le(1-\frac{1}{2k})\varepsilon\right)\wedge\left(S_{\pi,1}\le(1-\frac{1}{2k})\varepsilon\right)\right] \ge 1-\frac{\varepsilon}{6k}. \tag{0.0.18}$$

Let us define a set

$$\mathcal{W} = \left\{\pi : \left(S_{\pi,0}\le(1-\frac{1}{2k})\varepsilon\right)\wedge\left(S_{\pi,1}\le(1-\frac{1}{2k})\varepsilon\right)\right\} \tag{0.0.19}$$

and use $\mathcal{W}^c$ to denote the complement of $\mathcal{W}$. We bound the second summand in (0.0.16)

$$\begin{aligned}
&\mathbb{E}_{\pi^*}\left[S_{\pi^*,0} + \Pr_\rho[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1](S_{\pi^*,1}-S_{\pi^*,0})\right] \\
&\quad = \mathbb{E}_{\pi^*\in\mathcal{W}^c}\left[S_{\pi^*,0} + \Pr_\rho[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1](S_{\pi^*,1}-S_{\pi^*,0})\right] \\
&\qquad + \mathbb{E}_{\pi^*\in\mathcal{W}}\left[S_{\pi^*,0} + \Pr_\rho[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1](S_{\pi^*,1}-S_{\pi^*,0})\right] \\
&\quad \le \frac{\varepsilon}{6k} + \mathbb{E}_{\pi^*\in\mathcal{W}^c}\left[S_{\pi^*,0} + \Pr_\rho[CanonicalSuccess^{P^{(1)},D,hash}(\pi^*,\rho)=1]((1-\frac{1}{2k})\varepsilon-S_{\pi^*,0})\right] \\
&\quad \le \frac{\varepsilon}{6k} + 1 - \frac{\varepsilon}{2k} = 1-\frac{\varepsilon}{3k}
\end{aligned} \tag{0.0.20}$$

Finally, we insert this result into equation (0.0.16) and make use of the fact

$$\begin{aligned}
\Pr_{u\leftarrow\mu_\delta^k}[g(u)=1] &= \Pr[u\in\mathcal{G}_0 \vee (u\in\mathcal{G}_1\setminus\mathcal{G}_0 \wedge u_1=1)] \\
&= \Pr[u\in\mathcal{G}_0] + \Pr[u\in\mathcal{G}_1\setminus\mathcal{G}_0]\Pr[u_1=1]
\end{aligned}$$

which yields

$$\Pr_{\pi,\rho}[CanonicalSuccess^{P^{(1)},D,hash}] \ge \mathbb{E}_{\pi^*}\left[\frac{\Pr_{\pi^{(k)}}[g(c)=1\mid\pi_1=\pi^*] - \Pr_{u\leftarrow\mu_\delta^k}[u\in G_0] - (1-\frac{1}{6k})\varepsilon}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\right]$$

Using the assumptions of Lemma 1.5, we get

$$\begin{aligned}
\Pr_{\pi,\rho}[CanonicalSuccess^{P^{(1)},D,hash}=1] &\ge \frac{\Pr_{u\leftarrow\mu_\delta^k}[g(u)=1] + \varepsilon + \Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_0] - (1-\frac{1}{6k})\varepsilon}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]} \\
&\ge \frac{\varepsilon + \delta\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0] - (1-\frac{1}{6k})\varepsilon}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]} \ge \delta + \frac{\varepsilon}{6k}
\end{aligned} \tag{0.0.21}$$

□

Lemma 1.8 *For fixed P, C and hash the following statement is true*

$$\Pr_{\pi,\rho}[CanonicalSuccess^{P,C,hash}(\pi,\rho)=1] \le \Pr_{\pi,\rho}[CanonicalSuccess^{P,\widetilde{C},hash}(\pi,\rho)=1]$$

Proof. For some π, ρ if C succeeds canonically then also $\widetilde{C}$ succeeds canonically. Using this observation, we conclude that

$$\begin{aligned}
&\Pr_{\pi,\rho}\left[CanonicalSuccess^{P,C,hash}(\pi,\rho)=1\right] \\
&\quad = \mathop{\mathbb{E}}_{\pi,\rho}\left[CanonicalSuccess^{P,C,hash}(\pi,\rho)=1\right] \\
&\quad \le \Pr_{\pi,\rho}\left[CanonicalSuccess^{P,\widetilde{C},hash}(\pi,\rho)=1\right]
\end{aligned}$$

□

Proof (Theorem 1.3). We show that Theorem 1.3 follows by Lemmas: 1.5, 1.4, 1.8. First given a solver circuit C such that

$$\Pr_{\pi^{(k)},\rho}\left[Success^{P^{(g)},C}(\pi^{(k)},\rho)=1\right] \ge 8(h+v)\left(\Pr_{u\leftarrow\mu_\delta^k}[g(u)=1]+\varepsilon\right)$$

by Lemma 1.4 we can find a function *hash* such that

$$\Pr_{\pi^{(k)},\rho}\left[CanonicalSuccess^{P^{(g)},C,hash}(\pi^{(k)},\rho)=1\right] \ge \Pr_{u\leftarrow\mu_\delta^k}[g(u)=1]+\varepsilon.$$

By Lemma 1.8 we know that we can find a circuit $\widetilde{C}$ such that

$$\Pr_{\pi^{(k)},\rho}\left[CanonicalSuccess^{P^{(g)},\widetilde{C},hash}(\pi^{(k)},\rho)=1\right] \ge \Pr_{u\leftarrow\mu_\delta^k}[g(u)=1]+\varepsilon.$$

Finally, we apply Lemma 1.5 with the function *hash* and the circuit $\widetilde{C}$ to obtain a circuit D such that

$$\Pr_{\pi,\rho}\left[CanonicalSuccess^{P^{(1)},D,hash}(\pi,\rho)=1\right] \ge \delta+\frac{\varepsilon}{6k}.$$

If D succeeds in the experiment *CanonicalSuccess* then it also succeeds in the experiment *Succees*. □