We write μ_δ to denote the Bernoulli distribution, where outcome 1 occurs with probability δ and 0 with probability $1-\delta$ where $0 \leq \delta \leq 1$. Moreover, we use μ_δ^k to denote a probability distribution over k-tuples, where each bit of a k-tuple is drawn independently according to μ_δ. Finally, let $u \leftarrow \mu_\delta^k$ 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 $\langle A, B\rangle_{trans}$ denote the transcript of communication between A and B.

We define a *two phase circuit* $C := (C_1, C_2)$ as a circuit where in the first phase a circuit C_1 is executed and in the second phase a circuit C_2.

We say that an event happens *almost surely* or with *high probability* if it occurs with probability at least $1 - 2^{-n}poly(n)$.

Definition 1.1 (Dynamic weakly verifiable puzzle.) *A dynamic weakly verifiable puzzle (DWVP) is defined by a family of probabilistic circuits $\{P_n\}$. A circuit belonging to $\{P_n\}$ is called a problem poser. A problem solver $C := (C_1, C_2)$ for P_n is a probabilistic two phase circuit. We write $P_n(\pi)$ to denote the execution of P_n with the randomness fixed to $\pi \in \{0,1\}^n$, and $(C_1, C_2)(\rho)$ to denote the execution of both C_1 and C_2 with the randomness fixed to $\rho \in \{0,1\}^*$.*

In the first phase, the poser $P_n(\pi)$ and the solver $C_1(\rho)$ interact. As the result of the interaction $P_n(\pi)$ outputs a verification circuit Γ_V and a hint circuit Γ_H. The circuit $C_1(\rho)$ 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 that 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, C_2 takes as input $x := \langle P_n(\pi), C_1(\rho)\rangle_{trans}$, and has oracle access to Γ_V and Γ_H. The execution of C_2 with the input x and the randomness fixed to ρ is denoted by $C_2(x, \rho)$. The queries of C_2 to Γ_V and Γ_H are called verification queries and hint queries respectively. The circuit C_2 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 $P_n^{(1)}$ a problem poser as in Definition 1.1. The k-wise direct product of $P_n^{(1)}$ is a DWVP defined by a circuit $P_{kn}^{(g)}$. We write $P_{kn}^{(g)}(\pi^{(k)})$ to denote the execution of $P_{kn}^{(g)}$ with the randomness fixed to $\pi^{(k)} := (\pi_1, \ldots, \pi_k)$ where for each $1 \leq i \leq n : \pi_i \in \{0,1\}^n$. Let $(C_1, C_2)(\rho)$ be a solver for $P_{nk}^{(g)}$ as in Definition 1.1. In the first phase, the algorithm $C_1(\rho)$ sequentially interacts in k rounds with $P_{kn}^{(g)}(\pi^{(k)})$. In the i-th round $C_1(\rho)$ interacts with $P_n^{(1)}(\pi_i)$, and as the result $P_n^{(1)}(\pi_i)$ generates circuits Γ_V^i, Γ_H^i. Finally, after k rounds $P_{kn}^{(g)}(\pi^{(k)})$ 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

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

If it is clear from a context, we omit the parameter n, and write $P(\pi)$ instead of $P_n(\pi)$ where $\pi \in \{0,1\}^n$.

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

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

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

---

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

run $C_2^{\Gamma_V,\Gamma_H}(x, \rho)$
  if $C_2^{\Gamma_V,\Gamma_H}(x, \rho)$ 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

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

Furthermore, we say that C succeeds for π, ρ if $\mathit{Success}^{P,C}(\pi, \rho) = 1$.

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

$$\Pr_{\substack{\pi^{(k)}\in\{0,1\}^{kn}\\ \rho\in\{0,1\}^*}}\left[\mathit{Success}^{P_{kn}^{(g)},C}(\pi^{(k)}, \rho) = 1\right] \geq 16(h+v)\left(\Pr_{u\leftarrow\mu_\delta^k}[g(u) = 1] + \varepsilon\right)$$

then D satisfies almost surely

$$\Pr_{\substack{\pi\in\{0,1\}^{n}\\ \rho\in\{0,1\}^*}}\left[\mathit{Success}^{P_n^{(1)},D}(\pi, \rho) = 1\right] \geq (\delta + \frac{\varepsilon}{6k}).$$

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

Let $\mathit{hash} : Q \to \{0, 1, \dots, 2(h+v)-1\}$, the idea is to partition Q such that the set of preimages of 0 for 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 $\mathit{hash}(q) = 0$. Therefore, if C makes a verification query (q, y) such that $\mathit{hash}(q) = 0$, then we know that no hint query is ever asked on this q.

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. We define now an experiment *CanonicalSuccess* in which we partition Q using a function *hash*. We say that a solver circuit *succeeds* in the experiment *CanonicalSuccess* if it asks a successful verification query (q_j, y_j) such that $\mathit{hash}(q_j) = 0$, and no hint query q_i is asked before (q_j, y_j) such that $\mathit{hash}(q_i) = 0$.

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

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

---

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

run $C_2^{\Gamma_V, \Gamma_H}(x, \rho)$
  if C_2 does not succeed for any verification query **then**
    return 0
  Let (q_j, y_j) be the first verification query of C_2 such that $\Gamma_V(q_j, y_j) = 1$.

if $(\forall i < j : hash(q_i) \neq 0)$ **and** $(hash(q_j) = 0)$ **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

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

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

Let $\mathcal{H}$ be the family of pairwise independent functions $Q \to \{0, 1, \ldots, 2(h+v)-1\}$. We write $hash \leftarrow \mathcal{H}$ to denote that *hash* is chosen from $\mathcal{H}$ uniformly at random. We show that if a solver circuit C for P often succeeds in the experiment *Success*, then there exists a function $hash \in \mathcal{H}$ such that C also often succeeds in the experiment *CanonicalSuccess*.[1]

Lemma 1.4 ***(Success probability in solving DWVP with respect to a function hash.)*** *For fixed P let C be a solver for P 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, the number of verification queries v and hint queries h, and has oracle access to C and P. Furthermore, FindHash runs in time $poly(h, v, \frac{1}{\gamma}, n)$, 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 $\frac{\gamma}{16(h+v)}$.*

Proof. We fix a problem poser P and a solver C for P in the whole proof of Lemma 1.4. For all $k, l \in \{1, \ldots, (h+v)\}$ and $\alpha, \beta \in \{0, 1, \ldots, 2(h+v)-1\}$ by the pairwise independence property of $\mathcal{H}$, we have

$$\forall q_k, q_l \in Q, q_k \neq q_l : \Pr_{hash \leftarrow \mathcal{H}}[hash(q_k) = \alpha \mid hash(q_l) = \beta] = \Pr_{hash \leftarrow \mathcal{H}}[hash(q_k) = \alpha] = \frac{1}{2(h+v)}. \tag{0.0.3}$$

Let $\mathcal{P}_{Success}$ be a set containing all (π, ρ) for which $Success^{P,C}(\pi, \rho) = 1$. We fix $(\pi^*, \rho^*) \in \mathcal{P}_{Success}$, and are interested in the probability over a choice of function *hash* of the event

[1] It is possible to implement a random function *hash* efficiently by for example building its function table on the fly.

$\mathit{CanonicalSuccess}^{P,C,hash}(\pi^*, \rho^*) = 1$. Let (q_j, y_j) denote the first query such that $\Gamma_V(q_j, y_j) = 1$. We have

$$\begin{aligned}
&\Pr_{hash \leftarrow \mathcal{H}}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi^*, \rho^*) = 1] \\
&\qquad = \Pr_{hash \leftarrow \mathcal{H}}[hash(q_j) = 0 \wedge (\forall i < j : hash(q_i) \neq 0)] \\
&\qquad = \Pr_{hash \leftarrow \mathcal{H}}[\forall i < j : hash(q_i) \neq 0 \mid hash(q_j) = 0] \Pr_{hash \leftarrow \mathcal{H}}[hash(q_j) = 0] \\
&\qquad \overset{(0.0.3)}{=} \frac{1}{2(h+v)} \left(1 - \Pr_{hash \leftarrow \mathcal{H}}[\exists i < j : hash(q_i) = 0 \mid hash(q_j) = 0]\right) \\
&\qquad \overset{(*)}{\geq} \frac{1}{2(h+v)} \left(1 - \sum_{i<j} \Pr_{hash \leftarrow \mathcal{H}}[hash(q_i) = 0 | \, hash(q_j) = 0]\right) \\
&\qquad \overset{(0.0.3)}{=} \frac{1}{2(h+v)} \left(1 - \sum_{i<j} \Pr_{hash \leftarrow \mathcal{H}}[hash(q_i) = 0]\right) \\
&\qquad \overset{(0.0.3)}{\geq} \frac{1}{4(h+v)}. \qquad (0.0.4)
\end{aligned}$$

Where in $(*)$ we used the union bound. Let us denote the set of those (π, ρ) for which $\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1$ by $\mathcal{P}_{\mathit{Canonical}}$. If for π, ρ the circuit C succeeds canonically, then for the same π, ρ we also have $\mathit{Success}^{P,C}(\pi, \rho) = 1$. Hence, $\mathcal{P}_{\mathit{Canonical}} \subseteq \mathcal{P}_{\mathit{Success}}$, and we conclude

$$\begin{aligned}
&\Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi,\rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1\right] \\
&\quad = \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi,\rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \in \mathcal{P}_{\mathit{Success}}\right] \Pr_{\pi,\rho}[(\pi, \rho) \in \mathcal{P}_{\mathit{Success}}] \\
&\qquad + \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi,\rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \notin \mathcal{P}_{\mathit{Success}}\right] \Pr_{\pi,\rho}[(\pi, \rho) \notin \mathcal{P}_{\mathit{Success}}] \\
&\quad = \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi,\rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \in \mathcal{P}_{\mathit{Success}}\right] \Pr_{\pi,\rho}[(\pi, \rho) \in \mathcal{P}_{\mathit{Success}}] \\
&\quad \geq \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi,\rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \in \mathcal{P}_{\mathit{Success}}\right] \cdot \gamma \\
&\quad = \mathop{\mathbb{E}}_{(\pi,\rho) \in \mathcal{P}_{\mathit{Success}}} \left[\Pr_{hash \leftarrow \mathcal{H}}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1]\right] \cdot \gamma \\
&\quad \overset{(0.0.4)}{\geq} \frac{\gamma}{4(h+v)} \qquad (0.0.5)
\end{aligned}$$

Algorithm FindHash(γ, n, h, v)

Oracle: A problem poser P, a solver circuit C for P.
Input: Parameters γ, n. The number of h hint and v verification queries.
Output: A function $hash : Q \to \{0, 1, \ldots, 2(h+v) - 1\}$.

---

for $i := 1$ **to** $32n(h+v)^2/\gamma^2$ **do:**
  $hash \leftarrow \mathcal{H}$
  $count := 0$
  for $j := 1$ **to** $32n(h+v)^2/\gamma^2$ **do:**

$\pi \xleftarrow{\$} \{0,1\}^n$
$\rho \xleftarrow{\$} \{0,1\}^*$
if $\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho) = 1$ **then**
$\mathit{count} := \mathit{count} + 1$
if $\mathit{count} \geq \frac{\gamma}{12(h+v)} \frac{32(h+v)^2}{\gamma^2} n$ **then**
return *hash*
return $\perp$

We show that FindHash chooses $hash \in \mathcal{H}$ such that the canonical success probability of C with respect to *hash* is at least $\frac{\gamma}{16(h+v)}$ almost surely. Let $\mathcal{H}_{Good}$ denote a family of functions $hash \in \mathcal{H}$ for which

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

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

$$\Pr_{\pi,\rho}\left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho) = 1\right] \leq \frac{\gamma}{16(h+v)}. \tag{0.0.7}$$

Let N denote the number of iterations of the inner loop of FindHash. For a fixed *hash*, we define independent, identically distributed binary random variables $X_1, \ldots, X_N$ such that

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

We show now that FindHash is unlikely to return $hash \in \mathcal{H}_{Bad}$. For $hash \in \mathcal{H}_{Bad}$ by (0.0.7) we have $\mathbb{E}_{\pi,\rho}[X_i] \leq \frac{\gamma}{16(h+v)}$. Therefore, for any fixed $hash \in \mathcal{H}_{Bad}$ using the Chernoff bound we get [2]

$$\Pr_{\pi,\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \geq \frac{\gamma}{12(h+v)}\right] \leq \Pr_{\pi,\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \geq (1+\frac{1}{3})\mathbb{E}[X_i]\right] \leq e^{-\frac{\gamma}{16(h+v)}N/27} \leq e^{-\frac{2}{27}\frac{(h+v)}{\gamma}n} \leq e^{-\frac{2}{27}n}.$$

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

$$\Pr_{\pi,\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \leq \frac{\gamma}{12(h+v)}\right] \leq \Pr_{\pi,\rho}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \leq (1-\frac{1}{3})\mathbb{E}[X_i]\right] \leq e^{-\frac{\gamma}{8(h+v)}N/18} = e^{-\frac{2}{9}\frac{(h+v)}{\gamma}n} \leq e^{-\frac{2}{9}n},$$

where we once more used the Chernoff bound. We show now that the probability of picking $hash \in \mathcal{H}_{Good}$ is at least $\frac{\gamma}{8(h+v)}$. We prove this statement by contradiction. Let us assume that

$$\Pr_{hash \leftarrow \mathcal{H}}[hash \in \mathcal{H}_{Good}] < \frac{\gamma}{8(h+v)}. \tag{0.0.8}$$

[2] For independent, identically distributed binary random variables $X = \sum_{i=1}^{N} X_i$ and $0 < \delta \leq 1$ we use the Chernoff bounds in the form $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\mathbb{E}[X]\delta^2/3}$ and $\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\mathbb{E}[X]\delta^2/2}$.

We have

$$
\begin{aligned}
&\Pr_{\substack{hash\leftarrow\mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho)=1]\\
&\quad= \Pr_{\substack{hash\leftarrow\mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho)=1 \mid hash\in\mathcal{H}_{Good}]\Pr_{hash\leftarrow\mathcal{H}}[hash\in\mathcal{H}_{Good}]\\
&\qquad+ \Pr_{\substack{hash\leftarrow\mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho)=1 \mid hash\notin\mathcal{H}_{Good}]\Pr_{hash\leftarrow\mathcal{H}}[hash\notin\mathcal{H}_{Good}]\\
&\quad\le \Pr_{hash\leftarrow\mathcal{H}}[hash\in\mathcal{H}_{Good}] + \Pr_{\substack{hash\leftarrow\mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho)=1 \mid hash\notin\mathcal{H}_{Good}]\\
&\quad\overset{\substack{(0.0.6)\\(0.0.8)}}{<} \frac{\gamma}{8(h+v)}+\frac{\gamma}{8(h+v)}=\frac{\gamma}{4(h+v)},
\end{aligned}
$$

but this contradicts (0.0.5). Therefore, we know that the probability of choosing a $hash \in \mathcal{H}_{Good}$ amounts at least $\frac{\gamma}{8(h+v)}$ where the probability is taken over a choice of $hash$.

We show that FindHash picks in one of its iteration $hash \in \mathcal{H}_{Good}$ almost surely. Let K be the number of iterations of the outer loop of FindHash and Y_i be a random variable for the event that in the i-th iteration of the outer loop $hash \notin \mathcal{H}_{Good}$ is picked. We conclude using $\Pr_{hash\leftarrow\mathcal{H}}[hash\in\mathcal{H}_{Good}] \ge \frac{\gamma}{8(g+v)}$ and $K \le \frac{32(h+v)^2}{\gamma^2}n$ that

$$
\Pr_{hash\leftarrow\mathcal{H}}[\bigcap_{1\le i\le K} Y_i] \le \left(1-\frac{\gamma}{8(h+v)}\right)^{\frac{32(h+v)^2}{\gamma^2}n} \le e^{-\frac{\gamma}{8(h+v)}\frac{32(h+v)^2}{\gamma^2}n} \le e^{-\frac{4(h+v)}{\gamma}n} \le e^{-n}. \qquad \square
$$

If we want to answer hint and verification queries of C_2 by ourselves we write $C_2^{(\cdot,\cdot)}$ to stress that C_2 does not obtain direct access to hint and verification circuits.

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

Oracle: A hint circuit Γ_H, a circuit C_2,
a function $hash : Q \to \{0,1,\ldots,2(h+v)-1\}$.
Input: Bitstrings $x \in \{0,1\}^*$, $\rho \in \{0,1\}^*$.
Output: A pair (q,y).

---

run $C_2^{(\cdot,\cdot)}(x,\rho)$
  if $C_2^{(\cdot,\cdot)}(x,\rho)$ asks a hint query on q **then**
    if $hash(q)=0$ **then**
      return $\perp$
    else
      answer the query of $C_2^{(\cdot,\cdot)}(x,\rho)$ using $\Gamma_H(q)$

  if $C_2^{(\cdot,\cdot)}(x,\rho)$ asks a verification query (q,y) **then**
    if $hash(q)=0$ **then**
      return (q,y)
    else
      answer the verification query of $C_2^{(\cdot,\cdot)}(x,\rho)$ with 0
return $\perp$

Given $C=(C_1,C_2)$ we define a circuit $\widetilde{C}=(C_1,\widetilde{C}_2)$. Every hint query q asked by $\widetilde{C}$ is such that $hash(q)\neq 0$. Furthermore, $\widetilde{C}$ asks no verification queries, instead it returns $\perp$ or (q,y) such that $hash(q)=0$.

We say that for a fixed π, ρ, *hash* the circuit $\widetilde{C}$ *succeeds* if for $x := \langle P(\pi), C_1(\rho)\rangle_{\text{trans}}$, $(\Gamma_V, \Gamma_H) := \langle P(\pi), C_1(\rho)\rangle_P$, we have

$$\Gamma_V(\widetilde{C}_2^{\Gamma_H, C_2, hash}(x, \rho)) = 1.$$

Lemma 1.5 *For fixed P, C and hash the following statement is true*

$$\Pr_{\pi,\rho}[CanonicalSuccess^{P,C,hash}(\pi,\rho) = 1] \leq \Pr_{\substack{\pi,\rho \\ x:=\langle P(\pi),C_1(\rho)\rangle_{trans} \\ (\Gamma_V,\Gamma_H):=\langle P(\pi),C_1(\rho)\rangle_P}} [\Gamma_V(\widetilde{C}_2^{\Gamma_H, C_2, hash}(x,\rho)) = 1]$$

Proof. If for some fixed π, ρ and *hash* the circuit C succeeds canonically, then for the same π, ρ and *hash* also $\widetilde{C}$ succeeds. Using this observation, we conclude that

$$\begin{aligned}
&\Pr_{\pi,\rho}\left[CanonicalSuccess^{P,C,hash}(\pi,\rho) = 1\right] \\
&\quad = \mathop{\mathbb{E}}_{\pi,\rho}\left[CanonicalSuccess^{P,C,hash}(\pi,\rho) = 1\right] \\
&\quad \leq \mathop{\mathbb{E}}_{\substack{\pi,\rho \\ x:=\langle P(\pi),C_1(\rho)\rangle_{trans} \\ (\Gamma_V,\Gamma_H):=\langle P(\pi),C_1(\rho)\rangle_P}} [\Gamma_V(\widetilde{C}_2^{\Gamma_H, C_2, hash}(x,\rho)) = 1] \\
&\quad = \Pr_{\substack{\pi,\rho \\ x:=\langle P(\pi),C_1(\rho)\rangle_{trans} \\ (\Gamma_V,\Gamma_H):=\langle P(\pi),C_1(\rho)\rangle_P}} [\Gamma_V(\widetilde{C}_2^{\Gamma_H, C_2, hash}(x,\rho)) = 1]
\end{aligned}$$ □

Lemma 1.6 ***(Security amplification of a dynamic weakly verifiable puzzle with respect to hash.)*** *For fixed $P_n^{(1)}$ there exists an algorithm Gen that takes as input parameters ε,δ, n, k, has oracle access to $P_n^{(1)}$, $\widetilde{C}$, functions $hash : Q \to \{0, 1, \ldots, 2(h + v - 1)\}$, $g : \{0,1\}^k \to \{0,1\}$, and outputs a circuit $D := (D_1, D_2)$ such that the following holds:*
If $\widetilde{C} := (C_1, \widetilde{C}_2)$ has oracle access to hash and a solver circuit $C := (C_1, C_2)$ for $P_{kn}^{(g)}$, which asks at most h hint queries and v verification queries, is such that

$$\Pr_{\substack{\pi^{(k)}\in\{0,1\}^{kn},\rho\in\{0,1\}^* \\ x:=\langle P^{(g)}(\pi^{(k)}),C_1(\rho)\rangle_{trans} \\ (\Gamma_H^{(k)},\Gamma_V^{(g)}):=\langle P^{(g)}(\pi^{(k)}),C_1(\rho)\rangle_{P^{(g)}}}} [\Gamma_V^{(g)}(\widetilde{C}_2^{\Gamma_H^{(k)}, C_2, hash}(x,\rho)) = 1] \geq \Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \varepsilon,$$

then D satisfies almost surely

$$\Pr_{\substack{\pi\in\{0,1\}^n,\rho\in\{0,1\}^* \\ x:=\langle P^{(1)}(\pi),D_1^{\widetilde{C}}(\rho)\rangle_{trans} \\ (\Gamma_H,\Gamma_V):=\langle P^{(1)}(\pi),D_1^{P^{(1)},\widetilde{C}}(\rho)\rangle_{P^{(1)}}}} [\Gamma_V(D_2^{P^{(1)},\widetilde{C},hash,g,\Gamma_H}(x,\rho)) = 1] \geq (\delta + \frac{\varepsilon}{6k}).$$

Furthermore, D has oracle access to a hint circuit, $P^{(1)}$, $\widetilde{C}$, hash, g, and asks at most $\frac{6k}{\varepsilon}\log\left(\frac{6k}{\varepsilon}\right) h$ hint queries and no verification queries. Finally, $Size(D) \leq Size(C)\frac{6k}{\varepsilon}$ and $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$.

Before proving Lemma 1.6 we define additional algorithms that are later used by Gen. First, we are interested in the probability that for $u \leftarrow \mu_\delta^k$ and a bit $b \in \{0,1\}$ the function g with the first input bit set to b takes value 1. The estimate of this probability is calculated by the algorithm EstimateFunctionProbability.

Algorithm EstimateFunctionProbability$^g(b, k, \varepsilon, \delta, n)$

Oracle: A function $g : \{0,1\}^k \to \{0,1\}$.
Input: A bit $b \in \{0,1\}$, parameters k, ε, δ, n.
Output: An estimate $\widetilde{g}$ of $\Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]$.

---

for $i := 1$ **to** $\frac{64^2}{\varepsilon^2} n$ **do:**
  $u \leftarrow \mu_\delta^k$
  $g_i := g(b, u_2, \ldots, u_k)$
return $\frac{\varepsilon^2}{64^2 n} \sum_{i=1}^{\frac{64^2}{\varepsilon^2} n} g_i$

Lemma 1.7 *The algorithm EstimateFunctionProbability$^g(b, k, \varepsilon, \delta)$ outputs an estimate $\widetilde{g}$ of $\Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]$ where $b \in \{0,1\}$ such that $|\widetilde{g} - \Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]| \leq \frac{\varepsilon}{8k}$ almost surely.*

Proof. We define independent, identically distributed binary random variables $K_1, K_2, \ldots, K_{\frac{64k^2}{\varepsilon^2} n}$ such that for each $1 \leq i \leq \frac{64k^2}{\varepsilon^2} n$ the random variable K_i takes value g_i. We use the Chernoff bound to obtain [3]

$$\Pr\left[\left|\left(\frac{\varepsilon^2}{64k^2 n} \sum_{i=1}^{\frac{64k^2}{\varepsilon^2} n} K_i\right) - \mathbb{E}[K_i]\right| \geq \frac{\varepsilon}{8k}\right] \leq 2 \cdot e^{-n/3}. \qquad \square$$

The next algorithm EvalutePuzzles$^{P^{(1)}, P^{(g)}, \widetilde{C}, hash}(\pi^{(k)}, \rho, n, k)$ evaluates which of k puzzles of the k-wise direct product defined by $P^{(g)}$ are solved successfully by $\widetilde{C}(\rho) := (C_1, \widetilde{C}_2)(\rho)$. To decide whether the i-th puzzle of the k-wise direct product is solved successfully we need to gain access to the verification oracle for the puzzle generated in the i-th round of the interaction between $P^{(g)}$ and $\widetilde{C}$. Therefore, in the algorithm EvalutePuzzles, we use $P^{(1)}$, and invoke it k times to simulate the interaction with $C_1(\rho)$. Let us introduce some additional notation. We denote by $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i$ the execution of the i-th round of the simulation, and by $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{P^{(1)}}$ the output of $P^{(1)}(\pi_i)$ in the i-th round. Furthermore, we write $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{trans}$ to denote a transcript of communication in the i-th round.

To make the notation easier in the code excerpts of circuits C_2, D_2 and EvalutePuzzles we omit some oracle signatures writing for example $\widetilde{C}_2^{\Gamma_H^{(k)}, hash}$ instead of $\widetilde{C}_2^{\Gamma_H^{(k)}, C, hash}$ where the access to oracle circuit C is omitted. We make sure that it is clear from a context which circuit C is used by $\widetilde{C}_2$.

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

Oracle: Problem posers $P^{(1)}$, a solver circuit $\widetilde{C} = (C_1, \widetilde{C}_2)$ for $P^{(g)}$,
  a function $hash : Q \to \{0, 1, \ldots, 2(h + v) - 1\}$.
Input: Bitstrings $\pi^{(k)} \in \{0,1\}^{kn}$, $\rho \in \{0,1\}^*$, parameters n, k.
Output: A tuple $(c_1, \ldots, c_k) \in \{0,1\}^k$.

---

for $i := 1$ **to** k **do:**   //simulate k rounds of interaction
  $(\Gamma_V^i, \Gamma_H^i) := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{P^{(1)}}$

[3] For independent Bernoulli distributed random variables $X_1, \ldots, X_n$ with $X := \sum_{i=1}^n X_i$ and $0 \leq \delta \leq 1$ we use the Chernoff bound in the form $\Pr[|X - \mathbb{E}[X]| \geq \delta \mathbb{E}[X]] \leq 2e^{-\mathbb{E}[X]\delta^2/3}$.

$x_i := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{\text{trans}}$
$x := (x_1, \ldots, x_k)$
$\Gamma^{(k)}_H := (\Gamma^1_H, \ldots, \Gamma^k_H)$
$(q, y_1, \ldots, y_k) := \widetilde{C}_2^{\Gamma^{(k)}_H, hash}(x, \rho)$
$(c_1, \ldots, c_k) := (\Gamma^1_V(q, y_1), \ldots, \Gamma^k_V(q, y_k))$
return $(c_1, \ldots, c_k)$

All puzzles used by the algorithm EvalutePuzzles are generated internally. Thus, the algorithm can answer itself all queries of $\widetilde{C}_2$ to the hint oracle.

We are interested in the success probability of $\widetilde{C}$ with the bitstring π_1 fixed to π^* when the fact whether $\widetilde{C}$ succeeds in solving the first puzzle defined by $P^{(1)}(\pi_1)$ is neglected, and instead a bit $b \in \{0,1\}$ is used. More formally, we define the surplus $S_{\pi^*,b}$ as

$$S_{\pi^*,b} = \Pr_{\pi^{(k)},\rho}[g(b, c_2, \ldots, c_k) = 1 \mid \pi_1 = \pi^*] - \Pr_{u \leftarrow \mu^k_\delta}[g(b, u_2, \ldots, u_k) = 1]. \tag{0.0.9}$$

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

Algorithm EstimateSurplus$^{P^{(1)},\widetilde{C},g,hash}(\pi^*, b, k, \varepsilon, \delta, n)$

Oracle: A problem poser $P^{(1)}$, a circuit $\widetilde{C}$ for $P^{(g)}$, a function $g : \{0,1\}^k \to \{0,1\}$ a function $hash : Q \to \{0, 1, \ldots, 2(h+v)-1\}$.
Input: A bistring $\pi^* \in \{0,1\}^n$, a bit $b \in \{0,1\}$, parameters k, ε, δ, n.
Output: An estimate $\widetilde{S}_{\pi^*,b}$ for $S_{\pi^*,b}$.

for $i := 1$ **to** $\frac{64k^2}{\varepsilon^2}n$ **do:**
$(\pi_2, \ldots, \pi_k) \xleftarrow{\$} \{0,1\}^{(k-1)n}$
$\rho \xleftarrow{\$} \{0,1\}^*$
$(c_1, \ldots, c_k) := \text{EvalutePuzzles}^{P^{(1)},\widetilde{C},hash}((\pi^*, \pi_2, \ldots, \pi_k), \rho)$
$\widetilde{s}^i_{\pi^*,b} := g(b, c_2, \ldots, c_k)$
$\widetilde{g}_b :=$ EstimateFunctionProbability$^g(b, k, \varepsilon, \delta, n)$
return $\left(\frac{\varepsilon^2}{64k^2n}\sum_{i=1}^{\frac{64k^2}{\varepsilon^2}n} \widetilde{s}^i_{\pi^*,b}\right) - \widetilde{g}_b$

Lemma 1.8 *The estimate $\widetilde{S}_{\pi^*,b}$ returned by EstimateSurplus differs from $S_{\pi^*,b}$ by at most $\frac{\varepsilon}{4k}$ almost surely.*

Proof. We use the union bound and similar argument as in Lemma 1.7 which yields that $\frac{\varepsilon^2}{64k^2n}\sum_{i=1}^{\frac{64k^2}{\varepsilon^2}n} \widetilde{s}^i_{\pi^*,b}$ differs from $\mathbb{E}[g(b, c_2, \ldots, c_k)]$ by at most $\frac{\varepsilon}{8k}$ almost surely. Together, with Lemma 1.7 we conclude that the surplus estimate returned by EstimateSurplus differs from $S_{\pi^*,b}$ by at most $\frac{\varepsilon}{4k}$ almost surely. □

We define now the following circuit $C' = (C'_1, C'_2)$, which is a solver for a $(k-1)$-wise direct product of $P^{(1)}$.

Circuit $C_1'^{\widetilde{C},P^{(1)}}(\rho)$

Oracle: A solver circuit $\widetilde{C} = (C_1, \widetilde{C}_2)$ for $P^{(g)}$, a poser $P^{(1)}$.
Input: A bitstring $\rho \in \{0,1\}^*$

Hard-coded: A bitstring $\pi^* \in \{0,1\}^n$

Simulate $\langle P^{(1)}(\pi^*), C_1(\rho)\rangle^1$
Use $C_1(\rho)$ for the remaining $k-1$ rounds of interaction.

Circuit $\widetilde{C}_2'^{\Gamma_H^{(k-1)},\widetilde{C},hash}(x^{(k-1)},\rho)$

Oracle: A hint oracle $\Gamma_H^{(k-1)} := (\Gamma_H^2,\dots,\Gamma_H^k)$, a solver circuit $\widetilde{C} = (C_1,\widetilde{C}_2)$ for $P^{(g)}$, a function $hash : Q \to \{0,1,\dots,2(h+v)-1\}$
Input: A tuple $x^{(k-1)} := (x_2,\dots,x_k) \in \{0,1\}^*$, a bitstring $\rho \in \{0,1\}^*$
Hard-coded: A bitstring $\pi^* \in \{0,1\}^n$

Simulate $\langle P^{(1)}(\pi^*), C_1(\rho)\rangle^1$
  $(\Gamma_H^*,\Gamma_V^*) := \langle P^{(1)}(\pi^*), C_1(\rho)\rangle^1_{P^{(1)}}$
  $x^* := \langle P^{(1)}(\pi^*), C_1(\rho)\rangle^1_{trans}$
$\Gamma_H^{(k)} := (\Gamma_H^*,\Gamma_H^2,\dots,\Gamma_H^k)$
$x^{(k)} := (x^*,x_2,\dots,x_k)$
$(q,y_1,\dots,y_k) := \widetilde{C}_2^{\Gamma_H^{(k)},hash}(x^{(k)},\rho)$
return $(q,y_2,\dots,y_k)$

We are ready to define the solver circuit $D = (D_1,D_2)$ for $P^{(1)}$ and the algorithm Gen.

Circuit $D_1^{\widetilde{C}}(\rho)$

Oracle: A solver circuit $\widetilde{C} = (C_1,\widetilde{C}_2)$ for $P^{(g)}$.
Input: A tuple $\rho := (\tau,\sigma)$ where $\rho \in \{0,1\}^*$ and $\sigma \in \{0,1\}^*$.

Run the first round of interation with a problem poser $\langle P^{(1)}, C_1(\tau)\rangle^1$.
Let $x^* := \langle P^{(1)}, C_1(\rho)\rangle^1_{trans}$.

Circuit $D_2^{P^{(1)},\widetilde{C},hash,g,\Gamma_H^*}(x^*,\rho)$

Oracle: A poser $P^{(1)}$, a solver circuit $\widetilde{C} = (C_1,\widetilde{C}_2)$, functions $hash : Q \to \{0,1,\dots,2(h+v)-1\}$, $g : \{0,1\}^k \to \{0,1\}$, a hint circuit Γ_H^* for $P^{(1)}$.
Input: A communiation transcript $x^* \in \{0,1\}^*$, a bitstring $\rho := (\tau,\sigma)$ where $\tau \in \{0,1\}^*$ and $\sigma \in \{0,1\}^*$
Output: A tuple (q,y^*).

for at most $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})$ iterations **do:**
  $(\pi_2,\dots,\pi_k) \leftarrow$ read next $(k-1)\cdot n$ bits from σ
  for $i := 2$ **to** k **do:**
    run $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i$
      $(\Gamma_V^i,\Gamma_H^i) := \langle P^{(1)}(\pi_i), C_1(\tau)\rangle^i_{P^{(1)}}$
      $x_i := \langle P^{(1)}(\pi_i), C_1(\tau)\rangle^i_{trans}$
  $\Gamma_H^{(k)}(q) := (\Gamma_H^*(q),\Gamma_H^2(q),\dots,\Gamma_H^k(q))$
  $(q,y^*,y_2,\dots,y_k) := \widetilde{C}_2^{\Gamma_H^{(k)},hash}((x^*,x_2,\dots,x_k),\tau)$

$(c_2, \dots, c_k) := (\Gamma_V^2(q, y_2), \dots, \Gamma_V^k(q, y_k))$
if $g(1, c_2, \dots, c_k) = 1$ **and** $g(0, c_2, \dots, c_k) = 0$ **then**
return (q, y^*)
return $\perp$

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

Oracle: A poser $P^{(1)}$, a solver circuit $\widetilde{C}$ for $P^{(g)}$, functions $g : \{0,1\}^k \to \{0,1\}$, $hash : Q \to \{0, 1, \dots, 2(h+v) - 1\}$.
Input: Parameters ε, δ, n, k, the number of verification v and hint h queries.
Output: A circuit D.

for $i := 1$ **to** $\frac{6k}{\varepsilon} n$ **do:**
$\pi^* \xleftarrow{\$} \{0,1\}^n$
$\widetilde{S}_{\pi^*,0} := \mathrm{EstimateSurplus}^{P^{(1)},\widetilde{C},g,hash}(\pi^*, 0, k, \varepsilon, \delta, n)$
$\widetilde{S}_{\pi^*,1} := \mathrm{EstimateSurplus}^{P^{(1)},\widetilde{C},g,hash}(\pi^*, 1, k, \varepsilon, \delta, n)$
if $\exists b \in \{0,1\} : \widetilde{S}_{\pi^*,b} \geq (1 - \frac{3}{4k})\varepsilon$ **then**
Let C_1' have oracle access to $\widetilde{C}$, and have hard-coded π^*
Let $\widetilde{C}_2'$ have oracle access to $\widetilde{C}$, and have hard-coded π^*.
$\widetilde{C}' := (C_1', \widetilde{C}_2')$
$g'(b_2, \dots, b_k) := g(b, b_2, \dots, b_k)$
return $Gen^{P^{(1)},\widetilde{C}',g',hash}(\varepsilon, \delta, n, v, h, k-1)$
// all estimates are lower than $(1 - \frac{3}{4k})\varepsilon$
return $D^{P^{(1)},\widetilde{C},hash,g}$

Proof (Lemma 1.6). First let us consider the case where $k = 1$. The function $g : \{0,1\} \to \{0,1\}$ is either the identity or a constant function. If g is the identity function, then the circuit D returned by *Gen* directly uses $\widetilde{C}$ to find a solution. From the assumptions of Lemma 1.6 we know that $\widetilde{C}$ succeeds with probability at least $\delta + \varepsilon$. Hence, D trivially satisfies the statement of Lemma 1.6. If g is a constant function the statement is vacuously true.

The general case is more involved. We distinguish two possibilities. If *Gen* manages to find in one of the iterations π^* such that an estimate $\widetilde{S}_{\pi^*,b} \geq (1 - \frac{3}{4k})\varepsilon$, then we define a new monotone function $g'(b_2, \dots, b_k) := g(b, b_2, \dots, b_k)$ and a circuit $\widetilde{C}' = (C_1', \widetilde{C}_2')$ with oracle access to $\widetilde{C} := (C_1, \widetilde{C}_2)$, where C_1' first internally simulates the interaction between C_1 and $P^{(1)}(\pi^*)$, and then use C_1 to interact with $P^{(g')}$. The circuit $\widetilde{C}_2'$ uses $\widetilde{C}$ to obtain a solution $(q, y_1, \dots, y_k)$ for the k-wise direct product with π_1 fixed to π^*, and returns $(q, y_2, \dots, y_k)$. We know that one of the surplus estimates $\widetilde{S}_{\pi^*,b}$ is greater or equal $1 - \frac{3}{4k}\varepsilon$, and using Lemma 1.8 we conclude that $S_{\pi^*,b} \geq \widetilde{S}_{\pi^*,b} - \frac{\varepsilon}{4k} \geq \varepsilon - \frac{\varepsilon}{k}$ almost surely. Therefore, the circuit $\widetilde{C}'$ succeeds in solving the $(k-1)$-wise direct product of puzzles with probability at least $\Pr_{u \leftarrow \mu_\delta^{k-1}}[g'(u_1, \dots, u_{k-1})] + \varepsilon$. We see that in this case $\widetilde{C}'$ satisfies the conditions of Lemma 1.6 for the $(k-1)$-wise direct product of puzzles, and we recurse using g' and $\widetilde{C}'$.

If all estimates are less than $(1 - \frac{3}{4k})\varepsilon$, then intuitively C does not succeeds on the remaining $k-1$ puzzles with much higher probability than an algorithm that correctly solves each puzzle with probability δ. However, from the assumptions of Lemma 1.6 we know that on all k puzzles the success probability of $\widetilde{C}$ is higher. Therefore, it is likely that the first puzzle is correctly solved unusual often. It remains to prove that this intuition is indeed correct. Let

$\mathcal{G}_b := \{b_1, b_2, \ldots, b_k : g(b, b_2, \ldots, b_k) = 1\}$ then we have

$$\Pr_{u \leftarrow \mu_\delta^k}[u \in G_b] = \Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]$$
$$\Pr_{\pi^{(k)},\rho}[c \in G_b] = \Pr_{\pi^{(k)},\rho}[g(b, c_2, \ldots, c_k) = 1]. \tag{0.0.10}$$

We fix π^* and use (0.0.9), (0.0.10) to obtain

$$\Pr_{u \leftarrow \mu_\delta^k}[u \in G_1] - \Pr_{u \leftarrow \mu_\delta^k}[u \in G_0] = \Pr_{\pi^{(k)},\rho}[c \in G_1 \mid \pi_1 = \pi^*] - \Pr_{\pi^{(k)},\rho}[c \in G_0 \mid \pi_1 = \pi^*] - (S_{\pi^*,1} - S_{\pi^*,0}) \tag{0.0.11}$$

Since g is monotone, we have that $\mathcal{G}_0 \subseteq \mathcal{G}_1$, and can write (0.0.11) as

$$\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0] = \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_1 \setminus \mathcal{G}_0 \mid \pi_1 = \pi^*] - (S_{\pi^*,1} - S_{\pi^*,0}). \tag{0.0.12}$$

Still fixing $\pi_1 = \pi^*$ we multiply both sides of (0.0.12) by

$$\Pr_{\substack{\rho \\ x := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x, \rho)) = 1] / \Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0].$$

which yields

$$\begin{aligned}
&\Pr_{\substack{\rho \\ x := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x, \rho)) = 1] \\
&\quad = \Pr_{\substack{\rho \\ x := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x, \rho)) = 1] \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_1 \setminus \mathcal{G}_0 \mid \pi = \pi^*] \frac{1}{\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0]} \\
&\quad - \Pr_{\substack{\rho \\ x := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x, \rho)) = 1] (S_{\pi^*,1} - S_{\pi^*,0}) \frac{1}{\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0]}
\end{aligned} \tag{0.0.13}$$

We make use of the facts that $\Gamma_V(D(x, \rho)) = 1$ implies that $c_1 = 1$ and $D_2(x, \rho) \neq \perp$, and that the event $D_2(x^*, \rho) \neq \perp$ implies $c \in \mathcal{G}_1 \setminus \mathcal{G}_0$, which let us write the numerator of the first summand of (0.0.13) as

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

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

$$\Pr_{\pi^{(k)}}[c_1 = 1 \mid c \in \mathcal{G}_1 \setminus \mathcal{G}_0, \pi_1 = \pi^*] \Pr_{\pi^{(k)}}[c \in \mathcal{G}_1 \setminus \mathcal{G}_0 \mid \pi_1 = \pi^*] \leq \frac{\varepsilon}{6k}, \tag{0.0.15}$$

for $\Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_1 \setminus \mathcal{G}_0 \mid \pi_1 = \pi^*] > \frac{\varepsilon}{6k}$ 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 $g(1, c_2, \ldots, c_k) = 1 \wedge g(0, c_2, \ldots, c_k) = 0$ (i.e. in none of the iterations $c \in \mathcal{G}_1 \setminus \mathcal{G}_0$) which happens with probability

$$\Pr_{\substack{\rho \\ x := \langle P^{(1)}(\pi), D_1(\rho)\rangle_{\text{trans}}}}[D_2(x, \rho) = \perp] \leq (1 - \frac{\varepsilon}{6k})^{\frac{6k}{\varepsilon}\log(\frac{\varepsilon}{6k})} \leq \frac{\varepsilon}{6k}. \tag{0.0.16}$$

We conclude that in both cases:

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

$$\begin{aligned}
&\geq \Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}\\
&=\Pr_{\pi^{(k)},\rho}[c_1=1\wedge c\in\mathcal{G}_0\setminus\mathcal{G}_1\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}\\
&=\Pr_{\pi^{(k)},\rho}[g(c_1,c_2,\ldots,c_k)=1\wedge g(0,c_2,\ldots,c_k)=0\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}\\
&=\Pr_{\pi^{(k)},\rho}[g(c)=1\mid\pi_1=\pi^*]-\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_0\mid\pi_1=\pi^*]-\frac{\varepsilon}{6k}\\
&\overset{(0.0.9)}{=}\Pr_{\pi^{(k)},\rho}[g(c_1,c_2,\ldots,c_k)=1\mid\pi_1=\pi^*]-\Pr_{u\leftarrow\mu_\delta^{(k)}}[u\in\mathcal{G}_0]-S_{\pi^*,0}-\frac{\varepsilon}{6k}.
\end{aligned}\tag{0.0.17}$$

For the second summand of (0.0.13) we show that if we do not recurse, then the majority of the estimates is low almost surely. Let us assume that

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

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

$$\Pr_{\pi,\rho}\left[\left(S_{\pi,0}\leq(1-\frac{1}{2k})\varepsilon\right)\wedge\left(S_{\pi,1}\leq(1-\frac{1}{2k})\varepsilon\right)\right]\geq1-\frac{\varepsilon}{6k}.\tag{0.0.19}$$

Let us define a set

$$\mathcal{W}=\left\{\pi:\left(S_{\pi,0}\leq(1-\frac{1}{2k})\varepsilon\right)\wedge\left(S_{\pi,1}\leq(1-\frac{1}{2k})\varepsilon\right)\right\}\tag{0.0.20}$$

and use $\mathcal{W}^c$ to denote the complement of $\mathcal{W}$. We bound the second summand in (0.0.13)

$$\begin{aligned}
&\mathbb{E}_{\pi^*}[S_{\pi^*,0}+\Pr_{\substack{\rho\\x:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}\\(\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,\rho))=1](S_{\pi^*,1}-S_{\pi^*,0})]\\
&=\mathbb{E}_{\pi^*\in\mathcal{W}^c}[S_{\pi^*,0}+\Pr_{\substack{\rho\\x:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}\\(\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,\rho))=1](S_{\pi^*,1}-S_{\pi^*,0})]\\
&\quad+\mathbb{E}_{\pi^*\in\mathcal{W}}[S_{\pi^*,0}+\Pr_{\substack{\rho\\x:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}\\(\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,\rho))=1](S_{\pi^*,1}-S_{\pi^*,0})]\\
&\leq\frac{\varepsilon}{6k}+\mathbb{E}_{\pi^*\in\mathcal{W}^c}[S_{\pi^*,0}+\Pr_{\substack{\rho\\x:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}}\\(\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2^{\widetilde{C}}(x,\rho))=1]((1-\frac{1}{2k})\varepsilon-S_{\pi^*,0})]\\
&\leq\frac{\varepsilon}{6k}+1-\frac{\varepsilon}{2k}=1-\frac{\varepsilon}{3k}.
\end{aligned}\tag{0.0.21}$$

We observe that

$$\begin{aligned}
\Pr_{u\leftarrow\mu_\delta^k}[g(u)=1]&=\Pr[u\in\mathcal{G}_0\vee(u\in\mathcal{G}_1\setminus\mathcal{G}_0\wedge u_1=1)]\\
&=\Pr[u\in\mathcal{G}_0]+\Pr[u\in\mathcal{G}_1\setminus\mathcal{G}_0]\Pr[u_1=1].
\end{aligned}\tag{0.0.22}$$

Finally, we insert (0.0.17) and (0.0.21) into equation (0.0.13) and use (0.0.22) to obtain

$$\Pr_{\substack{\rho \\ x:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,\rho))=1] \geq \mathbb{E}_{\pi^*}\left[\frac{\Pr_{\pi^{(k)},\rho}[g(c)=1 \mid \pi_1=\pi^*] - \Pr_{u\leftarrow\mu_\delta^k}[u\in G_0] - (1-\frac{1}{6k})\varepsilon}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\right].$$

From the assumptions of Lemma 1.6 we know that $\Pr_{\pi^{(k)},\rho}[g(c)=1] \geq \Pr_{u\leftarrow\mu_\delta^{(k)}}[g(u)=1]$, thus we get

$$\Pr_{\substack{\rho \\ x:=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,\rho))=1] \geq \frac{\Pr_{u\leftarrow\mu_\delta^k}[g(u)=1]+\varepsilon+\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_0]-(1-\frac{1}{6k})\varepsilon}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}$$

$$\geq \frac{\varepsilon+\delta\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]-(1-\frac{1}{6k})\varepsilon}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]} \geq \delta+\frac{\varepsilon}{6k} \tag{0.0.23}$$

□

Proof (Theorem 1.3). Let use first define following circuits.

Circuit $\widetilde{D}_2^{D,P^{(1)},hash,g,\Gamma_v}(x,\rho)$

Oracle: A circuit D for $P^{(1)}$, a problem poser $P^{(1)}$,
function $hash : Q \to \{0,1,\ldots,2(h+v)-1\}$, $g:\{0,1\}^k \to \{0,1\}$
a verification oracle Γ_V, a hint oracle Γ_H
Input: Bitstrings $x\in\{0,1\}^*$, $\rho\in\{0,1\}^*$

$(q,y) := D_2^{P^{(1)},P^{(g)},\widetilde{C},hash,g,\Gamma_H}(x,\rho)$
Make a verification query to Γ_V using (q,y)

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

Oracle: A problem poser $P^{(1)}$, a function $g:\{0,1\}^k\to\{0,1\}$,
a solver circuit C for $P^{(g)}$.
Input: Parameters n, ε, δ, k, h, v.

$hash := \text{FindHash}((h+v)\varepsilon,n,h,v)$
Let $\widetilde{C} := (C_1,\widetilde{C}_2)$ be as in Lemma 1.5 with oracle access to C, $hash$.
$D := Gen^{P^{(1)},P^{(g)},\widetilde{C},hash,g}(\varepsilon,\delta,n,k)$
return $\widetilde{D} := (D_1,\widetilde{D}_2)$

We show that Theorem 1.3 follows from Lemma 1.4 and Lemma 1.6. We use the algorithm $\widetilde{Gen}$ to obtain a circuit $D := (D_1,D_2)$. Then, the circuit D uses $\widetilde{D}$ to find (q,y). We will show that with high probability it holds

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

We fix $P^{(1)}$, g, $P^{(g)}$. Given a solver circuit $C = (C_1, C_2)$ such that

$$\Pr_{\pi^{(k)},\rho}\left[Success^{P^{(g)},C}(\pi^{(k)},\rho) = 1\right] \geq 16(h+v)\left(\Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \varepsilon\right)$$

we satisfy conditions of Lemma 1.4 and can use the algorithm FindHash to find a function *hash* such that

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

almost surely. By Lemma 1.5 we know that it is possible to create a circuit $\widetilde{C} = (C_1, \widetilde{C}_2)$ with oracle access to *hash* and C such that

$$\Pr_{\substack{\pi,\rho \\ x := \langle P(\pi), C_1(\rho)\rangle_{trans} \\ (\Gamma_V^{(g)}, \Gamma_H^{(k)}) := \langle P(\pi), C_1(\rho)\rangle_P}}[\Gamma_V^{(g)}(\widetilde{C}_2^{\Gamma_H^{(k)}, C_2, hash}(x,\rho)) = 1] \geq \Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \varepsilon.$$

Now, by Lemma 1.6 using *hash* and $\widetilde{C} = (C_1, \widetilde{C}_2)$ we use the algorithm Gen to obtain a circuit $D = (D_1, D_2)$ such that

$$\Pr_{\substack{\pi,\rho \\ x := \langle P^{(1)}(\pi), D_1(\rho)\rangle_{trans} \\ (\Gamma_H, \Gamma_V) := \langle P^{(1)}(\pi), D_1(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,\rho)) = 1] \geq (\delta + \frac{\varepsilon}{6k}) \tag{0.0.24}$$

almost surely. Hence, the circuit $\widetilde{D}$ output by $\widetilde{\text{Gen}}$ satisfies with high probability

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

This ends the proof of Theorem 1.3. □