Definition 1.1 ***Dynamic weakly verifiable puzzle***
A dynamic weakly verifiable puzzle (DWVP) is defined by a protocol between probabilistic algorithms $P(\pi)$ and $S(\rho)$. The algorithm P, called a problem poser, takes as input chosen uniformly at random bitstring π. The problem solver S takes as input a uniform random bitstring ρ. As the result of the protocol execution between P and S, P produces circuits Γ_V, Γ_H and a puzzle $x \in \{0,1\}^*$, S produces no output. 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 solver S receives a puzzle x, and 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.

TODO: Requirements on g (we calculate $\Pr(g(b_1,\dots,b_k) = 1)$).

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 is defined by a protocol between a probabilistic algorithms $P^{(g)}(\pi^{(k)})$ and $S(\rho)$, where $\pi^{(k)} := (\pi_1,\dots,\pi_k)$ and every $\pi_i \in \{0,1\}^l$ for $1 \le i \le k$ and ρ are chosen uniformly at random. The protocol execution generates sequentially k independent instances of DWVP, where the i-th instance $(x_i, \Gamma_V^i, \Gamma_H^i)$ is produced by $S(\rho)$ interacting with $P^{(1)}(\pi_i)$. 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 solver S receives puzzles $(x_1,\dots,x_k)$, has oracle access to $\Gamma_V^{(g)}, \Gamma_H^{(k)}$, and 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 $(q, y^{(k)}) := (q, y_1,\dots,y_k)$ such that $\Gamma_V^{(g)}(q, y^{(k)}) = 1$, and has not previously asked for a hint query on this q.

TODO: We abuse the notation using, when the protocol is executed $C(\rho)$, and in the second phase when the puzzles are solved $C^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x^{(k)},\rho)$.

Experiment $A^{P^{(k)},C^{(\cdot,\cdot)}}(\pi^{(k)},\rho)$
Solving a k-wise direct product of DWVP

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

$(x^{(k)}, \Gamma_V^{(g)}, \Gamma_H^{(k)}) := \langle P^{(k)}(\pi^{(k)}), C(\rho)\rangle_{P^{(k)}}$
Run $C^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x^{(k)},\rho)$
  Let $Q_{Solved} := \{q : C^{\Gamma_V^{(g)},\Gamma_H^{(k)}}$ asked a verification query $(q, y^{(k)})$ and $\Gamma_V^{(g)}(q, y^{(k)}) = 1\}$
  Let $Q_{Hint} := \{q : C^{\Gamma_V^{(g)},\Gamma_H^{(k)}}$ 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 a probabilistic algorithm $Gen(C, g, \varepsilon, \delta, n, v, h)$ which takes as input a solver circuit C for a 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

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

then D satisfies almost surely

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

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

Experiment $E^{P^{(g)},C^{(\cdot,\cdot)},hash}(\pi^{(k)},\rho)$

Oracle: A problem poser $P^{(g)}$ for a k-wise direct product.
  A solver circuit $C^{(\cdot,\cdot)}$ for a k-wise direct product.
  A function $hash : Q \to \{0,\dots,2(h+v)-1\}$.
Input: Random bitstrings: $\pi^{(k)}$, ρ.

$(x^{(k)}, \Gamma_V^{(g)}, \Gamma_H^{(k)}) := \langle P^{(g)}(\pi^{(k)}), C(\rho)\rangle$
Run $C^{\Gamma_V^{(g)},\Gamma_H^{(k)}}(x^{(k)},\rho)$
  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 $(\forall i < j : q_i \notin P_{hash}) \wedge q_j \in P_{hash} \wedge \Gamma_V^{(g)}(q_j, y_j^{(k)}) = 1$
  return 1
else
  return 0

Algorithm: FindHash

Oracle: A solver circuit $C^{(\cdot,\cdot)}$ for a k-wise direct product of DWVP.
  A problem poser $P^{(g)}$ for a k-wise direct product.
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 $32(h+v)^2/\gamma^2$

$\pi^{(k)} \xleftarrow{\$} \{0,1\}^{kl}$

$\rho \xleftarrow{\$} \{0,1\}^*$

If $E^{P(g),C^{(\cdot,\cdot)},hash}(\pi^{(k)}, \rho) = 1$ **then**

$count := count + 1$

If $\frac{\gamma^2}{32(h+v)^2} count \geq \frac{\gamma}{6(h+v)}$

return $hash$

return $\perp$

Circuit $\widetilde{C}^{\Gamma_V^{(g)},\Gamma_H^{(g)},hash,C}(x_1,\ldots,x_k)$

Circuit $\widetilde{C}$ has good canonical success probability.

Oracle: $\Gamma_V^{(g)}, \Gamma_H^{(k)}, hash, C$

Input: k-wise direct product of puzzles $(x_1,\ldots,x_k)$

Run $C^{(\cdot,\cdot)}(x_1,\ldots,x_k,\rho)$

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 $(q, y_1, \ldots, y_k)$ **then**

If $q \in P_{hash}$ **then**

return $(q, y_1, \ldots, y_k)$

else

answer the verification query with 0

return $\perp$

Algorithm $Gen(C, g, \varepsilon, \delta, n, v, h, hash)$

Oracle: C, g

Input: $\varepsilon, \delta, n, v, h$

Output: A circuit D, $hash$

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

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

return $Gen(\widetilde{C}', g, \varepsilon, \delta, n)$

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

return $D^{\widetilde{C}}$

EvaluateSurplus(π^*, b)

  For $i := 1$ to N_k

    $(\pi_2, \ldots, \pi_k) \xleftarrow{\$} \{0,1\}^{(k-1)l}$

    $(c_1, \ldots, c_k) := EvalutePuzzles(\pi^*, \pi_2, \ldots, \pi_k)$

    $\widetilde{S}^i_{\pi^*,b} := g(b, c_2, \ldots, c_k) - \Pr_{(u_2,\ldots,u_k)}[g(b, u_2, \ldots, u_k) = 1]$

  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

    $(x_i, \Gamma_V^i, \Gamma_H^i) := P^{(1)}(\pi_i)$

  $(q, y^k) := \widetilde{C}^{\Gamma_V^{(g)}, \Gamma_H^{(k)}}(x_1, x_2, \ldots, x_k)$

  For $i := 1$ to k

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

  return $(c_1, \ldots, c_k)$

Circuit $D^{\widetilde{C}, P^{(1)}}$

Oracle: A circuit $\widetilde{C}$ with the first n puzzles fixed, $P^{(1)}$

Input: A puzzle x^*, a random bitstring $r \in \{0,1\}^*$

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

  $\pi^{(k)} \leftarrow \{0,1\}^{(k-n-1)l}$ //read bits from r

  $(c_1, \ldots, c_{k-n-1}) := EvaluatePuzzles(\pi^{(k-n-1)})$

  If $g(1, c_2, \ldots, c_k) = 1 \wedge g(0, c_2, \ldots, c_k) = 0$

    For $i := 1$ to $k - n - 1$

      $(x_i, \Gamma_V^i, \Gamma_H^i) := P^{(1)}(\pi_i)$

    $(q, y_1, \ldots, y_{k-n-1}) := \widetilde{C}(x^*, x_2, \ldots, x_{k-n-1})$

    return y_1

return $\perp$