Contents

Chapter 1

Introduction

1.1 Security Amplification Theorems

Introduction to security amplification theorems and hardness implication statements. Example of classical results. Problems captured by weakly verifiable puzzles. Contribution of this thesis.

1.2 Weakly verifiable puzzles

1.3 Contribution of the Thesis

1.4 Organization of the Thesis

Overview of the content of the succeeding chapters.

Chapter 2

Preliminaries

In this section we set up notation and terminology used in the thesis.

2.1 Notation and Definitions

(Algorithms, Bitstrings and Circuits) We define a Boolean circuit as a directed acyclic graph with input vertices and vertices implementing logical functions *and*, *or*, and *not*. We denote Boolean circuits using capital letters from the Greek and English alphabet.

We define a *probabilistic circuit* as a Boolean circuit $C_{m,n} : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^*$, For input $x \in \{0,1\}^m$ we write to denote $C_{m,n}(x;r)$ where $r \in \{0,1\}^n$ is called auxiliary input. If a probabilistic circuit does not take input x, we slightly abuse notation and write $C_n(r)$. Similarly we use $\{C_n\}$ to denote a family of probabilist circuits that takes only auxiliary input. We make sure that it is clear from the context that probabilist circuits with only auxiliary input are not confused with circuits that do not take auxiliary input.

For a (probabilist) circuit C we write $\mathit{Size}(C)$ to denote the total number of vertices of C.

We define a *two phase circuit* $C := (C_1, C_2)$ as a circuit where in the first phase a circuit C_1 is used and in the second phase a circuit C_2. If C_1 and C_2 are probabilistic circuits we write $C(\delta) := (C_1, C_2)(\delta)$ to denote that in both phases C_1 and C_2 take as auxiliary input the same bitstring δ.

It is well known [AB09] that a probabilistic polynomial time algorithm can be represented as a circuit of polynomial size. Moreover, it can be computed in polynomial time and logarithmic space. Therefore, whenever we state a theorem about circuits we can also apply it to polynomial time algorithms.

We write $\mathit{poly}(\alpha_1, \ldots, \alpha_n)$ to denote a polynomial on variables $\alpha_1, \ldots, \alpha_n$. For an algorithm A we write $\mathit{Time}(A)$ to denote the number of steps it takes to execute A. Similarly, as for probabilistic circuits we write the randomness used by a probabilistic algorithm explicitly as a bitstring provided as an auxiliary input.

(Probabilities and distributions) For a finite set $\mathcal{R}$ we write $r \xleftarrow{\$} \mathcal{R}$ to denote that r is chosen from $\mathcal{R}$ uniformly at random. For $\delta \in \mathbb{R} : 0 \leq \delta \leq 1$ we write μ_δ to denote the Bernoulli distribution where outcome 1 occurs with probability δ and 0 with probability $1 - \delta$. Moreover, we use μ_δ^k to denote the probability distribution over k-tuples where each element 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.

Let $(\Omega, \mathcal{F}, \Pr)$ be a probability space and $n \in \mathbb{N}$. We say that an event $E_n \in \mathcal{F}$ happens *almost surely* or with *high probability* if $\Pr[E_n] \geq 1 - 2^{-n} poly(n)$.

TODO: Non-negligible function

(Interactive protocols) We are often interested in situations where two probabilistic algorithms interact with each other according to some protocol. We limit ourselves to the cases where algorithms interact by means of messages representable by bitstrings. A protocol execution between two probabilistic algorithms A and B is denoted by $\langle A, B \rangle$. The output of A in a protocol execution is denoted by $\langle A, B \rangle_A$ and of B by $\langle A, B \rangle_B$. A sequence of all messages sent by A and B in the protocol execution is called a communication transcript and is denoted by $\langle A, B \rangle_{trans}$.

(Oracle algorithms)

TODO: set up notation

Definition 2.1 (Polynomial time sampleable distribution) *We say that a distribution is polynomial time sampleable if it can be approximated by polynomial time algorithm up to exponential factor.*

Definition 2.2 (Pairwise independent family of efficient hash functions) *Let $\mathcal{D}$ and $\mathcal{R}$ be finite sets and $\mathcal{H}$ be a family of functions mapping values from $\mathcal{D}$ to values in $\mathcal{R}$. We say that $\mathcal{H}$ is an* efficient family of pairwise independent hash functions *if $\mathcal{H}$ has the following properties.*

(Pairwise independent) *For $\forall x \neq y \in \mathcal{D}$ and $\forall \alpha, \beta \in \mathcal{R}$, it holds*

$$\Pr_{hash \leftarrow \mathcal{H}}[hash(x) = \alpha \mid hash(y) = \beta] = \frac{1}{|\mathcal{R}|}.$$

(Polynomial time sampleable) *For every $hash \in \mathcal{H}$ the function hash is sampleable in time $poly(\log |\mathcal{D}|, \log |\mathcal{R}|)$.*

(Efficiently computable) *For every $hash \in \mathcal{H}$ there exists an algorithm running in time $poly(\log|\mathcal{D}|, \log|\mathcal{R}|)$ which on input $x \in \mathcal{D}$ outputs $y \in \mathcal{R}$ such that $y = hash(x)$.*

We note that the pairwise independence property is equivalent to

$$\Pr_{hash \leftarrow \mathcal{H}}[hash(x) = \alpha \wedge hash(y) = \beta] = \frac{1}{|\mathcal{R}|^2}.$$

It is well know [CW77] that there exists a family of hash functions meeting the above criteria.

Chapter 3

Weakly Verifiable Cryptographic Primitives

In this chapter we introduce weakly verifiable puzzles. Section 3.1 contains the formal definitions of dynamic weakly verifiable puzzles. In Section 3.2 we give a series of cryptographic primitives that are captured by dynamic weakly verifiable puzzles. Finally, in Section 3.3 we give an overview of the earlier research in this area that is primarily covered in [CHS04], [DIJK09], and [HS10].

3.1 Dynamic Weakly Verifiable Puzzles

We start by defining *dynamic weakly verifiable puzzles.*

Definition 3.1 (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 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 problem 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.*

The above definition generalizes the previous approaches contained in [CHS04], [DIJK09], [HS10]. Some of more interesting cryptographic primitives captured by this definition are described in Section 3.2. There is no loss of generality in assuming that a problem poser and problem solver are defined by probabilistic circuits. The definition embraces also a case a problem poser and a problem solver are probabilistic polynomial time algorithms by using the well know fact [Hol13b] that probabilistic algorithms can be transform into equivalent family of Boolean circuits of polynomial size.

3.2 Examples

In this section we give examples of cryptographic constructions that are generalized by definition 3.1.

3.2.1 Message Authentication Codes

Let us consider the setting in which two parties a *sender* and a *receiver* communicate over an insecure channel. Messages of the sender may be read and changed by a third party called an *adversary*. The receiver needs a way to ensure that received messages are indeed sent by the sender and have not been modified by the adversary. The solution is to use message authentication codes.

We consider a setting in which all parties are polynomial time algorithms, and messages can be represented as bitstring. Additionally, we assume that the sender and the receive share a secrete key to which an adversary has no access. The sender appends to every message a tag that is as computed as a function of the key and the message. The receiver, using the key, has a way to check whether the tag is valid for the received message. On the other hand, it should be hard for the adversary to find a tag and message that would be correctly validate by the receiver.

The following definition is based on [Mau13] and [Gol04].

Definition 3.2 (Message Authentication Codes (MAC)) *Let $\mathcal{M}$ be a set of messages, $\mathcal{K}$ a set of keys and finally $\mathcal{T}$ a set of tags. The message authentication code is a function $f : \mathcal{M} \times \mathcal{K} \to \mathcal{T}$, and it is secure if additionally satisfies the following condition:*

Let k be chosen uniformly at random from $\mathcal{K}$ and $\Gamma_H : \mathcal{M} \to \mathcal{T}$ be an oracle computing a tag for the key k and a message m. We say that MAC is secure if there is not probabilistic polynomial time algorithm that with non-negligible *probability outputs a message m', as well as corresponding MAC t' such that $f(m', k) = t$.*

We show how the above definition of MAC is captured by dynamic weakly verifiable puzzles. The first phase is non interactive. The k key corresponds to a bitstring $\pi \in \{0,1\}^n$ taken by a problem poser P_n. The problem poser outputs a hint circuit $\Gamma_H : \mathcal{M} \to \mathcal{T}$ that given a message computes a tag. and a verification circuit $\Gamma_V : \mathcal{M} \times \mathcal{T} \to \{0,1\}$ that on input $m \in \mathcal{M}$ and $t \in \mathcal{T}$ outputs one if and only if $f(m, k) = t$.

The adversary is a probabilistic polynomial time algorithm, thus equivalently it can be turn into a circuit of polynomial size. In the second phase the adversary takes no input (x^* is empty string), but is given oracle access to Γ_H and Γ_V. The adversary succeeds in breaking the security of MAC if it finds a valid tag $t' \in \mathcal{T}$ for a message $m' \in \mathcal{M}$ such that a hint for this message has no been asked before. That corresponds to asking a successful verification query to oracle circuit Γ_V. Conversely, if there is no polynomial time algorithm that succeeds in solving this dynamic weakly verifiable puzzle with non-negligible probability then the message authentication code is secure.

Thus, a game in which an adversary has to break security of MAC is a dynamic weakly verifiable puzzle which first phase is not interactive, as the adversary does not communicate with the problem poser.

3.2.2 Public Key Signature Scheme

First we give a definition of public key encryption scheme and security of such a scheme. Both definition are based on [Gol04].

Definition 3.3 (Public key signature scheme) *A public key encryption scheme is defined by a triple of probabilistic polynomial time algorithms: G the key generation algorithm, V the verification algorithm, S the verification algorithm such that the following conditions are satisfied:*

$G(1^n)$ outputs a pair of bitstrings k_{priv} a private key, k_{pub} a public key.

For every k_{priv}, k_{pub} output by G and $q \in Q$ it holds

$$\Pr[V(k_{pub}, q, S(k_{priv}, q))] = 1.$$

Definition 3.4 ***(Security of public key signature scheme with respect to the chosen message attack)*** *Let us consider a situation in which an adversary is given a public key k_{pub} and can ask for a signature of chosen messages. We say that the adversary succeeds if it finds a valid signature for a message for which a signature has not been asked. Furthermore, we say that the public key encryption scheme is secure if probability that any polynomial time bounded adversary succeeds in breaking the signature scheme with negligible probability.*

We show now that the public key signature schemes are also captured by definition 3.1. First the problem poser uses algorithm $G(1^n)$ to obtain a public and private key. Then it generates a hint circuit $\Gamma_H : \mathcal{M} \to S$ and a verification circuit $\Gamma_V : \mathcal{M} \times \mathcal{S} \to \{0,1\}$. The hint circuit takes as input $m \in \mathcal{M}$ and outputs a signature for m. The verification circuits checks whether signature $s \in S$ is a valid for message $m \in \mathcal{M}$. In the second phase the problem solver obtains k_{pub}, and is given oracle access to the hint circuit and verification circuit. The problem solver can ask a hint oracle for signature on chosen message.

Finally, it is clear that if a problem solver succeeds in solving a DWVP defined as above, it also breaks the secuirty of the public signature schemes.

Public key signature schemes are type of puzzles that are dynamic - the problem poser gain oracle access to hint and verification circuits, but are not interactive.

3.2.3 Bit Commitments

[Hol13a] Let us consider a bit commitment protocol, in which there are two parties a *sender* and a *receiver*. In the first phase the sender and the receiver interact and the sender commit to a value $b \in \{0,1\}$. We require that it is hard for the sender, after the commitment phase, to guess correctly the bit b. In the send phase the sender open the commitment by sending to the receiver the information y that let the receiver validate the commitment. We say that the protocol is secure if it is computationally hard for the sender to find to such strings y_0 and y_1 that both $(0, y_0)$ and $(1, y_1)$ are valid.

TODO: Security definition

TODO: Type of puzzles generalized by this

3.2.4 Automated Turing Tests

The goal of *Automated Turing Tests* is to distinguish humans from computers. A frequent application of *Automated Turing Tests* is preventing computer programs from accessing resources for human beings (like creating an email account). An example of implementation of *Automated Turing Tests* is CAPTCHA that was defined in [VABHL03]. Loossly speaking, CAPTCHA is a test that human can solve with probability close to 1, but it is hard to write a computer program that has a success probability comparable to the one achieved by humans. An example of CAPTACHA is sending a distorted text. It is easy for most humans to guess the right text, but it might be hard to write a program for which it would also be easy. We note that the definition of hardness is not precesily defined, as it is possible that there exists a program that reaches the success probability close to the one achieved by humans.

CAPTCHAs basing on guessing the distorted text are yet another example of weakly verifiable puzzles. In the first round the problem poser and problem solver engage in interactive protocol, such that after the execution of the protocol the problem poser has a way to verify the solution. The problem poser in the second round takes as input a distorted image, and try to guess the text that was used to generated it. We note that the standard distorted text base CAPTCHAs are non dynamic. The problem poser does not gain access to the hint oracle, only to the verification oracle to which it asks a single query.

Our definition captures also this the above type of problems, additionally it is also applicable in the broader context for a different AI problems.

As it is not know how good the possible algorithm can be to recognize CAPTCHA it is likely that the gap between human performance and a performance of computer programs may be small. Therefore, it is of interest to find a way to amplify this gap. It turns out that it is indeed possible which for not dynamic puzzles was proved in [HS10]. The proof presented in 4 applies also to dynamic context.

3.3 Previous results

3.3.1 Results of R.Canetti, S.Halevi, and M.Steiner

3.3.2 Results of Y.Dodis, R.Impagliazzo, R.Jaiswal, V.Kabanets

3.3.3 Results of T.Holenstein and G.Scheonebeck

3.4 Limitations of Security Amplification

Chapter 4

Security amplification for dynamic weakly verifiable puzzles

In this chapter we show that it is possible to amplify security of dynamic weakly verifiable puzzles. In section 4.1 we state the theorem, which is next proved in three steps. First, in Section 4.1.3, we show how to use to partition the domain on which hint and verification queries are asked, next we give a prove of security amplification under the simplifying assumption that there is no collisions of hint and verification queries. Finally, in Section 4.1.5 we combine both former steps which yields the desirable result.

4.1 Main theorem

We start by giving the definition of the k-wise direct product of weaky verifiable puzzles.

4.1.1 The k-wise direct product of weakly verifiable puzzle

Definition 4.1 (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 3.1. 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 $P_{kn}^{(g)}$ as in Definition 3.1. 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

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

If it is clear from the context, we omit the subscript 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 cannot be a verification query for which C 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 $(q, y)$ s.t. $\Gamma_V(q, y) = 1$ then
        return 1
return 0
```

We define the *success probability* of C in solving a puzzle defined by P as

$$\Pr_{\pi,\rho}[\mathit{Success}^{P,C}(\pi, \rho) = 1]. \tag{4.1}$$

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

Theorem 4.2 (Security amplification for dynamic weakly verifiable puzzles.) *Let $P_n^{(1)}$ be a fixed problem poser as in Definition 3.1 and $P_{kn}^{(g)}$ a problem poser for the k-wise direct product of $P_n^{(1)}$. Additionally, 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 $P_n^{(1)}$, $P_{kn}^{(g)}$. Furthermore, 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

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

then D is a two phase probabilistic circuit and 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, hint and verification circuits and asks at most $\frac{6k}{\varepsilon}\log\left(\frac{6k}{\varepsilon}\right)h$ hint queries and one verification query. Finally, $Time(Gen) = poly(k, \frac{1}{\varepsilon}, n, v, h)$ with oracle access to C.

4.1.2 Intuition

TODO: add intuition for hashing

The idea of the algorithm *Gen* is to output a circuit D that solves the input puzzle often. We know that C has high success probability in solving the k-wise direct product of $P^{(1)}$. The algorithm *Gen* tries to find a puzzle such that when C runs with this puzzle fixed on the first position and disregards whether this puzzle is correctly solved then the assumptions of Theorem 4.2 are true for the $(k-1)$-wise direct product. If it was possible to find such a puzzle, then *Gen* could recurse and solve a smaller problem. In the optimistic case we can reach $k=1$, which means that we found a good circuit for solving a single puzzle by just fixing the initial puzzles of C.

Otherwise, when the first position is disregarded then the success probability of C is not substantially better. This is remarkable, as we know that C performs good for the k-wise direct product. It means that the first position is important, in the sense that C solves the puzzle on that position unusually often. Therefore, it is reasonable to construct the circuit D using C by placing the input puzzle of D on that position, and then finding remaining $k-1$ puzzles. The $(k-1)$ remaining puzzles are generated by the circuit D, hence it is possible to check whether they are correctly solved by the circuit C. We know that circuit C has good success probability, and the puzzle on the first position is important. Therefore, if we are able to find a $(k-1)$ puzzles such that the fact whether the k-wise direct product is correctly solved depends on whether the puzzle on the first position is correctly solved then we can assume that C is often correct on this first position.

There are some problems with this approach, first we have to ensure that we can make a decision when the algorithm *Gen* should recurse and when not correctly with high probability. Then, we have to show that it is possible to find a puzzles such that C is often correct on the first position. Finally, we also have to be sure that we do not ask a hint query, on the final verification query to the oracle. To satisfy the last requirement we split the set Q.

4.1.3 Domain partitioning

Let $hash : Q \to \{0, 1, \ldots, 2(h + v) - 1\}$, 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 $hash(q) = 0$. Therefore, if C makes a verification query (q, y) such that $hash(q) = 0$, 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 the following experiment *CanonicalSuccess* in which the set Q is partitioned 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 $hash(q_j) = 0$, and no hint query q_i is asked before (q_j, y_j) such that $hash(q_i) = 0$.

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

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 $C_2^{\Gamma_V, \Gamma_H}(x, \rho)$ does not succeed for any verification query then
        return 0
    Let $(q_j, y_j)$ be the first verification query 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 circuit C for P with respect to a function *hash* as

$$\Pr_{\pi,\rho}[\mathit{CanonicalSuccess}^{P,C,hash}(\pi,\rho)=1]. \tag{4.2}$$

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

We show that if a solver circuit C for P often succeeds in the experiment *Success*, then there exists a function *hash* such that C also often succeeds in the experiment *CanonicalSuccess*.

Lemma 4.3 ***(Success probability in solving DWVP with respect to a function hash.)*** *For fixed P_n let C be a solver for P_n with success probability at least γ, asking at most h hint queries and v verification queries. Let $\mathcal{H}$ be an efficient family of pairwise independent hash functions $Q \to \{0,1,\ldots,2(h+v)-1\}$. There exists a probabilistic algorithm FindHash that takes as input parameters γ, n, h, v, and has oracle access to C and P_n. 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 (4.3). We fix a problem poser P and a solver C for P in the whole proof of Lemma 4.3. For $k,l \in \{1,\ldots,(h+v)\}$ and $\alpha,\beta \in \{0,1,\ldots,2(h+v)-1\}$ by the pairwise independence property, we have

$$\forall q_k \neq q_l \in Q : \Pr_{hash\leftarrow\mathcal{H}}[hash(q_k)=\alpha \mid hash(q_l)=\beta] = \Pr_{hash\leftarrow\mathcal{H}}[hash(q_k)=\alpha] = \frac{1}{2(h+v)}. \tag{4.3}$$

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

(q_j, y_j) denote the first query such that $\Gamma_V(q_j, y_j) = 1$. We have

$$\begin{aligned}
&\Pr_{hash \leftarrow \mathcal{H}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi^*, \rho^*) = 1 \right] \\
&\quad = \Pr_{hash \leftarrow \mathcal{H}} [hash(q_j) = 0 \wedge (\forall i < j : hash(q_i) \neq 0)] \\
&\quad = \Pr_{hash \leftarrow \mathcal{H}} [\forall i < j : hash(q_i) \neq 0 \mid hash(q_j) = 0] \Pr_{hash \leftarrow \mathcal{H}} [hash(q_j) = 0] \\
&\quad \overset{(4.3)}{=} \frac{1}{2(h+v)} \left(1 - \Pr_{hash \leftarrow \mathcal{H}} [\exists i < j : hash(q_i) = 0 \mid hash(q_j) = 0] \right) \\
&\quad \overset{(*)}{\geq} \frac{1}{2(h+v)} \left(1 - \sum_{i<j} \Pr_{hash \leftarrow \mathcal{H}} [hash(q_i) = 0 \mid hash(q_j) = 0] \right) \\
&\quad \overset{(4.3)}{=} \frac{1}{2(h+v)} \left(1 - \sum_{i<j} \Pr_{hash \leftarrow \mathcal{H}} [hash(q_i) = 0] \right) \\
&\quad \overset{(4.3)}{\geq} \frac{1}{4(h+v)},
\end{aligned} \tag{4.4}$$

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

$$\begin{aligned}
&\Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi, \rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \right] \\
&\quad = \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi, \rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \in \mathcal{P}_{\mathit{Success}} \right] \Pr_{\pi, \rho} [(\pi, \rho) \in \mathcal{P}_{\mathit{Success}}] \\
&\qquad + \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi, \rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \notin \mathcal{P}_{\mathit{Success}} \right] \Pr_{\pi, \rho} [(\pi, \rho) \notin \mathcal{P}_{\mathit{Success}}] \\
&\quad = \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi, \rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \in \mathcal{P}_{\mathit{Success}} \right] \Pr_{\pi, \rho} [(\pi, \rho) \in \mathcal{P}_{\mathit{Success}}] \\
&\quad \geq \Pr_{\substack{hash \leftarrow \mathcal{H} \\ \pi, \rho}} \left[\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1 \mid (\pi, \rho) \in \mathcal{P}_{\mathit{Success}} \right] \cdot \gamma \\
&\quad = \mathop{\mathbb{E}}_{(\pi, \rho) \in \mathcal{P}_{\mathit{Success}}} \left[\Pr_{hash \leftarrow \mathcal{H}} [\mathit{CanonicalSuccess}^{P,C,hash}(\pi, \rho) = 1] \right] \cdot \gamma \\
&\quad \overset{(4.4)}{\geq} \frac{\gamma}{4(h+v)}.
\end{aligned} \tag{4.5}$$

Algorithm FindHash$^{P,C}(\gamma, n, h, v)$

Oracle: A problem poser P, a solver circuit C for P.
Input: Parameters γ, n. The number of hint queries h and of verification queries v.
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:
        $\pi \leftarrow \{0,1\}^n$
        $\rho \leftarrow \{0,1\}^*$
        if $CanonicalSuccess^{P,C,hash}(\pi, \rho) = 1$ then
            $count := count + 1$
    if $count \geq \frac{\gamma}{12(h+v)} \frac{32(h+v)^2}{\gamma^2} n$ 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

$$\Pr_{\pi,\rho}\left[CanonicalSuccess^{P,C,hash}(\pi, \rho) = 1\right] \geq \frac{\gamma}{8(h+v)}, \tag{4.6}$$

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

$$\Pr_{\pi,\rho}\left[CanonicalSuccess^{P,C,hash}(\pi, \rho) = 1\right] \leq \frac{\gamma}{16(h+v)}. \tag{4.7}$$

Let N denote the number of iterations of the inner loop of FindHash. We consider a single iteration of the outer loop of FindHash in which $hash$ is fixed. We define independent, identically distributed, binary random variables $X_1, \ldots, X_N$ such that

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

We now turn to the case when $hash \in \mathcal{H}_{Bad}$ and show that it is unlikely that $hash$ is returned by FindHash. From (4.7) it follows that $\mathbb{E}_{\pi,\rho}[X_i] \leq \frac{\gamma}{16(h+v)}$. Therefore, for any fixed $hash \in \mathcal{H}_{Bad}$ using the Chernoff bound we get

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

where in $(*)$ we used the trivial facts that $h+v \geq 1$ and $\gamma \leq 1$. The probability that $\mathit{hash} \in \mathcal{H}_{\mathit{Good}}$, when picked, is not returned amounts

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

where we once more used the Chernoff bound. We now show that the probability of picking $\mathit{hash} \in \mathcal{H}_{\mathit{Good}}$ is at least $\frac{\gamma}{8(h+v)}$. To obtain a contradiction suppose that

$$\Pr_{\mathit{hash} \leftarrow \mathcal{H}}[\mathit{hash} \in \mathcal{H}_{\mathit{Good}}] < \frac{\gamma}{8(h+v)}. \tag{4.8}$$

From this it follows that we can bound probability of canonical success as follows

$$\begin{aligned}
&\Pr_{\substack{\mathit{hash} \leftarrow \mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,\mathit{hash}}(\pi,\rho)=1]\\
&\quad= \Pr_{\substack{\mathit{hash} \leftarrow \mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,\mathit{hash}}(\pi,\rho)=1 \mid \mathit{hash} \in \mathcal{H}_{\mathit{Good}}]\Pr_{\mathit{hash} \leftarrow \mathcal{H}}[\mathit{hash} \in \mathcal{H}_{\mathit{Good}}]\\
&\qquad+ \Pr_{\substack{\mathit{hash} \leftarrow \mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,\mathit{hash}}(\pi,\rho)=1 \mid \mathit{hash} \notin \mathcal{H}_{\mathit{Good}}]\Pr_{\mathit{hash} \leftarrow \mathcal{H}}[\mathit{hash} \notin \mathcal{H}_{\mathit{Good}}]\\
&\quad\leq \Pr_{\mathit{hash} \leftarrow \mathcal{H}}[\mathit{hash} \in \mathcal{H}_{\mathit{Good}}] + \Pr_{\substack{\mathit{hash} \leftarrow \mathcal{H}\\ \pi,\rho}}[\mathit{CanonicalSuccess}^{P,C,\mathit{hash}}(\pi,\rho)=1 \mid \mathit{hash} \notin \mathcal{H}_{\mathit{Good}}]\\
&\quad\overset{\substack{(4.6)\\(4.8)}}{<} \frac{\gamma}{8(h+v)} + \frac{\gamma}{8(h+v)} = \frac{\gamma}{4(h+v)},
\end{aligned}$$

which contradicts (4.5). Therefore, we conclude that the probability of choosing a $\mathit{hash} \in \mathcal{H}_{\mathit{Good}}$ amounts at least $\frac{\gamma}{8(h+v)}$.

We show that FindHash picks in one of its iteration $\mathit{hash} \in \mathcal{H}_{\mathit{Good}}$ 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 $\mathit{hash} \notin \mathcal{H}_{\mathit{Good}}$ is picked. We use $\Pr_{\mathit{hash} \leftarrow \mathcal{H}}[\mathit{hash} \in \mathcal{H}_{\mathit{Good}}] \geq \frac{\gamma}{8(h+v)}$ and $K \leq \frac{32(h+v)^2}{\gamma^2}n$, and conclude

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

It is clear that running time of FindHash is $\mathit{poly}(n,h,v,\gamma)$ with oracle access. This finishes the proof of Lemma 4.3. □

4.1.4 Amplification proof for partitioned domain

Let $C = (C_1, C_2)$ be a solver circuit for a dynamic weakly verifiable puzzle as in definition 3.1. We write $C_2^{(\cdot,\cdot)}$ to emphasize that C_2 does not obtain direct access to hint and verification circuits. Instead, whenever C_2 ask hint or verification queries, then it is answered explicitly as in the following code excerpt of the circuit $\widetilde{C}_2$.

```
Circuit $\widetilde{C}_2^{\Gamma_H, C_2, hash}(x, \rho)$
--------------------------------------------------------------------
Oracle: A hint circuit $\Gamma_H$, a circuit $C_2$,
        a function $hash : Q \to \{0, 1, \dots, 2(h+v) - 1\}$.
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 the 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 (q, y) such that $hash(q) = 0$ or $\perp$.

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

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

Lemma 4.4 *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 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

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

Lemma 4.5 ***(Security amplification for dynamic weakly verifiable puzzles with respect to hash.)*** *Let $g:\{0,1\}^k\to\{0,1\}$ be a monotone function and $P_n^{(1)}$ a fixed problem poser as in Definition 3.1 and $\widetilde{C}:=(C_1,\widetilde{C}_2)$ a circuit with oracle access to a function $hash:Q\to\{0,1,\ldots,2(h+v-1)\}$ and a solver circuit $C:=(C_1,C_2)$ for $P_{kn}^{(g)}$ which asks at most h hint queries and v verification queries. There exists an algorithm Gen that takes as input parameters ε, δ, n, k, has oracle access to $P_n^{(1)}$, $\widetilde{C}$, hash, g, and outputs a circuit $D:=(D_1,D_2)$ such that the following holds:*
If $\widetilde{C}$ 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

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

Furthermore, D asks at most $\frac{6k}{\varepsilon}\log\left(\frac{6k}{\varepsilon}\right)h$ hint queries and no verification queries. Finally, $Time(Gen)=poly(k,\frac{1}{\varepsilon},n)$ with oracle access to $\widetilde{C}$.

Before we give the proof of Lemma 4.5 we define additional algorithms. First, we are interested in the probability that for $u\leftarrow\mu_\delta^k$ and a bit b we have $g(b,u_2,\ldots,u_k)=1$. The estimate of this probability is calculated by 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 $k$, $\varepsilon$, $\delta$, $n$.
Output: An estimate $\widetilde{g}_b$ of $\Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]$.

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

Lemma 4.6 *The algorithm EstimateFunctionProbability$^g(b, k, \varepsilon, \delta, n)$ outputs an estimate $\widetilde{g}_b$ such that $|\widetilde{g}_b - \Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]| \leq \frac{\varepsilon}{8k}$ almost surely.*

Proof. We fix the notation as in the algorithm EstimateFunctionProbability. Let us define independent, identically distributed binary random variables $K_1, K_2, \ldots, K_N$ such that for each $1 \leq i \leq N$ the random variable K_i takes value g_i. We use the Chernoff bound to obtain

$$\Pr\Big[\Big|\widetilde{g}_b - \Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1]\Big| \geq \frac{\varepsilon}{8k}\Big]$$
$$= \Pr\Big[\Big|\Big(\frac{1}{N}\sum_{i=1}^{N} K_i\Big) - \mathbb{E}_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k)]\Big| \geq \frac{\varepsilon}{8k}\Big] \leq 2 \cdot e^{-n/3}. \square$$

The algorithm EvalutePuzzles$^{P^{(1)}, \widetilde{C}, hash}(\pi^{(k)}, \rho, n, k)$ evaluates which of the 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 circuit for the puzzle generated in the i-th round of the interaction between $P^{(g)}$ and $\widetilde{C}$. Therefore, the algorithm EvalutePuzzles runs k times $P^{(1)}$ to simulate the interaction with $C_1(\rho)$ each time with a fresh random bitstring $\pi_i \in \{0,1\}^n$ where $1 \leq i \leq k$.

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 sequential interaction. We use $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{P^{(1)}}$ to denote the output of $P^{(1)}(\pi_i)$ in the i-th round. Finally, we write $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{trans}$ to denote the transcript of communication in the i-th round. We note that the i-th round of the interaction between $P^{(1)}$ and C_1 is well defined only if all previous rounds have been executed before.

To make the notation easier in the code excerpts of circuits C_2, D_2 and EvalutePuzzles we omit superscripts of some oracles. Exemplary, we write $\widetilde{C}_2^{\Gamma_H^{(k)}, hash}$ instead of $\widetilde{C}_2^{\Gamma_H^{(k)}, C, hash}$ where the superscript of the oracle circuit C

is omitted. We make sure that it is clear from the context which oracles are used.

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

**Oracle:** A problem poser $P^{(1)}$, 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
    $(\Gamma_V^i, \Gamma_H^i) := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{P^{(1)}}$
    $x_i := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{trans}$
$x := (x_1, \ldots, x_k)$
$\Gamma_H^{(k)} := (\Gamma_H^1, \ldots, \Gamma_H^k)$
$(q, y_1, \ldots, y_k) := \widetilde{C}_2^{\Gamma_H^{(k)},hash}(x, \rho)$
if $(q, y_1, \ldots, y_k) = \bot$ then
    return $(0, \ldots, 0)$
$(c_1, \ldots, c_k) := (\Gamma_V^1(q, y_1), \ldots, \Gamma_V^k(q, y_k))$
return $(c_1, \ldots, c_k)$
```

All puzzles used by EvalutePuzzles are generated internally thus the algorithm has access to hint circuit, and can answer itself all queries of $\widetilde{C}_2$.

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

$$S_{\pi^*,b} = \Pr_{\pi^{(k)},\rho}[g(b, c_2, \ldots, c_k) = 1 \mid \pi_1 = \pi^*] - \Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \ldots, u_k) = 1], \tag{4.9}$$

where $(c_2, c_3, \ldots, c_k)$ is obtained as in EvalutePuzzles.

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 $P^{(g)}$, functions
            $g : \{0, 1\}^k \to \{0, 1\}$ and $hash : Q \to \{0, 1, \ldots, 2(h + v) - 1\}$.
**Input:** A bistring $\pi^* \in \{0, 1\}^n$, a bit $b \in \{0, 1\}$, parameters $k$, $\varepsilon$, $\delta$, $n$.
**Output:** An estimate $\widetilde{S}_{\pi^*,b}$ for $S_{\pi^*,b}$.
```

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

Lemma 4.7 *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 4.6 which yields that $\frac{1}{N}\sum_{i=1}^{N}\widetilde{s}^i_{\pi^*,b}$ differs from $\mathbb{E}[g(b, c_2, \ldots, c_k)]$ by at most $\frac{\varepsilon}{8k}$ almost surely. Together, with Lemma 4.6 we conclude that the surplus estimate returned by EstimateSurplus differs from $S_{\pi^*,b}$ by at most $\frac{\varepsilon}{4k}$ almost surely.□

We define the following solver circuit $C' = (C'_1, C'_2)$ for the $(k-1)$-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 $\Gamma_H^{(k-1)} := (\Gamma_H^2, \ldots, \Gamma_H^k)$,
        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: A transcript of $k-1$ rounds of interaction
        $x^{(k-1)} := (x_2, \ldots, x_k) \in \{0,1\}^*$, 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** $D_1^{\widetilde{C}}(r)$

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

Interact with the problem poser $\langle P^{(1)}, C_1(\rho)\rangle^1$.
Let $x^* := \langle P^{(1)}, C_1(\rho)\rangle^1_{trans}$.
```

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

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

**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 $\sigma$
    Use $x^*$ to simulate the first round of interiaction of $C_1(\rho)$ with the problem poser $P^{(1)}$
    **for** $i := 2$ **to** $k$ **do:**
        **run** $\langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i$
            $(\Gamma_V^i, \Gamma_H^i) := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{P^{(1)}}$
            $x_i := \langle P^{(1)}(\pi_i), C_1(\rho)\rangle^i_{trans}$
    $\Gamma_H^{(k)}(q) := (\Gamma_H(q), \Gamma_H^2(q), \dots, \Gamma_H^k(q))$
    $(q, y^*, y_2, \dots, y_k) := \widetilde{C}_2^{\Gamma_H^{(k)}, hash}((x^*, x_2, \dots, x_k), \rho)$
    $(c_2, \dots, c_k) := (\Gamma_V^2(q, y_2), \dots, \Gamma_V^k(q, y_k))$
    **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 $\text{Gen}^{P^{(1)},\widetilde{C},g,hash}(\varepsilon, \delta, n, k)$

Oracle: A poser $P^{(1)}$, a solver circuit $\widetilde{C}$ for $P^{(g)}$, functions $g : \{0,1\}^k \to$
$\{0,1\}$,
        $hash : Q \to \{0, 1, \ldots, 2(h+v)-1\}$.
Input: Parameters $\varepsilon$, $\delta$, $n$, $k$.
Output: A circuit $D$.

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

Proof (Lemma 4.5). 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 4.5 it follows that $\widetilde{C}$ succeeds with probability at least $\delta + \varepsilon$. Hence, D trivially satisfies the statement of Lemma 4.5. In the latter case g is a constant function, and the statement is vacuously true.

For the general case, we consider two possibilities. Either Gen in one of the iterations finds an estimate $\widetilde{S}_{\pi,b} \geq (1 - \frac{3}{4k}\varepsilon)$ or it fails and returns the circuit D.

In the former case we define a new monotone function $g'(b_2, \ldots, b_k) := g(b, b_2, \ldots, b_k)$ and a new circuit $\widetilde{C}' = (C_1', \widetilde{C}_2')$ with oracle access to $\widetilde{C} := (C_1, \widetilde{C}_2)$. By Lemma 4.7 it follows that $S_{\pi^*,b} \geq \widetilde{S}_{\pi^*,b} - \frac{\varepsilon}{4k} \geq (1 - \frac{1}{k})\varepsilon$ almost surely. Therefore, the circuit $\widetilde{C}'$ succeeds in solving the $(k-1)$-wise direct product of puzzles with probability at least $\Pr_{u \leftarrow \mu_\delta^{(k-1)}}[g'(u_1, \ldots, u_{k-1})] + (1 - \frac{1}{k})\varepsilon$. In this case $\widetilde{C}'$ satisfies the conditions of Lemma 4.5 for the $(k-1)$-wise direct product of puzzles. Therefore, the recursive call to Gen with access to g' and $\widetilde{C}$ returns a circuit $D = (D_1, D_2)$ that with high probability satisfies

$$\Pr_{\substack{\pi,\rho \\ x:=\langle P^{(1)}(\pi),D_1^{\widetilde{C}}(\rho)\rangle_{trans} \\ (\Gamma_H,\Gamma_V):=\langle P^{(1)}(\pi),D_1^{\widetilde{C}}(\rho)\rangle_{P^{(1)}}}}[\Gamma_V(D_2^{P^{(1)},\widetilde{C},hash,g,\Gamma_H}(x,\rho)) = 1] \geq \delta + \Big(1 - \frac{1}{k}\Big)\frac{\varepsilon}{6(k-1)} = \delta + \frac{\varepsilon}{6k}. \quad (4.10)$$

The only point remaining concerns the behavior of Gen when none of the estimates is greater than $(1 - \frac{3}{4k})\varepsilon$. Assume that...

TODO : Give more intuitoin (and the correct one with references to the equations)

Intuitively this means that $\widetilde{C}$ does not succeed 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 4.5 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 unusually often. It remains to prove that this intuition is indeed correct.

We fix the notation used in the code excerpt of the circuit D_2. We consider a single iteration of the outer loop of D_2, in which values $\pi_2, \dots, \pi_k$ are fixed. Additionally, we write π_1 to denote the randomness of the problem poser and define $c_1 := \Gamma_V(q, y_1)$, where Γ_V is the verification circuit generated by $P^{(1)}(\pi_1)$ in the first phase when interacting with $D_1(r)$. Finally, we introduce the additional notation $\mathcal{G}_b := \{(b_1, b_2, \dots, b_k) : g(b, b_2, \dots, b_k) = 1\}$ and $c = (c_1, c_2, \dots, c_k)$. Using the new notation the following holds

$$\begin{aligned}\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_b] &= \Pr_{u \leftarrow \mu_\delta^k}[g(b, u_2, \dots, u_k) = 1] \\ \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_b] &= \Pr_{\pi^{(k)},\rho}[g(b, c_2, \dots, c_k) = 1]. \qquad (4.11)\end{aligned}$$

We fix the randomness π_1 of the problem poser $P^{(1)}$ to π^* and use (4.9), (4.11) to obtain

$$\begin{aligned}&\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1] - \Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_0] \\ &\quad = \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_1 \mid \pi_1 = \pi^*] - \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_0 \mid \pi_1 = \pi^*] - (S_{\pi^*,1} - S_{\pi^*,0}) \qquad (4.12)\end{aligned}$$

We know that the function g is monotone, hence it holds $\mathcal{G}_0 \subseteq \mathcal{G}_1$, and we write (4.12) as

$$\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0] = \Pr_{\pi^{(k)},\rho}[c \in \mathcal{G}_1 \setminus \mathcal{G}_0 \mid \pi_1 = \pi^*] - (S_{\pi^*,1} - S_{\pi^*,0}). \qquad (4.13)$$

Still having $\pi_1 = \pi^*$ fixed we multiply both sides of (4.13) by

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

which yields

$$\begin{aligned}
&\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1]\\
&\quad=\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\frac{1}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\\
&\qquad-\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1](S_{\pi^*,1}-S_{\pi^*,0})\frac{1}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}.
\end{aligned}\tag{4.14}$$

We analyze the first summand of (4.14). First, we have

$$\begin{aligned}
&\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1]\\
&\quad=\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1|D_2(x^*,r)\neq\bot]\Pr_{\substack{r\\ x^*=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}}}[D_2(x^*,r)\neq\bot]\\
&\quad\overset{(*)}{=}\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\substack{r\\ x^*=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}}}[D_2(x^*,r)\neq\bot],
\end{aligned}\tag{4.15}$$

where in $(*)$ we use the observation that the event $D_2(x^*,r)\neq\bot$ happens if and only if the circuit $D_2(x^*,r)$ finds $\pi^{(k)}$ such that $c\in\mathcal{G}_1\setminus\mathcal{G}_0$. Furthermore, conditioned on $c\in\mathcal{G}_1\setminus\mathcal{G}_0$ we have that $\Gamma_V(D_2(x^*,r))=1$ occurs if and only if $c_1=1$. Inserting (4.15) to the numerator of the first summand of (4.14) yields

$$\begin{aligned}
&\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\\
&\quad=\Pr_{\substack{r\\ x^*=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}}}[D_2(x^*,r)\neq\bot]\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*].
\end{aligned}\tag{4.16}$$

We consider the following 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

$$\Pr_{\pi^{(k)},\rho}[c_1=1\mid c\in\mathcal{G}_1\setminus\mathcal{G}_0,\pi_1=\pi^*]\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]\leq\frac{\varepsilon}{6k}.\tag{4.17}$$

When $\Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0\mid\pi_1=\pi^*]>\frac{\varepsilon}{6k}$ the circuit D_2 outputs $\bot$ if and only if it fails in all $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})$ iterations to find $\pi^{(k)}$ such that $c\in\mathcal{G}_1\setminus\mathcal{G}_0$ which happens with probability

$$\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}}}[D_2(x^*,r)=\bot] \leq (1-\frac{\varepsilon}{6k})^{\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})} \leq \frac{\varepsilon}{6k}. \tag{4.18}$$

We conclude that in both cases by (4.17) and (4.18) we have

$$\begin{aligned}
&\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}}}[D_2(x^*,r)\neq\bot] \Pr_{\pi^{(k)},\rho}[c_1=1 \mid c\in\mathcal{G}_1\setminus\mathcal{G}_0, \pi_1=\pi^*] \Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0 \mid \pi_1=\pi^*]\\
&\geq \Pr_{\pi^{(k)},\rho}[c_1=1 \mid c\in\mathcal{G}_1\setminus\mathcal{G}_0, \pi_1=\pi^*] \Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_1\setminus\mathcal{G}_0 \mid \pi_1=\pi^*] - \frac{\varepsilon}{6k}\\
&= \Pr_{\pi^{(k)},\rho}[c_1=1 \land c\in\mathcal{G}_1\setminus\mathcal{G}_0 \mid \pi_1=\pi^*] - \frac{\varepsilon}{6k}\\
&= \Pr_{\pi^{(k)},\rho}[g(c)=1 \mid \pi_1=\pi^*] - \Pr_{\pi^{(k)},\rho}[c\in\mathcal{G}_0 \mid \pi_1=\pi^*] - \frac{\varepsilon}{6k}\\
&\overset{(4.9)}{=} \Pr_{\pi^{(k)},\rho}[g(c)=1 \mid \pi_1=\pi^*] - \Pr_{u\leftarrow\mu_\delta^{(k)}}[u\in\mathcal{G}_0] - S_{\pi^*,0} - \frac{\varepsilon}{6k}. \qquad (4.19)
\end{aligned}$$

We insert (4.19) to (4.14) and calculate the expected value of over π^* which yields

$$\begin{aligned}
\Pr_{\substack{\pi,r\\ x:=\langle P^{(1)}(\pi),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x,r))=1] &\geq \mathbb{E}_{\pi^*}\left[\frac{\Pr_{\pi^{(k)},\rho}[g(c)=1|\pi_1=\pi^*] - \Pr_{u\leftarrow\mu_\delta^{(k)}}[u\in\mathcal{G}_0] - \frac{\varepsilon}{6k}}{\Pr_{u\leftarrow\mu_\delta^{(k)}}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\right]\\
&- \mathbb{E}_{\pi^*}\left[\left(\Pr_{\substack{r\\ x^*:=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{trans}\\ (\Gamma_V,\Gamma_H):=\langle P^{(1)}(\pi^*),D_1(r)\rangle_{P^{(1)}}}}[\Gamma_V(D_2(x^*,r))=1](S_{\pi^*,1}-S_{\pi^*,0})+S_{\pi^*,0}\right)\frac{1}{\Pr_{u\leftarrow\mu_\delta^k}[u\in\mathcal{G}_1\setminus\mathcal{G}_0]}\right].
\end{aligned} \tag{4.20}$$

We show that if Gen does not recurse, then the majority of estimates is low almost surely. Let us assume that

$$\Pr_{\pi}\left[\left(S_{\pi,0}\leq(1-\frac{1}{2k})\varepsilon\right)\land\left(S_{\pi,1}\leq(1-\frac{1}{2k})\varepsilon\right)\right] < 1-\frac{\varepsilon}{6k}, \tag{4.21}$$

then Gen recurses almost surely, because the probability that Gen does not find $\widetilde{S}_{\pi,b}\geq(1-\frac{3}{4k})\varepsilon$ in all of the $\frac{6k}{\varepsilon}n$ iterations is at most

$$\left(1-\frac{\varepsilon}{6k}\right)^{\frac{6k}{\varepsilon}n} \leq e^{-n}$$

almost surely, where we used Lemma 4.7. Therefore, under the assumption that Gen does not recurse with high probability it holds

$$\Pr_{\pi,\rho}\left[\left(S_{\pi,0}\leq(1-\frac{1}{2k})\varepsilon\right)\land\left(S_{\pi,1}\leq(1-\frac{1}{2k})\varepsilon\right)\right] \geq 1-\frac{\varepsilon}{6k}. \tag{4.22}$$

Let us define a set

$$\mathcal{W} = \left\{ \pi : \left(S_{\pi,0} \le (1 - \frac{1}{2k})\varepsilon \right) \wedge \left(S_{\pi,1} \le (1 - \frac{1}{2k})\varepsilon \right) \right\}, \tag{4.23}$$

and use $\overline{\mathcal{W}}$ to denote the complement of $\mathcal{W}$. We bound the numerator of the second summand in (4.20)

$$\begin{aligned}
&\mathbb{E}_{\pi^*}[S_{\pi^*,0} + \Pr_{\substack{r \\ x^* := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{trans} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{P(1)}}} [\Gamma_V(D_2(x^*, r)) = 1](S_{\pi^*,1} - S_{\pi^*,0})] \\
&= \mathbb{E}_{\pi^*}\Big[S_{\pi^*,0} + \Pr_{\substack{r \\ x^* := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{trans} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{P(1)}}} [\Gamma_V(D_2(x^*, r) = 1](S_{\pi^*,1} - S_{\pi^*,0}) \mid \pi^* \in \overline{\mathcal{W}}] \Pr_{\pi^*}[\pi^* \in \overline{\mathcal{W}}\Big] \\
&\quad + \mathbb{E}_{\pi^*}\Big[S_{\pi^*,0} + \Pr_{\substack{r \\ x^* := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{trans} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{P(1)}}} [\Gamma_V(D_2(x^*, r)) = 1](S_{\pi^*,1} - S_{\pi^*,0}) \mid \pi^* \in \mathcal{W}] \Pr_{\pi^*}[\pi^* \in \mathcal{W}\Big] \\
&\le \frac{\varepsilon}{6k} + \mathbb{E}_{\pi^*}\Big[S_{\pi^*,0} + \Pr_{\substack{r \\ x := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{trans} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(r)\rangle_{P(1)}}} [\Gamma_V(D_2^{\widetilde{C}}(x^*, r)) = 1]((1 - \frac{1}{2k})\varepsilon - S_{\pi^*,0}) \mid \pi^* \in \mathcal{W}\Big] \Pr_{\pi^*}[\pi^* \in \mathcal{W}] \\
&\le \frac{\varepsilon}{6k} + (1 - \frac{1}{2k})\varepsilon = (1 - \frac{1}{3k})\varepsilon. \qquad (4.24)
\end{aligned}$$

We observe that

$$\begin{aligned}
\Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] &= \Pr[u \in \mathcal{G}_0 \vee (u \in \mathcal{G}_1 \setminus \mathcal{G}_0 \wedge u_1 = 1)] \\
&= \Pr[u \in \mathcal{G}_0] + \delta \Pr[u \in \mathcal{G}_1 \setminus \mathcal{G}_0]. \qquad (4.25)
\end{aligned}$$

Finally, we insert (4.24) into (4.20) which yields

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

From the assumptions of Lemma 4.5 it follows that

$$\Pr_{\pi^{(k)},\rho}[g(c) = 1] \ge \Pr_{u \leftarrow \mu_\delta^{(k)}}[g(u) = 1] + \varepsilon.$$

thus we get

$$\begin{aligned}
\Pr_{\substack{\pi^*, \rho \\ x := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{\text{trans}} \\ (\Gamma_V, \Gamma_H) := \langle P^{(1)}(\pi^*), D_1(\rho)\rangle_{P(1)}}} [\Gamma_V(D_2(x, \rho)) = 1] \quad &\ge \quad \frac{\Pr_{u \leftarrow \mu_\delta^k}[g(u) = 1] + \varepsilon - \Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_0] - (1 - \frac{1}{6k})\varepsilon}{\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0]} \\
&\overset{(4.25)}{\ge} \quad \frac{\varepsilon + \delta \Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0] - (1 - \frac{1}{6k})\varepsilon}{\Pr_{u \leftarrow \mu_\delta^k}[u \in \mathcal{G}_1 \setminus \mathcal{G}_0]} \ge \delta + \frac{\varepsilon}{6k} \\
&\qquad (4.26)
\end{aligned}$$

Clearly, the running time of Gen is $poly(k, \frac{1}{\varepsilon}, n)$. □

4.1.5 Putting it together

In the previous sections we have shown that it is possible to partition the domain Q such that if the number of hint and verification queries is small, then success probability of a solver for the k-wise direct product is still substantial. As shown in Lemma 4.5 we can build a circuit that returns a solution that is likely to be good. It remains to use these build blocks and prove Theorem 4.2.

Proof (of Theorem 4.2). Let $\widetilde{\text{Gen}}$ be the algorithm from Theorem 4.2, and $\widetilde{D} = (D_1, D_2)$ the corresponding solver circuit for P that is output by $\widetilde{\text{Gen}}$ as in Theorem 4.2. They are defined on the following code excerpts.

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

Oracle: A circuit $D := (D_1, \widetilde{D}_2)$ from Lemma 4.5, a problem poser $P^{(1)}$,
        functions $hash : Q \to \{0, 1, \ldots, 2(h+v)-1\}$, $g : \{0,1\}^k \to \{0,1\}$
        a verification oracle $\Gamma_V$, a hint oracle $\Gamma_H$.
Input: Bitstrings $x \in \{0,1\}^*$, $\rho \in \{0,1\}^*$.

$(q, y) := D_2^{P^{(1)},\widetilde{C},hash,g,\Gamma_H}(x, \rho)$
Ask verification query $(q, y)$ to $\Gamma_V$.
```

```
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 $n$, $\varepsilon$, $\delta$, $k$, $h$, $v$.
Return: A circuit $\widetilde{D} = (D_1, \widetilde{D}_2)$.

$hash :=$ FindHash($(h+v)\varepsilon, n, h, v$)
Let $\widetilde{C} := (C_1, \widetilde{C}_2)$ be as in Lemma 4.4 with oracle access to $C$, $hash$.
$D := Gen^{P^{(1)},\widetilde{C},g,hash}(\varepsilon, \delta, n, k)$
return $\widetilde{D} := (D_1, \widetilde{D}_2)$
```

We show that Theorem 4.2 follows from Lemma 4.3 and Lemma 4.5. We fix $P^{(1)}$, g, $P^{(g)}$ in the whole proof and consider a solver circuit $C = (C_1, C_2)$, asking at most h hint queries and v verification queries, such that

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

First, we note that C meets the requirements of Lemma 4.3. Thus, the algorithm $\widetilde{\text{Gen}}$ can call FindHash to obtain *hash* such that with high probability it holds

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

Applying Lemma 4.4 for C we obtain a circuit $\widetilde{C}=(C_1,\widetilde{C}_2)$ that satisfies

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

Now, we use the algorithm Gen as in Lemma 4.5 that yields a circuit $D=(D_1,D_2)$ which with high probability satisfies

$$\Pr_{\substack{\pi,\rho\\ x:=\langle P^{(1)}(\pi),D_1^{\widetilde{C}}(\rho)\rangle_{trans}\\ (\Gamma_H,\Gamma_V):=\langle P^{(1)}(\pi),D_1^{\widetilde{C}}(\rho)\rangle_{P(1)}}}[\Gamma_V(D_2^{P^{(1)},\widetilde{C},\mathit{hash},g,\Gamma_H}(x,\rho))=1]\geq(\delta+\frac{\varepsilon}{6k}).\tag{4.27}$$

Finally, $\widetilde{\text{Gen}}$ outputs $\widetilde{D}=(D_1,\widetilde{D}_2)$ with oracle access to D, $P^{(1)}$, *hash*, g such that with high probability it holds

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

The running time of FindHash is $\mathit{poly}(h,v,\frac{1}{\varepsilon},n)$ with oracle calls and of Gen $\mathit{poly}(k,\frac{1}{\varepsilon},n)$ with oracle access. Thus, the overall running time of $\widetilde{\mathit{Gen}}$ is $\mathit{poly}(k,\frac{1}{\varepsilon},h,v,n,t)$ with oracle access. Furthermore, the circuit $\widetilde{D}$ asks at most $\frac{6k}{\varepsilon}\log(\frac{6k}{\varepsilon})h$ hint queries and one verification query. Finally, we have $\mathit{Size}(\widetilde{D})\leq\mathit{Size}(C)\cdot\frac{6k}{\varepsilon}$. This finishes the proof of Theorem 4.2. □

4.2 Discussion

Appendix A

Appendix

A.1 Basic Inequalities

Lemma A.1 (Chernoff Bounds) *For independent, identically distributed Bernoulli random variables* $X_1, \ldots, X_n$ *with* $X := \sum_{i=1}^{n} X_i$ *with* $\Pr[X_i = 1] = p_i$ *and* $\Pr[X_i = 0] = 1 - p_i$ *for all* $1 \le i \le n$*. we have the following inequalities for* $0 \le \delta \le 1$ *and* $\mathbb{E}[X] = \sum_{i=1}^{n} p_i$*:*

$$\Pr[X \ge (1+\delta)\mathbb{E}[X]] \le e^{-\mathbb{E}[X]\delta^2/3} \tag{A.1}$$

$$\Pr[X \le (1-\delta)\mathbb{E}[X]] \le e^{-\mathbb{E}[X]\delta^2/2} \tag{A.2}$$

$$\Pr[|X - \mathbb{E}[X]| \ge \delta\mathbb{E}[X]] \le 2e^{-\mathbb{E}[X]\delta^2/3}. \tag{A.3}$$

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[CHS04] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. 2004.

[CW77] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, STOC '77, pages 106–112, New York, NY, USA, 1977. ACM.

[DIJK09] Yevgeniy Dodis, Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Security amplification for interactive cryptographic primitives. In *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, TCC '09, pages 128–145, Berlin, Heidelberg, 2009. Springer-Verlag.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, New York, NY, USA, 2004.

[Hol13a] Thomas Holenstein. Complexity theoretic cryptography, Spring 2013.

[Hol13b] Thomas Holenstein. Lecture notes in complexity theory, Spring 2013.

[HS10] Thomas Holenstein and Grant Schoenebeck. General hardness amplification of predicates and puzzles. *CoRR*, abs/1002.3534, 2010.

[Mau13] Ueli Maurer. Lecture notes in cryptography, Spring 2013.

[VABHL03] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *Advances in Cryptology—EUROCRYPT 2003*, pages 294–311. Springer, 2003.