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 Γ_V, Γ_H and a puzzle $x \in \{0,1\}^*$. The circuit Γ_V takes as input $q \in Q$ and an answer $y \in \{0,1\}^*$. If $\Gamma_V(q,y) = 1$ then y is a correct solution of a puzzle x for q. The circuit Γ_H on input q provides a hint such that $\Gamma_V(q, \Gamma_H(q)) = 1$. The probabilistic algorithm S, called a solver, has oracle access to Γ_V and Γ_H. The calls of S to Γ_V are verification queries and to Γ_H are hint queries. The solver S can ask at most h hint queries, v verification queries, and successfully solves 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 $g : \{0,1\}^k \to \{0,1\}$ be a monotone function, and $P^{(1)}$ a problem poser used to generate an instance of DWVP. A k-wise direct product of dynamic weakly verifiable puzzles ($DWVP^k$) is defined by a probabilistic algorithm $P^{(g)}(\pi_1, \ldots, \pi_k)$, where $(\pi_1, \ldots, \pi_k) \in \{0,1\}^{k \cdot l}$ is chosen uniformly at random. The algorithm $P^{(g)}(\pi_1, \ldots, \pi_k)$ generates k independent instances of dynamic weakly verifiable puzzles, where the i-th instance $(x_i, \Gamma_V^{(i)}, \Gamma_H^{(i)})$ is produced by executing $P^{(1)}(\pi_i)$. Finally, $P^{(g)}$ outputs a verification circuit

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

a hint circuit

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

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

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 $(q, y^{(k)}) := (q, y_1, \ldots, y_k)$ such that $\Gamma_V^{(g)}(q, y_1, \ldots, y_k) = 1$, and it has not previously asked for a hint query on this q.

A dynamic weakly verifiable puzzle is special case of k-wise direct product, when k equals one and g is identity function g. Therefore, we can consider following random experiment in which a k-wise direct product of DWVP (or for k equal one a single DWVP) defined by $P^{(k)}$ is solved by a circuit C that takes as input puzzles and possibly a random bitstring.

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

Oracle: A problem poser $P^{(\cdot)}$ and a solver circuit $C^{(\cdot,\cdot)}$.
Input: Bitstrings $\pi^{(\cdot)}$ and r.

```
$(x^{(\cdot)}, \Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)}) := P^{(\cdot)}(\pi^{(\cdot)})$
Run $C^{\Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)}}(x^{(\cdot)}, r)$
    Let $Q_{Solved} := \{q : C^{\Gamma_V^{(\cdot)}, \Gamma_V^{(\cdot)}}$ asked a verification query $(q, y^{(\cdot)})$ and $\Gamma_V^{(\cdot)}(q, y^{(\cdot)}) = 1\}$
    Let $Q_{Hint} := \{q : C^{\Gamma_V^{(\cdot)}, \Gamma_H^{(\cdot)}}$ asked a hint query on q$\}$
If $\exists q \in Q_{solved} : q \notin Q_{Hint}$ then
    return 1
else
    return 0
```

Theorem 1.3 ***Security amplification for 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 ε, δ, 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

$$\Pr_{(\pi_1,\ldots,\pi_k)\in\{0,1\}^{kl}}[A^{P^{(g)},C}(\pi_1,\ldots,\pi_k,r)=1] \geq \frac{(h+v)}{8}\left(\Pr_{\mu\leftarrow\mu_\delta^k}[g(\mu)=1]+\varepsilon\right)$$

then D satisfies almost surely

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

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

From theorem (1.3) we conclude that if there is no good algorithm for a single DWVP then it is not possible to find a good algorithm for k-wise direct product of DWVP.

The algorithm Gen tries to find $k-1$ puzzles and a position for an input puzzle x, such that when C runs with $k-1$ puzzles and x placed on a right position, then x is solved correctly often. To find a good position for x and good remaining $k-1$ puzzles we need to run C several times. It may happen that in one of this runs C ask for a hint query on some index q, and in one of the later runs we find a set of puzzles and a position for x such that x is solved often. However, we need an additional requirement that this happens often for q on which a hint query was not asked before. To satisfy this new requirement we split the set Q.

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

Experiment $E^{P^{(g)},C^{(\cdot)(\cdot)},hash}(\pi_1,\ldots,\pi_k,r)$
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 $hash : Q \leftarrow \{0,\ldots,2(h+v)-1\}$
Input: Random bitstrings: $(\pi_1,\ldots,\pi_k) \in \{0,1\}^{kl}$ and $r$.

$\pi^{(k)} := (\pi_1,\ldots,\pi_k)$
$(x^k, \Gamma_V^{(g)}, \Gamma_H^{(k)}) := P^{(g)}(\pi^k)$
Run $C^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x^{(k)}, r)$
    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
```

For fixed $hash$ and $P^{(1)}$ a canonical success of C is a situation when $E^{P^{(g)},C^{(\cdot)(\cdot)},hash}(\pi_1,\ldots,\pi_k,r) = 1$. We show that if C often solves successfully the k-wise direct product of DWVP, then it also often succeeds canonically.

Lemma 1.4 ***Success probability in solving a k-wise direct product of DWVP with respect to a function hash.***
For a fixed $P^{(g)}$ let C succeed in solving a k-wise direct product of DWVP produced by $P^{(g)}$ with probability γ, asking at most 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 $hash : Q \to \{0,\ldots,2(h+v)-1\}$ such that canonical success probability of C with respect to P_{hash} is at least $\frac{\gamma}{8(h+v)}$.

Proof Let $\mathcal{H}$ be a family of pairwise independent hash functions $Q \to \{0,1,\ldots,2(h+v)-1\}$. For all $i \neq j \in \{1,\ldots,(h+v)\}$ and $k,l \in \{0,1,\ldots,2(h+v)-1\}$ by pairwise independence property of $\mathcal{H}$ 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 $P^{(g)}, C$ and $(\pi_1,\ldots,\pi_k)$ in the random experiment E 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 $hash(q_i)\neq 0$. Conditioned on the event $hash(q_i)=0$, we get

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

Now we use (0.0.1) twice and obtain

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

Finally, we use union bound and $j \leq (h+v)$ 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_A (G_E) denote the set of all $(\pi_1,\ldots,\pi_k)$ for which C succeeds in the random experiment A (E). If for fixed $(\pi_1,\ldots,\pi_k)$ C succeeds in the random experiment E, then it also succeeds in the random experiment A. Hence, $G_E \subseteq G_A$ and we get

$$\Pr_{\substack{hash\leftarrow\mathcal{H}\\(\pi_1,\ldots,\pi_k)}}[E^{P^{(g)},C^{(\cdot,\cdot)},hash}(\pi_1,\ldots,\pi_k)=1] = \mathbb{E}_{(\pi_1,\ldots,\pi_k)\in G_A}\left[\Pr_{hash\leftarrow\mathcal{H}}[X=1]\right] \geq \frac{\gamma}{4(h+v)}. \quad (0.0.2)$$

Algorithm: FindHash

Oracle: A solver circuit for a k-wise direct product of DWVP $C^{(\cdot,\cdot)}$.
Input: A set $\mathcal{H}$.

```
For $i = 1$ to $32(h+v)^2/\gamma^2$
```

```
    $hash \xleftarrow{\$} \mathcal{H}$
    $count := 0$
    **For** $j := 1$ to $32(h+v)^2/\gamma^2$
        $(\pi_1, \ldots, \pi_k) \xleftarrow{\$} \{0,1\}^{kl}$
        **If** $E^{P(g), C^{(\cdot,\cdot)}, hash}(\pi_1, \ldots, \pi_k) = 1$ **then**
            $count := count + 1$
    **If** $\frac{\gamma^2}{32(h+v)^2} count \geq \frac{\gamma}{6(h+v)}$
        **return** $hash$
**return** $\perp$
```

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

$$\Pr_{(\pi_1,\ldots,\pi_k)}[E^{P(g),C^{(\cdot,\cdot)},hash}(\pi_1,\ldots,\pi_k) = 1] \geq \frac{\gamma}{4(h+v)},$$

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

$$\Pr_{(\pi_1,\ldots,\pi_k)}[E^{P(g),C^{(\cdot,\cdot)},hash}(\pi_1,\ldots,\pi_k) = 1] \leq \frac{\gamma}{8(h+v)}.$$

Additionally, for a fixed *hash*, we define the binary random variables $X_1, \ldots, X_i, \ldots, X_N$ such that

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

We first show that it is unlikely that **FindHash** returns $hash \in \mathcal{H}_{Bad}$. For $hash \in \mathcal{H}_{Bad}$ we have $\mathbb{E}_{(\pi_1,\ldots,\pi_k)}[X_i] < \frac{\gamma}{8(h+v)}$. Therefore, for any fixed $hash \in \mathcal{H}_{Bad}$ using the Chernoff bound we get

$$\Pr_{(\pi_1,\ldots,\pi_k)}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \geq \frac{\gamma}{6(h+v)}\right] \leq \Pr_{(\pi_1,\ldots,\pi_k)}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \geq (1+\frac{1}{3})\mathbb{E}[X_i]\right] \leq e^{-\frac{\gamma}{4(h+v)}N/27}.$$

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

$$\Pr_{(\pi_1,\ldots,\pi_k)}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \leq \frac{\gamma}{6(h+v)}\right] \leq \Pr_{(\pi_1,\ldots,\pi_k)}\left[\frac{1}{N}\sum_{i=1}^{N} X_i \leq (1-\frac{1}{3})\mathbb{E}[X_i]\right] \leq e^{-\frac{\gamma}{4(h+v)}N/27}.$$

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

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

From equation (0.0.2) 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

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

The bound stated in the Lemma 1.4 is achieved for $\delta = \frac{1}{2}$ and $K = N = 32(h+v)^2/\gamma^2$. □

```
**Random experiment** $F^{P^{(1)},D,hash}(\pi)$
Solving a single DWVP with respect to the set $P_{hash}$
---
**Oracle:** A problem poser $P^{(1)}$ for DWVP.
    A solver circuit $D$ for a single DWVP.
    A function $hash : Q \to \{0,1,\dots,2(h+v)-1\}$.
**Input:** A random bitstrings $\pi \in \{0,1\}^l$, $r \in \{0,1\}^*$.
---
$(x, \Gamma_v, \Gamma_H) := P^{(1)}(\pi)$
Run $D^{\Gamma_V,\Gamma_H}(x,r)$
    Let $(\widetilde{q_j}, \widetilde{r_j})$ be the first successful verification query if $D^{\Gamma_V,\Gamma_H}(x)$ succeeds or
    an arbitrary verification query when it fails.
If $(\forall i < j : q_i \notin P_{hash}) \wedge q_j \in P_{hash} \wedge \Gamma_V(q_j) = 1$ then
    return 1
else
    return 0
```

Lemma 1.5 ***Security amplification of a dynamic weakly verifiable puzzle with respect to P_{hash}.***
For fixed $P^{(1)}$ there exists an algorithm $Gen(C, g, \varepsilon, \delta, n, v, h, hash)$, which takes as input a solver circuit C, a monotone function g, a function $hash : Q \to \{0,\dots,2(h+v)-1\}$, parameters ε, δ, 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

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

then D satisfies almost surely

$$\Pr_{\pi}[\Gamma_V^{(g)}(D^{P^{(1)},\widetilde{C},hash}(\pi)) = 1] \geq (\delta + \frac{\varepsilon}{6k})$$

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

TODO: Write sth about correspondents between $\pi \leftarrow x$

We define a solver circuit $\widetilde{C}$ such that if it succeeds, then it also succeeds canonical.

```
**Circuit** $\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},hash,C}(x_1,\dots,x_k)$
Circuit $\widetilde{C}$ has good canonical success probability.
---
**Oracle:** $\Gamma_V^{(g)}, \Gamma_H^{(k)}, hash, C$
**Input:** k-wise direct product of puzzles $(x_1,\dots,x_k)$
---
Run $C^{(\cdot,\cdot)}(x_1,\dots,x_k)$
    If $C$ asks a hint query $q$ then
        If $q \in P_{hash}$ then
            return $\perp$
        else
            return $\Gamma_H^{(k)}(q)$ to $C$
    If $C$ asks a verification query on $(q, y_1, \dots, y_k)$ then
        If $q \in P_{hash}$ then
            return $(q, y_1, \dots, y_k)$
        else
            answer the verification query with 0
return $\perp$
```

Lemma 1.6 *For fixed $P^{(g)}$, hash the following statement is true*

$$\Pr_{(\pi_1,\dots,\pi_k)}[E^{P^{(g)},C,hash}(\pi_1,\dots,\pi_k)=1] \le \Pr_{(\pi_1,\dots,\pi_k)}[\Gamma_V^{(g)}(\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},hash}(\pi_1,\dots,\pi_k))=1].$$

Proof We fix the a random bitstring $(\pi_1,\dots,\pi_k)$, *hash.* If C succeeds canonically then

$$\Gamma_V^{(g)}(\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},hash}(\pi_1,\dots,\pi_k))=1.$$

Using this observation, we conclude that

$$\begin{aligned}\Pr_{(\pi_1,\dots,\pi_k)}[E^{P^{(g)},C,hash}(\pi^{(k)})=1] &= \sum_{\pi^{(k)}\in\{0,1\}^{kl}} \Pr[E^{P^{(g)},C,hash}(\widetilde{\pi}^{(k)})=1|\pi^{(k)}=\widetilde{\pi}^{(k)}]\Pr[\pi^{(k)}=\widetilde{\pi}^{(k)}]\\ &\le \sum_{\pi^{(k)}\in\{0,1\}^{kl}} \Pr[\Gamma_V^{(g)}(\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},hash}(\widetilde{\pi}^{(k)}))=1 \mid \pi^{(k)}=\widetilde{\pi}^{(k)}]\Pr[\pi^{(k)}=\widetilde{\pi}^{(k)}]\\ &= \Pr[E^{P^{(g)},\widetilde{C},hash}(\pi^{(k)})=1]\end{aligned}$$ □

```
Algorithm $Gen(\widetilde{C}, g, \varepsilon, \delta, n)$
-----------------------------------------------------
Oracle: $\widetilde{C}, g$
Input: $\varepsilon, \delta, n$
Output: A circuit $D$
-----------------------------------------------------
If the number of puzzles to solve equals one then
    return $\widetilde{C}$

For $i := 1$ to $\frac{6k}{\varepsilon}\log(n)$
    $\pi^* \leftarrow \{0,1\}^l$
    $\widetilde{S}_{\pi^*,0} := EvaluateSurplus(\pi^*, 0)$
    $\widetilde{S}_{\pi^*,1} := EvaluateSurplus(\pi^*, 1)$
    If $\widetilde{S}_{\pi^*,0} \ge (1-\frac{3}{4k})\varepsilon$ or $\widetilde{S}_{\pi^*,1} \ge (1-\frac{3}{4k})\varepsilon$
        $\widetilde{C}' := \widetilde{C}$ with the first input fixed on $\pi^*$
        return $Gen(\widetilde{C}', g, \varepsilon, \delta, n)$
// all estimates are lower than $(1-\frac{3}{4k})\varepsilon$
return $D^{\widetilde{C}}$

EvaluateSurplus$(\pi^*, b)$
    For $i := 1$ to $N_k$
        $(\pi_2,\dots,\pi_k) \xleftarrow{\$} \{0,1\}^{(k-1)l}$
        $(c_1,\dots,c_k) := EvalutePuzzles(\pi^*, \pi_2, \dots, \pi_k)$
        $\widetilde{S}^i_{\pi^*,b} := g(b, c_2, \dots, c_k) - \Pr_{(u_2,\dots,u_k)}[g(b, u_2, \dots, u_k) = 1]$
    return $\frac{1}{N_k}\sum_{i=1}^{N_k}\widetilde{S}^i_{\pi^*,b}$

EvalutePuzzles($\pi^{(k)}$)
    $(x^{(k)},\Gamma_V^{(g)},\Gamma_H^{(k)}) := P^{(g)}(\pi^{(k)})$
    For $i := 1$ to $k$
        $(x_i,\Gamma_V^i,\Gamma_H^i) := P^{(1)}(\pi_i)$
    $(q,y^k) := \widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x_1,x_2,\dots,x_k)$
    For $i := 1$ to $k$
        $c_i := \Gamma_v^i(q,y_i)$
    return $(c_1,\dots,c_k)$
```

TODO: Circuit $\widetilde{C}$ gets as input puzzle find a nice way to genereate the puzzles as it is used in many places in the code. Also make EvalutePuzzles more general maybe it should take $\widetilde{C}$ as input?

```
Circuit $D^{\widetilde{C}}$
-----------------------------------------------
Oracle: $\widetilde{C}, P^{(1)}$
Input: puzzle $x^*$, a random bitstring $r \in \{0,1\}^*$
-----------------------------------------------
For $i := 1$ to $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})$
    $\pi^{(k)} \leftarrow \{0,1\}^{kl}$ //read $k \cdot l$ bits from $r$
    $(c_1,\dots,c_k) := EvaluatePuzzles(\pi^{(k)})$
    If $g(1,c_2,\dots,c_k) = 1$ and $g(0,c_2,\dots,c_k) = 0$
        $(q,y_1,\dots,y_k) := \widetilde{C}(x^*,x_2,\dots,x_k)$
        return $y_1$
return $\perp$
```

For $k = 1$ function $g(b)$ is either identity or a constant function. If g is identity then the success probability of $\widetilde{C}$ is as least $\delta + \varepsilon$ and $\widetilde{C}$ can be directly used to solve a puzzle. If the function g is constant the statement is vacuously true.

Let $(q, y_1, \dots, y_k)$ denote the output of $\widetilde{C}$. Additionally, let us denote by $c_i = \Gamma_V(q, y_i)$ whether (q, y_i) is a correct solution for a single puzzle. We define surplus as the following quantity:

$$S_{\pi^*,b} = \Pr_{\pi^{(k)}}[g(b,c_2,\dots,c_k) = 1] - \Pr_{\mu^{(k)}}[g(b,u_2,\dots,u_k) = 1] \tag{0.0.3}$$

The surplus $S_{\pi^*,b}$ tells us how good the algorithm $\widetilde{C}$ performs when the first puzzle is fixed, and value of c_1 is neglected. The procedure **EvaluateSurplus** returns the estimate for $\widetilde{S}_{\pi^*,b}$. All puzzles used during obtaining the estimate are generated by **EvaluatePuzzles**. Therefore, it is possible to provide answers for all hint and verification queries. The returned estimate $\widetilde{S}_{\pi^*,b}$ that differs from $S_{\pi*,b}$ by at most $\frac{\varepsilon}{4k}$ almost surely. Therefore, if $\widetilde{S}_{\pi^*,b} \geq (1 - \frac{3}{4k})\varepsilon$ then with high probability $S_{\pi*,b} \geq (1 - \frac{1}{k})\varepsilon$. In this case we use a new monotone binary function $g'(b_2,\dots,b_k) := g(b,b_2,\dots,b_k)$, and fix the first puzzle of $\widetilde{C}$ for the one generated by using the randomness π^*. The new circuit satisfies the conditions of Lemma 1.5 which means that we can use algorithm *Gen* for the new circuit $\widetilde{C}$ and monotone function g'.

If all estimates are less than $(1-\frac{1}{4k})\varepsilon$, then intuitively $\widetilde{C}$ does not perform much better on the remaining $k-1$ puzzles than an algorithm that solves each puzzle independent with probability δ. However, from the assumption we know that on all k puzzles $\widetilde{C}$ has high success probability. It means that in this case the first puzzle has to be correctly solved with substantial probability.

TODO: Explain the intuition why it may happen that we still can fail in the case of circuit $\widetilde{D}$.

We have to show that the success probability when *Gen* does not recurse is substantial. We fix a randomness π^* and thus also a puzzle x^*. For this fixed puzzle using (0.0.3) we get

$$\Pr_{\mu_\delta^k}[g(1,\mu_2,\dots,\mu_k)=1]-\Pr_{\mu_\delta^k}[g(0,\mu_2,\dots,\mu_k)=1]=$$
$$\Pr_{\pi^{(k)}}[g(1,c_2,\dots,c_k)=1\mid \pi_1=\pi^*]-\Pr_{\pi^k}[g(0,c_2,\dots,c_k)=1\mid \pi_1=\pi^*]-(S_{\pi^*,1}-S_{\pi^*,0}) \tag{0.0.4}$$

TODO: Better explain why we can write $\Pr(g()=1\wedge g()=0)$ as the equivalence for the difference.

From the monotonicity of g we know that for any set of tuples $(b_1,\dots,b_k)$ and sets $G_0=\{(b_1,b_2,\dots,b_k): g(0,b_2,\dots,b_k)=1\}$, $G_1=\{(b_1,b_2,\dots,b_k): g(1,b_2,\dots,b_k)=1\}$ we have $G_0\subseteq G_1$. Hence, we can write (0.0.4):

$$\Pr_{\mu_\delta^k}[g(1,\mu_2,\dots,\mu_k)=1\wedge g(0,\mu_2,\dots,\mu_k)=0]=$$
$$\Pr_{\pi^{(k)}}[g(1,c_2,\dots,c_k)=1\wedge g(0,c_2,\dots,c_k)=0\mid \pi_1=\pi^*]-(S_{\pi^*,1}-S_{\pi^*,0}). \tag{0.0.5}$$

Let $G_{\mu^{(k)}}$ denote the event $g(1,\mu_2,\dots,\mu_k)=1\wedge g(0,\mu_2,\dots,\mu_k)=0$, and correspondingly $G_{\pi^{(k)}}:=g(1,\pi_2,\dots,\pi_k)=1\wedge g(0,\pi_2,\dots,\pi_k)=0$. Then multiplying and dividing $\Pr[\Gamma_v^{(g)}(D(x^*,\pi^{(k)}))=1\mid \pi_1=\pi^*]$ by (0.0.5) we get

$$\Pr_r[\Gamma_V^{(g)}(D(x^*,r))=1\mid\pi_1=\pi^*]=\frac{\Pr_r[\Gamma_V^{(g)}(D(x^*,r))=1\mid\pi_1=\pi^*]\Pr_{\pi^{(k)}}[G_\pi\mid\pi_1=\pi^*]}{\Pr_{\mu_\delta^k}[G_\mu]}$$
$$-\frac{\Pr_r[\Gamma_V^{(g)}(D(x^*,r))=1\mid\pi_1=\pi^*](S_{\pi^*,1}-S_{\pi^*,0})}{\Pr_{\mu_\delta^k}[G_\mu]} \tag{0.0.6}$$

If output of circuit $D(x^*,r)\neq\perp$ then we denote $c_i:=\Gamma_V^i(q,y_i)$. We can write the first summand of (0.0.6) as

$$\Pr_r[\Gamma_V^{(g)}(D(x^*,r))=1\mid\pi_1=\pi^*]\Pr_{\pi^{(k)}}[G_\pi\mid\pi_1=\pi^*]=$$
$$\Pr_r[D(x^*,r)\neq\perp\mid\pi_1=\pi^*]\Pr_{\pi^{(k)}}[c_1=1\mid G_\pi,\pi_1=\pi^*]\Pr_{\pi^{(k)}}[G_\pi\mid\pi_1=\pi^*] \tag{0.0.7}$$

where we make use of the fact that the event G_π implies $D(x^*,r)\neq\perp$. We consider two cases. If $\Pr_{\pi^k}[g(1,c_2,\dots,c_k)=1\wedge g(0,c_2,\dots,c_k)=0\mid\pi_1=\pi^*]\le\frac{\varepsilon}{6k}$ then also

$$\Pr_{\pi^{(k)}}[c_1=1\mid G_\pi,\pi_1=\pi^*]\Pr_{\pi^{(k)}}[G_\pi\mid\pi_1=\pi^*]\le\frac{\varepsilon}{6k} \tag{0.0.8}$$

and in the case when $\Pr_{\pi^k}[g(1,c_2,\dots,c_k) = 1 \wedge g(0,c_2,\dots,c_k) = 0] > \frac{\varepsilon}{6k}$ then circuit D outputs $\perp$ only if it fails in all $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})$ iterations to find $\pi^{(k)}$ such that $g(1,c_2,\dots,c_k) = 1 \wedge g(0,c_2,\dots,c_k) = 0$ which happens with probability

$$\Pr_r[D(x^*,r) = \perp \mid \pi_1 = \pi^*] \le (1 - \frac{\varepsilon}{6k})^{\frac{6k}{\varepsilon}\log(\frac{\varepsilon}{6k})} \le \frac{\varepsilon}{6k}. \tag{0.0.9}$$

We conclude that in both cases we have

$$\begin{aligned}&\Pr_r[D(x^*,r) \neq \perp \mid \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[c_1 = 1 \mid G_\pi, \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[G_\pi \mid \pi_1 = \pi^*]\\ &\quad\ge \Pr_{\pi^{(k)}}[c_1 = 1 \mid G_\pi, \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[G_\pi \mid \pi_1 = \pi^*] - \frac{\varepsilon}{6k}\end{aligned} \tag{0.0.10}$$

Using definition of conditional probability we get

$$\begin{aligned}&\Pr_r[D(x^*,r) \neq \perp \mid \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[c_1 = 1 \mid G_\pi, \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[G_\pi \mid \pi_1 = \pi^*]\\ &\quad= \Pr_{\pi^{(k)}}[c_1 = 1 \wedge g(1,c_2,\dots,c_k) = 1 \wedge g(0,c_2,\dots,c_k) = 0 \mid \pi_1 = \pi^*] - \frac{\varepsilon}{6k}\\ &\quad= \Pr_{\pi^{(k)}}[g(c_1,c_2,\dots,c_k) = 1 \wedge g(0,c_2,\dots,c_k) = 0 \mid \pi_1 = \pi^*] - \frac{\varepsilon}{6k}\\ &\quad= \Pr_{\pi^{(k)}}[g(c_1,c_2,\dots,c_k) = 1 \mid \pi_1 = \pi^*] - \Pr_{\pi^{(k)}}[g(0,c_2,\dots,c_k) = 0 \mid \pi_1 = \pi^*] - \frac{\varepsilon}{6k}\end{aligned}$$

and finally by (0.0.3)

$$\begin{aligned}&\Pr_r[D(x^*,r) \neq \perp \mid \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[c_1 = 1 \mid G_\pi, \pi_1 = \pi^*]\Pr_{\pi^{(k)}}[G_\pi \mid \pi_1 = \pi^*]\\ &\quad= \Pr_{\pi^{(k)}}[g(c_1,c_2,\dots,c_k) = 1 \mid \pi_1 = \pi^*] - \Pr_{\mu_\delta^{(k)}}[g(0,\mu_2,\dots,\mu_k) = 0 \mid \pi_1 = \pi^*] - S_{\pi^*,0} - \frac{\varepsilon}{6k}.\end{aligned} \tag{0.0.11}$$

We insert this result into equation (0.0.6) to get

$$\begin{aligned}\Pr_{r,\pi}[D(x,r) = 1] &= \mathbb{E}_\pi[\Pr_r[D(x,r) = 1 \mid \pi_1 = \pi^*]]\\ &= \mathbb{E}_\pi\left[\frac{\Pr_{\pi^{(k)}}[g(c) = 1 \mid \pi_1 = \pi^*] - \Pr_{\mu_\delta^{(k)}}[g(0,\mu_2,\dots,\mu_k) = 0 \mid \pi_1 = \pi^*] - \frac{\varepsilon}{6k}}{\Pr_{\mu_\delta^k}[G_\mu]}\right]\\ &\quad- \mathbb{E}_\pi\left[\frac{S_{\pi^*,0} + \Pr_r[\Gamma_V^{(g)}(D(x^*,r)) = 1 \mid \pi_1 = \pi^*](S_{\pi^*,1} - S_{\pi^*,0})}{\Pr_{\mu_\delta^k}[G_\mu]}\right]\end{aligned} \tag{0.0.12}$$

For the second summand we want to show first that almost all estimates all low if we do not recurse. Let assume that

$$\Pr_\pi\left[\left(S_{\pi,0} \le (1 - \frac{1}{2k})\varepsilon\right) \wedge \left(S_{\pi,1} \le (1 - \frac{1}{2k})\varepsilon\right)\right] < 1 - \frac{\varepsilon}{6k}, \tag{0.0.13}$$

then the algorithm would recurse almost surely. Therefore, under the assumption that we do not recurse, we have almost surely

$$\Pr_\pi\left[\left(S_{\pi,0} \le (1 - \frac{1}{2k})\varepsilon\right) \wedge \left(S_{\pi,1} \le (1 - \frac{1}{2k})\varepsilon\right)\right] \ge 1 - \frac{\varepsilon}{6k}. \tag{0.0.14}$$

Let us define a set

$$\mathbb{X} = \left\{\pi : \left(S_{\pi,0} \le (1 - \frac{1}{2k})\varepsilon\right) \wedge \left(S_{\pi,1} \le (1 - \frac{1}{2k})\varepsilon\right)\right\} \tag{0.0.15}$$

and the complement of this set $\mathbb{X}^c$. We bound the second summand in (0.0.12)

$$\begin{aligned}
&\mathbb{E}_\pi \left[S_{\pi^*,0} + \Pr_r[\Gamma_V^{(g)}(D(x^*, r)) = 1 \mid \pi_1 = \pi^*](S_{\pi^*,1} - S_{\pi^*,0}) \right] \\
&\quad = \mathbb{E}_{\pi \in \mathbb{X}^c} \left[S_{\pi^*,0} + \Pr_r[\Gamma_V^{(g)}(D(x^*, r)) = 1 \mid \pi = \pi^*](S_{\pi^*,1} - S_{\pi^*,0}) \right] \\
&\qquad + \mathbb{E}_{\pi \in \mathbb{X}} \left[S_{\pi^*,0} + \Pr_r[\Gamma_V^{(g)}(D(x^*, r)) = 1 \mid \pi = \pi^*](S_{\pi^*,1} - S_{\pi^*,0}) \right] && (0.0.16) \\
&\quad \le \frac{\varepsilon}{6k} + \mathbb{E}_{\pi \in \mathbb{X}^c} \left[S_{\pi^*,0} + \Pr_r[\Gamma_V^{(g)}(D(x^*, r)) = 1 \mid \pi = \pi^*]((1 - \frac{1}{2k})\varepsilon - S_{\pi^*,0}) \right] && (0.0.17) \\
&\quad \le \frac{\varepsilon}{6k} + 1 - \frac{\varepsilon}{2k} = 1 - \frac{\varepsilon}{3k} && (0.0.18)
\end{aligned}$$

Finally, we insert this result into equation (0.0.12) and make use of the fact

$$\begin{aligned}
\Pr[g(u) = 1] &= \Pr[(g(0, \mu_2, \dots, \mu_k) = 1) \vee (g(1, \mu_2, \dots, \mu_k) = 1 \wedge g(0, \mu_2, \dots, \mu_k) = 0 \wedge \mu_1 = 1)] \\
&= \Pr[g(0, \mu_2, \dots, \mu_k) = 1] + \Pr[g(1, \mu_2, \dots, \mu_k) = 1 \wedge g(0, \mu_2, \dots, \mu_k) = 0] \Pr[\mu_1 = 1]
\end{aligned}$$

which yields

$$\Pr_{r,\pi}[D(x, r) = 1] \ge \mathbb{E}_\pi \left[\frac{\Pr_{\pi^{(k)}}[g(c) = 1 \mid \pi_1 = \pi^*] - \Pr_{\mu_\delta^{(k)}}[g(0, \mu_2, \dots, \mu_k) = 0] - (1 - \frac{1}{6k})\varepsilon}{\Pr_{\mu_\delta^k}[G_\mu]} \right]$$

Using the assumptions of Lemma 1.5, we get

$$\begin{aligned}
\Pr_{r,\pi}[D(x, r) = 1] &\ge \frac{\Pr_{\mu_\delta^{(k)}}[g(\mu) = 1] + \varepsilon + \Pr_{\mu_\delta^{(k)}}[g(0, \mu_2, \dots, \mu_k) = 0] - (1 - \frac{1}{6k})\varepsilon}{\Pr_{\mu_\delta^k}[G_\mu]} \\
&\ge \frac{\varepsilon + \delta \Pr_{\mu_\delta^{(k)}}[G_\mu] - (1 - \frac{1}{6k})\varepsilon}{\Pr_{\mu_\delta^k}[G_\mu]} \ge \delta + \frac{\varepsilon}{6k}
\end{aligned}$$