

**Definition 1.1** *Dynamic weakly verifiable puzzle (non interactive version)*

A dynamic weakly verifiable puzzle (DWVP) is defined by a probabilistic algorithm  $P(\pi)$ , called a problem poser, that takes as input chosen uniformly at random bitstring  $\pi \in \{0,1\}^l$ , and produces circuits  $\Gamma_V$ ,  $\Gamma_H$  and a puzzle  $x \in \{0,1\}^*$ . The circuit  $\Gamma_V$  takes as its input  $q \in Q$  and an answer  $y$ . If  $\Gamma_V(q, y) = 1$  then  $y$  is a correct solution of puzzle  $x$  for  $q$ . The circuit  $\Gamma_H$  on input  $q$  provides a hint such that  $\Gamma_V(q, \Gamma_H(q)) = 1$ . The algorithm  $S$ , called a solver, has oracle access to  $\Gamma_V$  and  $\Gamma_H$ . The calls of  $S$  to  $\Gamma_V$  are called verification queries and the calls to  $\Gamma_H$  are hint queries. The solver  $S$  can ask at most  $h$  hint queries,  $v$  verification queries, and successfully solves a DWVP if and only if it makes a verification query  $(q, y)$  such that  $\Gamma_V(q, y) = 1$ , when it has not previously asked for a hint query on this  $q$ .

**Definition 1.2** *k-wise direct product of dynamic weakly verifiable puzzles*

Let  $g : \{0,1\}^k \rightarrow \{0,1\}$  be a monotone function, and  $P^{(1)}$  a probabilistic algorithm used to generate an instance of DWVP. A  $k$ -wise direct product of dynamic weakly verifiable puzzles is defined by a probabilistic algorithm  $P^{(g)}(\pi_1, \dots, \pi_k)$ , where  $(\pi_1, \dots, \pi_k) \in \{0,1\}^{k \cdot l}$  are chosen uniformly at random.  $P^{(g)}(\pi_1, \dots, \pi_k)$  sequentially generates  $k$  independent instances of dynamic weakly verifiable puzzles, where in the  $i$ -th round  $P^{(g)}$  runs  $P^{(1)}(\pi_i)$  and obtains  $(x_i, \Gamma_V^{(i)}, \Gamma_H^{(i)})$ . Finally,  $P^{(g)}$  outputs a verification circuit

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

a hint circuit

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

and a puzzle  $x^{(k)} := (x_1, \dots, x_k)$ .

The probabilistic algorithm  $S$ , called a solver, has oracle access to  $\Gamma_V^{(g)}, \Gamma_H^{(k)}$ . The solver  $S$  can ask at most  $v$  verification queries to  $\Gamma_V^{(g)}$ ,  $h$  hint queries to  $\Gamma_H^{(k)}$  and successfully solves the puzzle  $x^{(k)}$  if and only if it asks a verification query  $(q, y_1, \dots, y_k)$  such that  $\Gamma_V^{(g)}(q, y_1, \dots, y_k) = 1$ , and it has not previously asked for a hint query on this  $q$ .

**Experiment**  $A^{P^{(\cdot)}, C^{(\cdot, \cdot)}}(\pi^{(\cdot)})$

**Oracle:** A problem poser  $P^{(\cdot)}$  and a solver circuit  $D^{(\cdot, \cdot)}$ .

**Input:** A bitstring  $\pi^{(\cdot)}$ .

$(x^{(\cdot)}, \Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)}) := P^{(\cdot)}(\pi^{(\cdot)})$

Run  $D^{(\cdot, \cdot)}(\Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)})(x^{(\cdot)})$

$Q_{Solved} := \{q : D^{(\cdot, \cdot)}(\Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)})(x^{(\cdot)}) \text{ asked a verification query } (q, y^{(\cdot)}) \text{ and } \Gamma_V^{(\cdot)}(q, y^{(\cdot)}) = 1\}$

$Q_{Hint} := \{q : D^{(\cdot, \cdot)}(\Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)})(x^{(\cdot)}) \text{ asked a hint query on } q\}$

**If**  $\exists q \in Q_{Solved} : q \notin Q_{Hint}$

**return** 1

**else**

**return** 0

**Theorem 1.3** *Security amplification of a dynamic weakly verifiable puzzle.*

For a fixed problem poser  $P^{(1)}$  there exists an algorithm  $Gen(C, g, \varepsilon, \delta, n, v, h)$  which takes as input a solver circuit  $C$  for  $k$ -wise direct product of DWVP, a monotone function  $g$ , parameters

$\varepsilon, \delta, n$ , the number of verification  $v$ , and hint  $h$  queries asked by  $C$ , and outputs a circuit  $D$  such that following holds:

If  $C$  is such that

$$\Pr_{(\pi_1, \dots, \pi_k) \in \{0,1\}^{kl}} [A^{P^{(g)}, C}(\pi_1, \dots, \pi_k) = 1] \geq \Pr_{\mu \leftarrow \mu_\delta^k} [g(\mu) = 1] + \varepsilon$$

then  $D$  satisfies almost surely

$$\Pr_{\pi \in \{0,1\}^l} [A^{P^{(1)}, D}(\pi) = 1] \geq (\delta + \frac{\varepsilon}{6k})$$

Additionally,  $D$  and  $Gen$  require only oracle access to  $g$  and  $C$ . Furthermore,  $D$  asks at most  $h$  hint queries,  $v$  verification queries and  $Size(D) \leq Size(C) \frac{6k}{\varepsilon}$  and  $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$ .

Let  $hash : Q \rightarrow \{0, 1, \dots, 2(h+v) - 1\}$  and  $P_{hash}$ , defined with respect to  $hash$ , is a preimage of 0 for function  $hash$ .

**Lemma 1.4 Success probability with respect to hash function.**

For a fixed  $P^{(g)}$  let  $C$  succeed in solving the  $k$ -wise direct product of DWVP produced by  $P^{(g)}$  with probability  $\gamma$  making  $h$  hint and  $v$  verification queries. There exists a probabilistic algorithm, with oracle access to  $C$ , that runs in time  $O((h+v)^4/\gamma^4)$  and with high probability outputs a function  $hash : Q \rightarrow \{0, \dots, 2(h+v) - 1\}$  such that success probability of  $C$  in random experiment  $E$  with respect to the set  $P_{hash}$  is at least  $\frac{\gamma}{8(h+v)}$ .

**Proof** Let  $\mathcal{H}$  be a family of pairwise independent hash functions  $Q \rightarrow \{0, 1, \dots, 2(h+v) - 1\}$ . By a pairwise independence property of  $\mathcal{H}$  we know that for all  $i \neq j \in \{1, \dots, (h+v)\}$  and  $k, l \in \{0, 1, \dots, 2(h+v) - 1\}$  we have the following

$$\forall q_i, q_j \in Q : \Pr_{hash \leftarrow \mathcal{H}} [hash(q_i) = k \mid hash(q_j) = l] = \Pr_{hash \leftarrow \mathcal{H}} [hash(q_i) = k] = \frac{1}{2(h+v)}. \quad (0.0.1)$$

For a fixed  $P^{(g)}$  and  $(\pi_1, \dots, \pi_k)$  in the random experiment  $A$  we define a binary random variable  $X$  for the event that  $hash(q_j) = 0$ , and for every query  $q_i$  asked before  $q_j$   $hash(q_i) \neq 0$ . By definition of conditional probability

$$\begin{aligned} \Pr_{hash \leftarrow \mathcal{H}} [X = 1] &= \Pr_{hash \leftarrow \mathcal{H}} [hash(q_j) = 0 \wedge \forall i < j : hash(q_i) \neq 0] \\ &= \Pr_{hash \leftarrow \mathcal{H}} [\forall i < j : hash(q_i) \neq 0 \mid hash(q_j) = 0] \Pr_{hash \leftarrow \mathcal{H}} [hash(q_j) = 0]. \end{aligned}$$

Now we use (0.0.1) and obtain

$$\Pr_{hash \leftarrow \mathcal{H}} [X = 1] = \frac{1}{2(h+v)} \left( 1 - \Pr_{hash \leftarrow \mathcal{H}} [\exists i < j : hash(q_i) = 0 \mid hash(q_j) = 0] \right)$$

Using pairwise independence property we conclude

$$\Pr_{hash \leftarrow \mathcal{H}} [X = 1] = \frac{1}{2(h+v)} \left( 1 - \Pr_{hash \leftarrow \mathcal{H}} [\exists i < j : hash(q_i) = 0] \right).$$

Finally, we use union bound and the fact  $j \leq (h+v)$  to get

$$\Pr_{hash \leftarrow \mathcal{H}} [X = 1] \geq \frac{1}{2(h+v)} \left( 1 - \sum_{i < j} \Pr_{hash \leftarrow \mathcal{H}} [hash(q_i) = 0] \right) \geq \frac{1}{4(h+v)}$$

Let  $G$  denote the set of all  $(\pi_1, \dots, \pi_k)$  for which  $C$  succeeds in the random experiment  $A$ . Then

$$\begin{aligned} \Pr_{\substack{hash \leftarrow \mathcal{H} \\ (\pi_1, \dots, \pi_k)}} [X = 1] &= \sum_{(\pi_1, \dots, \pi_k) \in G} \Pr_{hash \leftarrow \mathcal{H}} [X = 1 \mid (\pi_1, \dots, \pi_k)] \cdot \Pr_{(\tilde{\pi}_1, \dots, \tilde{\pi}_k)} [(\tilde{\pi}_1, \dots, \tilde{\pi}_k) = (\pi_1, \dots, \pi_k)] \\ &\geq \frac{1}{4(h+v)} \sum_{(\pi_1, \dots, \pi_k) \in G} \Pr_{(\tilde{\pi}_1, \dots, \tilde{\pi}_k)} [(\tilde{\pi}_1, \dots, \tilde{\pi}_k) = (\pi_1, \dots, \pi_k)] = \frac{\gamma}{4(h+v)} \end{aligned}$$

---

**Algorithm: FindHash**

---

**Oracle:** A solver circuit for  $k$ -wise direct product of DWVP  $C^{(\cdot, \cdot)}$  with oracle access to hint and verification oracle.

**Input:**  $\mathcal{H}$  a family of pairwise independent hash functions  $Q \rightarrow \{0, 1, \dots, 2(h+v) - 1\}$

---

For  $i = 1$  to  $16(h+v)^2/\gamma^2$

$hash \xleftarrow{\$} \mathcal{H}$

$count := 0$

**For**  $j := 1$  to  $16(h+v)^2/\gamma^2$

$(\pi_1, \dots, \pi_k) \xleftarrow{\$} \{0, 1\}^{kl}$

Run  $A^{P^{(g)}, C^{(\cdot, \cdot)}}(\pi_1, \dots, \pi_k)$

Let  $(\tilde{q}, y^{(k)})$  be the first successful verification query.

Let  $G$  be a set of all  $q$  used in hint or verification queries asked before  $(\tilde{q}, y^{(k)})$ .

**If**  $\Gamma_V^{(g)}(\tilde{q}, y^{(k)}) = 1 \wedge G \subseteq P_{hash}$

$count := count + 1$

**If**  $count \geq 4(h+v)/\gamma$

**return**  $hash$

**return**  $\perp$

---

We show that the algorithm **FindHash** chooses a hash function such that almost surely the success probability of  $C$  in random experiment  $E$  with respect to set  $P_{hash}$  is at least  $\frac{\gamma}{4(h+v)}$ . Let  $\mathcal{H}_{Good}$  denote the family of hash functions for which  $\Pr_{(\pi_1, \dots, \pi_k)} [X] \geq \frac{\gamma}{4(h+v)}$  and  $X_1, \dots, X_i$  be binary random variables such that for a fixed hash function

$$X_i = \begin{cases} 1 & \text{if in } i\text{th iteration variable } count \text{ is increased} \\ 0 & \text{otherwise} \end{cases}$$

We first show that it is unlikely that the algorithm **FindHash** returns  $hash \notin \mathcal{H}_{Good}$ . For  $hash \notin \mathcal{H}_{Good}$  we have  $\mathbb{E}_{(\pi_1, \dots, \pi_k)} [X_i] < \frac{\gamma}{4(h+v)}$ . We use Chernoff inequality and obtain

$$\Pr_{(\pi_1, \dots, \pi_k)} \left[ \frac{1}{N} \sum_{i=1}^N X_i \geq (1 + \delta) \frac{\gamma}{4(h+v)} \right] \leq \Pr_{(\pi_1, \dots, \pi_k)} \left[ \frac{1}{N} \sum_{i=1}^N X_i \geq (1 + \delta) \mathbb{E}[X_i] \right] \leq e^{-\frac{\gamma}{4(h+v)} N \delta^2 / 3}$$

The probability that  $hash \in \mathcal{H}_{Good}$  is not returned by the algorithm is

$$\Pr_{(\pi_1, \dots, \pi_k)} \left[ \frac{1}{N} \sum_{i=1}^N X_i \leq (1 - \delta) \frac{\gamma}{4(h+v)} \right] \leq \Pr_{(\pi_1, \dots, \pi_k)} \left[ \frac{1}{N} \sum_{i=1}^N X_i \leq (1 - \delta) \mathbb{E}[X_i] \right] \leq e^{-\frac{\gamma}{4(h+v)} N \delta^2 / 3}$$

Finally, we show that almost surely **FindHash** picks in one of its iteration a hash function that is in  $\mathcal{H}_{Good}$ . From the fact that the random variable  $X$  is binary distributed we have

$$\mathbb{E}_{\substack{hash \leftarrow \mathcal{H} \\ (\pi_1, \dots, \pi_k)}} [X] \geq \frac{\gamma}{4(h+v)}$$

Let  $Y_i$  be a binary random variable

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

We make use of the fact that if a function from  $\mathcal{H}_{Good}$  is picked, then it is returned almost surely. Therefore,  $\mathbb{E}[Y_i] \geq \frac{\gamma}{4(h+v)}$  and we can use Chernoff bound to obtain

$$\begin{aligned} \Pr_{hash \leftarrow \mathcal{H}} \left[ \frac{1}{K} \sum_{i=1}^K Y_i = 0 \right] &\leq \Pr_{hash \leftarrow \mathcal{H}} \left[ \frac{1}{K} \sum_{i=1}^K Y_i \leq (1 - \delta) \frac{\gamma}{4(h+v)} \right] \\ &\leq \Pr_{hash \leftarrow \mathcal{H}} \left[ \frac{1}{K} \sum_{i=1}^K Y_i \leq (1 - \delta) \mathbb{E}[Y_i] \right] \leq e^{-\delta^2 K \mathbb{E}[Y_i]/2} \end{aligned}$$

We see that the bound stated in the lemma 1.4 is achieved for valid for  $\delta = \frac{1}{2}$  and  $K = N = 16(h+v)^2/\gamma^2$   $\square$

**Experiment**  $E^{P^{(g)}, C^{(\cdot, \cdot)}, hash}(\pi_1, \dots, \pi_k)$

Solving  $k$ -wise direct product of DWVP with respect to the set  $P_{hash}$

**Oracle:** Problem poser for  $k$ -wise direct product  $P^{(g)}$

A solver circuit for  $k$ -wise direct product  $C^{(\cdot, \cdot)}$

A function  $hash : Q \leftarrow \{0, \dots, 2(h+v) - 1\}$

**Input:** Random bitstring  $(\pi_1, \dots, \pi_k) \in \{0, 1\}^{kl}$

$\pi^{(k)} := (\pi_1, \dots, \pi_k)$

$(x^k, \Gamma_V^{(g)}, \Gamma_H^{(k)}) := P^{(g)}(\pi^k)$

Run  $C^{\Gamma_V^{(g)}, \Gamma_H^{(k)}}(x^{(k)})$

Let  $(q_j, y_j^{(k)})$  be the first successful verification query if  $C^{\Gamma_V^{(g)}, \Gamma_H^{(k)}}$  succeeds or an arbitrary verification query when it fails.

**If**  $(\forall i < j : q_i \notin P_{hash})$  and  $q_j \in P_{hash}$  and  $\Gamma_V^{(g)}(q_j, y_j^{(k)}) = 1$

**return** 1

**else**

**return** 0

A canonical success is a situation when a solver  $C$  for fixed  $hash$  and  $P^{(1)}$  succeeds in a random experiment  $E$ .

**Random experiment**  $F^{P^{(1)}, D, hash}(\pi)$

Solving a single DWVP with respect to the set  $P_{hash}$

**Oracle:** A dynamic weakly verifiable puzzle  $P^{(1)}$

A solver circuit for a single DWVP  $D$

A function  $hash : Q \rightarrow \{0, 1, \dots, 2(h+v) - 1\}$

**Input:** Random bitstring  $\pi \in \{0, 1\}^l$

$(x, \Gamma_v, \Gamma_H) := P^{(1)}(\pi)$

Run  $D^{\Gamma_v, \Gamma_H}(x)$

Let  $(\tilde{q}_j, \tilde{r}_j)$  be the first successful verification query if  $D^{\Gamma_v, \Gamma_H}(x)$  succeeds or

```

    an arbitrary verification query when it fails.
If  $(\forall i < j : q_i \notin P_{hash})$  and  $q_j \in P_{hash}$  and  $\Gamma_V(q_j) = 1$  then
    return 1
else
    return 0

```

**Lemma 1.5** *Security amplification of a dynamic weakly verifiable puzzle with respect to set  $P_{hash}$ .*

For a fixed dynamic weakly verifiable puzzle  $P^{(1)}$  there exists an algorithm  $Gen(C, g, \varepsilon, \delta, n, v, h, hash)$ , which takes as input a circuit  $C$ , a monotone function  $g$ , a function  $hash : Q \rightarrow \{0, \dots, 2(h+v) - 1\}$ , parameters  $\varepsilon, \delta, n$ , number of verification  $v$ , and hint  $h$  queries asked by  $C$ , and outputs a circuit  $D$  such that following holds:  
If  $C$  is such that

$$\Pr_{(\pi_1, \dots, \pi_k)} [E^{P^{(g)}, C, Hash}(\pi_1, \dots, \pi_k)] \geq \Pr_{\mu \leftarrow \mu_\delta^k} [g(\mu) = 1] + \varepsilon$$

then  $D$  satisfies almost surely

$$\Pr_\pi [F^{P^{(1)}, D, Hash}(\pi) = 1] \geq (\delta + \frac{\varepsilon}{6k})$$

and  $Size(D) \leq Size(C) \frac{6k}{\varepsilon}$  and  $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$ .

**Circuit**  $\tilde{C}^{\Gamma_V^{(g)}, \Gamma_H^{(g)}, hash, C}(x_1, \dots, x_k)$   
Circuit  $\tilde{C}$  has good canonical success probability.

**Oracle:**  $\Gamma_V^{(g)}, \Gamma_H^{(g)}, hash, C$   
**Input:**  $k$ -wise direct product of puzzles  $(x_1, \dots, x_k)$

```

Run  $C^{(\cdot)}(x_1, \dots, x_k)$ 
If  $C$  asks a hint query  $q$  then
    If  $q \in P_{hash}$  then
        return  $\perp$ 
    else
        answer the hint query with  $\Gamma_H^{(k)}(q)$ 

If  $C$  asks a verification query  $(q, y_1, \dots, y_k)$  then
    If  $q \in P_{hash}$  then
        ask the verification query  $(q, y_1, \dots, y_k)$ 
        stop the execution
    else
        answer verification query with 0

return  $\perp$ 

```

**Lemma 1.6** For fixed  $P^{(g)}$  it is true that

$$\Pr_{(\pi_1, \dots, \pi_k)} [E^{P^{(g)}, C, Hash}(\pi_1, \dots, \pi_k) = 1] \leq \Pr_{(\pi_1, \dots, \pi_k)} [\Gamma_V^{(g)}(\tilde{C}^{\Gamma_V^{(g)}, \Gamma_H^{(g)}, Hash}( \pi_1, \dots, \pi_k)) = 1].$$

**Proof** We fix the randomness  $(\pi_1, \dots, \pi_k)$  used in the random experiment  $E$ . Let  $x^{(k)} = (x_1, \dots, x_k)$  be a set of puzzles generated in the random experiment  $E$  for the randomness  $(\pi_1, \dots, \pi_k)$ . If  $C$  succeeds canonically for the set of puzzles  $x^{(k)}$ , then also circuit  $\tilde{C}$  that runs  $C$  on the same set of puzzles succeeds. Using the definition of conditional expectation, we conclude that

$$\begin{aligned}
\Pr[E^{P^{(g)}, C, \text{hash}}(\pi^{(k)}) = 1] &= \sum_{\pi^{(k)} \in \{0,1\}^{kl}} \Pr[E^{P^{(g)}, C, \text{hash}}(\tilde{\pi}^{(k)}) = 1 | \pi^{(k)} = \tilde{\pi}^{(k)}] \Pr[\pi^{(k)} = \tilde{\pi}^{(k)}] \\
&\leq \sum_{\pi^{(k)} \in \{0,1\}^{kl}} \Pr[E^{P^{(g)}, \tilde{C}, \text{hash}}(\tilde{\pi}^{(k)}) = 1 | \pi^{(k)} = \tilde{\pi}^{(k)}] \Pr[\pi^{(k)} = \tilde{\pi}^{(k)}] \\
&= \Pr[E^{P^{(g)}, \tilde{C}, \text{hash}}(\pi^{(k)}) = 1]
\end{aligned}$$

□

---

**Algorithm**  $\text{Gen}(\tilde{C}, g, \varepsilon, \delta, n)$

---

**Oracle:**  $\tilde{C}, g$

**Input:**  $\varepsilon, \delta, n$

**Output:** A circuit  $D$

---

**For**  $i := 1$  to  $\frac{6k}{\varepsilon} \log(n)$

$\pi^* \leftarrow \{0, 1\}^l$

$\tilde{S}_{\pi^*, 0} := \text{EvaluateSurplus}(\pi^*, 0)$

$\tilde{S}_{\pi^*, 1} := \text{EvaluateSurplus}(\pi^*, 1)$

**If**  $\tilde{S}_{\pi^*, 0} \geq (1 - \frac{3}{4k})\varepsilon$  or  $\tilde{S}_{\pi^*, 1} \geq (1 - \frac{3}{4k})\varepsilon$

$\tilde{C}' := \tilde{C}$  with the first input fixed on  $\pi^*$

**return**  $\text{Gen}(\tilde{C}', g, \varepsilon, \delta, n)$

// all estimates are lower than  $(1 - \frac{3}{4k})\varepsilon$

**return**  $D^{\tilde{C}}$

**EvaluateSurplus** $(\pi^*, b)$

**For**  $i := 1$  to  $N_k$

$\pi^{(k)} \leftarrow \{0, 1\}^{lk}$

$(c_1, \dots, c_k) := \text{EvaluatePuzzles}(\pi^*, \pi^{(k)})$

$\tilde{S}_{\pi^*, b}^i := g(b, c_2, \dots, c_k) - \Pr_{(u_2, \dots, u_k)}[g(b, u_2, \dots, u_k) = 1]$

**return**  $\frac{1}{N_k} \sum_{i=1}^{N_k} \tilde{S}_{\pi^*, b}^i$

**EvalutePuzzles** $(\pi^*, \pi^{(k)})$

$(x^k, \Gamma_V^{(g)}, \Gamma_H^{(g)}) := P^{(g)}(\pi^*, \pi_2, \dots, \pi_k)$

**For**  $i = 2$  to  $k$

$(x_1, \Gamma_v^{(i)}, \Gamma_H^{(i)}) := P^{(1)}(\pi_i)$

$(q, y^k) := \tilde{C}_{\Gamma_V^{(g)}, \Gamma_H^{(g)}}(x^*, x_2, \dots, x_k)$

**For**  $i = 1$  to  $k$

$c_i := \Gamma_v^i(q, y_i)$

**return**  $(c_1, \dots, c_k)$

**Circuit**  $D^{\tilde{C}}$

---

**Oracle:**  $\tilde{C}, P^{(1)}$

---

**For**  $i := 1$  to  $\frac{6k}{\varepsilon} \log(\frac{6k}{\varepsilon})$   
     $\pi^k \leftarrow \{0, 1\}^k$   
     $(c_1, \dots, c_k) := \text{EvaluatePuzzles}(\pi, \pi^{(k)})$   
    **If**  $g(1, c_2, \dots, c_k) = 1$  and  $g(0, c_2, \dots, c_k) = 0$   
         $(q, y_1, \dots, y_k) := \tilde{C}(\pi^*, \pi_2, \dots, \pi_k)$   
        **return**  $y_1$   
**return**  $\perp$

The algorithm *Gen* recursively builds the circuit that have high success probability in solving a single DWVP.