**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 $g : \{0,1\}^k \to \{0,1\}$ 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 $P^{(g)}(\pi_1,\ldots,\pi_k)$, where $(\pi_1,\ldots,\pi_k) \in \{0,1\}^{k\cdot l}$ are chosen uniformly at random. $P^{(g)}(\pi_1,\ldots,\pi_k)$ 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*

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

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

**Experiment** $A^{P^{(1)},D}(\pi)$
Solving a dynamic weakly verifiable puzzle.

---

**Oracle:** A problem poser $P$ for DWVP.
A solver circuit $D^{(\cdot,\cdot)}$ for DWVP.
**Input:** A bitstring $\pi \in \{0,1\}^l$.

---

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

**Experiment** $B^{P^{(g)},C^{(\cdot,\cdot)}}(\pi_1,\ldots,\pi_k)$
Solving k-wise direct product of dynamic weakly verifiable puzzles.

---

**Oracle:** A problem poser for $k$-wise direct product $P^{(g)}$.
A solver circuit for $k$-wise direct product $C^{(\cdot,\cdot)}$.
**Input:** Random bitstring $\{\pi_1,\ldots,\pi_k\} \in \{0,1\}^{kl}$.

---

$(x^{(k)}, \Gamma_V^{(g)}, \Gamma_H^{(g)}) := P^{(g)}(\pi^{(k)})$
Run $C^{(\Gamma_V^{(g)},\Gamma_H^{(k)})}(x^{(k)})$
Let $Q_{Solved} := \{q : D^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x^{(k)})$ asked a verification query on $(q, y^{(k)})$ and $\Gamma_V(q, y^{(k)}) = 1\}$
Let $Q_{Hint} := \{q : D^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x^{(k)})$ asked a hint query on q$\}$
**If** $q \notin Q_{Hint}$
**return** 1
**else**
**return** 0

**Theorem 1.3** ***Security amplification of a dynamic weakly verifiable puzzle.***
*Fix a problem poser $P^{(1)}$. There exists an algorithm $Gen(C, g, \varepsilon, \delta, n, v, h)$ which takes as input a circuit $C$, a monotone function $g$, parameters $\varepsilon, \delta$, a security parameter $n$, 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*

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

*then $D$ satisfies almost surely*

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

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

**Experiment** $E^{P^{(g)},C^{(\cdot)(\cdot)},Hash}(\pi_1,\ldots,\pi_k)$
Solving $k$-wise direct product with respect to the set $P_{hash}$

---

**Oracle:** Problem poser for k-wise direct product $P^{(g)}$
Solver circuit $C^{(\cdot)(\cdot)}$ with oracle access to hint and verification circuits
Function $Hash : Q \leftarrow \{0,\ldots,2(h+v)-1\}$
**Input:** Random bitstring $(\pi_1,\ldots,\pi_k) \in \{0,1\}^{lk}$

---

$\pi^{(k)} := (\pi_1,\ldots,\pi_k)$
$(x^k, \Gamma_V^{(g)}, \Gamma_H^{(g)}) := P^{(g)}(\pi^k)$
Run $C^{\Gamma_V^{(g)},\Gamma_H^{(g)}}(x^{(k)})$
Let $(q_j, y_j^{(k)})$ be the first successful verification query if $C^{\Gamma_V^{(g)},\Gamma_H^{(g)}}$ succeeds or an arbitrary verification query when it fails.

**If** $(\forall i < j : Hash(q_i) \neq 0)$ and $(Hash(q_j) = 1 \wedge \Gamma_V^{(g)}(q_j, y_j^{(k)}) = 1)$
**return** 1

**else**

**return** 0

**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 $\varepsilon$ making $h$ hint and $v$ verification queries. There exists a probabilistic algorithm, with oracle access to $C$, that runs in time $O((h+v)^4/\varepsilon^4)$ and with high probability outputs a function $Hash : Q \to \{0,\ldots,2(h+v)-1\}$ such that success probability of $C$ in random experiment $E$ with respect to the set $P_{Hash}$ is at least $\frac{\varepsilon}{8(h+v)}$.*

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

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

For a fixed $(\pi_1,\ldots,\pi_k)$ we define an event, denoted by $X$, that $hash(q_j) = 0$ and for every query $q_i$ asked before $j$ $hash(q_i) \neq 0$. We have

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

Now we use (0.0.1) and obtain

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

Using once more the property (0.0.1)

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

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

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

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

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

**Algorithm: FindHash**
Pick a hash function with high canonical success probability

---

**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 $hash : Q \to \{0,1,\ldots,2(h+v)-1\}$

For $i = 1$ to $64(h+v)^2/\varepsilon^2$

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

  $count := 0$

  **For** $j := 1$ to $64(h+v)^2/\varepsilon^2$

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

    $result := A^{P^{(g)}, C^{(\cdot),(\cdot)}}(\pi_1, \ldots, \pi_k)$

    **If** $result = 1$

      $count := count + 1$

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

    **return** $hash$

**return** $\perp$

We now show that the algorithm **FindHash** chooses a hash function such that almost surly the success probability of $C$ in random experiment $E$ with respect to set $P_{hash}$ is at least $\frac{\varepsilon}{4(h+v)}$. From the fact that the random variable $X$ is binary distributed we have

$$\mathop{\mathbb{E}}_{hash \leftarrow \mathcal{H}}[X] \geq \frac{\varepsilon}{4(h+v)}$$

Let $\mathcal{H}_{Good}$ denote the family of hash function for which $\Pr_{(\pi_1,\ldots,\pi_k)}[X] \geq \frac{\varepsilon}{4(h+v)}$. and $X_i$ be a binary random variable such that

$$X_i = \begin{cases} 1 & \text{if in } i\text{th iteration } A^{P^{(g)}, C^{(\cdot)(\cdot)}} = 1 \\ 0 & \text{otherwise .} \end{cases}$$

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 $\mathbb{E}_{(\pi_1,\ldots,\pi_k)}[X] < \frac{\varepsilon}{4(h+v)}$. We use Chernoff inequality and obtain

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

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

$$\Pr_{(\pi_1,\ldots,\pi_k)}[\frac{1}{N_i}\sum_{i=1}^{N_i} X_i \leq (1-\delta)\frac{\varepsilon}{4(h+v)}] \leq \Pr_{(\pi_1,\ldots,\pi_k)}[\frac{1}{N_i}\sum_{i=1}^{N_i} X_i \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{\varepsilon}{4(h+v)}N_i\delta^2/3}$$

Finally, we can similarly show that **FindHash** picks with high probability with one of its iteration a hash function that is in $\mathcal{H}_{Good}$. □

**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 $Hash : Q \to \{0, \ldots, 2(h+v)-1\}$, 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*

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

*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 $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$.*

**Random experiment** $F^{P^{(1)},D,Hash}(\pi)$
Solving a single DWVP with respect to the set $P_{hash}$

---

**Oracle:** A circuit D, a function $Hash$, a dynamic weakly verifiable puzzle $P^{(1)}$
**Input:** Random bitstring $\pi$

---

$(x, \Gamma_v, \Gamma_H) := P^{(1)}(\pi)$
Run $D^{\Gamma_V,\Gamma_H}(x)$
    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** $(\forall i < j : Hash(q_i) \neq 0)$ and $Hash(q_j) = 1$
    **return** 1
**else**
    **return** 0

**Circuit** $\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},Hash}(x_1, \ldots, x_k)$
Circuit $\widetilde{C}$ has good canonical success probability.

---

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

---

Run $C^{(.),(.)}(x_1, \ldots, x_k)$
    **If** $C$ asks hint query $q$ **then**
        **If** $Hash(q) = 0$ **then**
            **return** $\perp$
        **else**
            answer with $\Gamma_H^{(g)}(q)$

    **If** $C$ asks verification query $(q, y_1, \ldots, y_k)$ **then**
        **If** $hash(q) = 0$ **then**
            **return** $(q, y_1, \ldots, y_k)$
        **else**
            answer verification query with 0 **return** $\perp$

**Lemma 1.6**

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

**Proof** If $E^{P^{(g)},C,Hash}(\pi_1, \ldots, \pi_k) = 1$ then circuit $\Gamma_V^{(g)}(\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},Hash}(\pi_1, \ldots, \pi_k)) = 1$. □

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

---

**Oracle:** $\widetilde{C}, g$
**Input:** $\varepsilon, \delta, n$
**Output:** A circuit $D$

---

**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** $\widetilde{S}_{\pi^*,0} \geq (1 - \frac{3}{4k})\varepsilon$ or $\widetilde{S}_{\pi^*,1} \geq (1 - \frac{3}{4k})\varepsilon$
    $\widetilde{C}' := \widetilde{C}$ with the first input fixed on $\pi^*$
    **return** $Gen(\widetilde{C}', g, \varepsilon, \delta, n)$
// all estimates are lower than $(1 - \frac{3}{4k})\varepsilon$
$SolvePuzzle(\pi, \widetilde{C})$

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

**EvalutePuzzles**$(\pi^*, \pi^{(k)})$
  $(x^k, \Gamma_V^{(g)}, \Gamma_H^{(g)}) := P^{(g)}(\pi^*, \pi_2, \ldots, \pi_k)$
  **For** $i = 2$ to $k$
    $(x_1, \Gamma_v^{(i)}, \Gamma_H^{(i)}) := P^{(1)}(\pi_i)$
  $(q, y^k) := \widetilde{C}^{\Gamma_V^{(g)}, \Gamma_H^{(g)}}(x^*, x_2, \ldots, x_k)$
  **For** $i = 1$ to $k$
    $c_i := \Gamma_v^i(q, y_i)$
  **return** $(c_1, \ldots, c_k)$

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

---

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

---

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