國立臺灣大學電機資訊學院資訊網路與多媒體研究所

碩士論文

Graduate Institute of Networking and Multimedia

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

打造公平的遊戲轉蛋：在不洩漏原始碼的前提下驗證虛擬轉蛋的機率

Securing Fair Loot Box Games: An Efficient Approach to Verifying Loot Box Probability Statement Without Source Code Disclosure

王靖傑

Jing-Jie Wang

指導教授: 蕭旭君 博士

Advisor: Hsu-Chun Hsiao Ph.D.

中華民國 112 年 6 月

June, 2023

doi:10.6342/NTU202301027

國立臺灣大學碩士學位論文
口試委員會審定書

打造公平的遊戲轉蛋:在不洩漏原始碼的前提下驗證虛擬轉蛋的機率

Securing Fair Loot Box Games: An Efficient Approach to Verifying Loot Box Probability Statement Without Source Code Disclosure

本論文係王靖傑君（學號 R10944074）在國立臺灣大學資訊網路與多媒體研究所完成之碩士學位論文，於民國一百一十二年六月十四日承下列考試委員審查通過及口試及格，特此證明。

口試委員：

（簽名）

（指導教授）

所　長：

摘要

虛擬轉蛋是一種隨機獲取獎勵的機制，並且已經是近期遊戲產業主要的盈利模式。然而，世界各地發生過許多起針對虛擬轉蛋消費糾紛事件，例如近期台灣發生的「丁特 v.s. 天堂 M」事件，玩家實測出的掉寶機率與遊戲公告的機率明顯不符，後續公平交易委員會也證實遊戲公司確實有廣告不實。面對這些事件，許多國家制定法律，要求遊戲公司需明確揭露虛擬轉蛋中獎機率。然而，目前並沒有一個公認可行的方式來驗證虛擬轉蛋的實際中獎機率，其中一些方法並無法完整驗證資料，一些方法要求將程式碼開源。但考量到現實情況，要求遊戲公司開源是不實際的，因為轉蛋演算法往往是重要的商業機密。於是本篇研究提出在不洩漏原始碼的前提下驗證虛擬轉蛋中獎機率的方法。我們提出的方法包括兩種協定：機率驗證協定以及轉蛋抽獎協定。機率驗證協定使用隨機信標及函數承諾等密碼學工具，提供不可預測的隨機源及執行正確性的驗證，以此獲得測試資料進行機率驗證；轉蛋抽獎協定讓讓玩家及遊戲伺服器共同產生一個隨機參數，使得雙方都不能預測或控制轉蛋的機率，保障玩家在抽獎過程並沒有被降低機率。此外，我們也實作提出的兩種協定，並對於不同參數進行實驗比較。結果顯示我們的系統擁有良好的效能及實用性。

關鍵字：資訊安全、密碼學、虛擬轉蛋

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

Abstract

Loot boxes, which have become a predominant revenue model in contemporary mobile games, offer players the opportunity to acquire random rewards. However, there have been numerous incidents of disputes related to loot box consumption around the world. For example, in the recent "Dinter v.s. Lineage M" incident in Taiwan, where players conducted tests and found that the winning probability did not align with the probability stated by the game company. The Fair Trade Commission later confirmed that the game company had engaged in false advertising. Faced with these incidents, some countries have adopted laws mandating the explicit disclosure of probabilities. However, currently, there is no practical method to verify the probability statements. As a result, in this work, we propose an efficient approach for verifying the probability of loot boxes without disclosing the source code. In particular, we propose two protocols: a probability verification protocol and a loot box opening protocol. The probability verification protocol allows players to verify the winning probability of loot boxes using publicly verifiable random

doi:10.6342/NTU202301027

sources. On the other hand, the loot box opening protocol establishes a mechanism for the game servers and players to agree on a random input, ensuring that neither party can manipulate the outcome. We also implemented our protocols and conducted experiments to evaluate their performance. The results demonstrate that these protocols are efficient and practical.

Keywords: security, cryptography, loot box

doi:10.6342/NTU202301027

Contents

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

Chapter 1 Introduction

1.1 What is a loot box

Loot boxes are commonly utilized in the game industry to describe a mechanism where players spend in-game currency to receive a random in-game item. Currently, the majority of games that incorporate loot box mechanisms are mobile freemium games, allowing players to use them at no cost. However, fees will be charged if players require supplementary services, such as obtaining rare items. The loot box business model has gained significant popularity in recent years and currently dominates the gaming industry. In 2020, free-to-play games were responsible for generating 78% of the total revenue in the gaming industry, as reported by SuperData [32].

Previous research has demonstrated a strong correlation between engagement in loot boxes and the severity of problem gambling, which is characterized by the extent to which an individual is affected by addictive or problematic gambling behavior [15] [33] [8] [25]. This problem is even more significant among adolescent players, as suggested by [23]. Besides pointing out the solid correlation for problem gambling, certain studies have suggested that regulations should be implemented to mandate game companies to disclose the probability of obtaining rare items in loot boxes [36].

doi:10.6342/NTU202301027

Several countries, game industries, and platforms have already published regulations regarding loot boxes, either through legal power or self-regulation. In Section 2, we will examine these regulations in greater detail.

1.2 Current verification of loot box probability statement

Although there are already some laws and regulations requiring game companies to disclose probabilities explicitly, verifying probability statements remains a challenging issue.

Recently, the Taiwanese government, collaborating with NTUST Gamelab, published a third-party verification website [19]. Game players can upload their loot box opening records on this platform, along with a screen recording video that provides proof of authenticity for the record. Once a sufficient number of loot box opening records have been collected, the platform can display the validity of probability statements with different confidence intervals.

However, using this third-party verification platform has the following issues:

1. Proving the authenticity of records can be challenging and expensive. The platform utilizes screen recording videos as a method to ensure record authenticity, but this approach is costly and does not guarantee 100% accuracy of the data.

2. The platform does not offer a means for the public to authenticate the data. While the platform claims to have a dedicated team responsible for data verification, the lack of transparency leaves us with no choice but to trust the platform, which is not ideal.

doi:10.6342/NTU202301027

3. Detecting hidden inputs is not possible. Given the doubts surrounding the treatment of different players with varying winning probabilities, such as the case described in [26], it is crucial to confirm that there are no hidden inputs in the loot box algorithm.

4. There is a possibility of selective data uploading. For instance, game companies may select and upload only winning records from a large sample of opening records, leading to a biased probability.

On the other hand, Carvalho proposed a transparent loot box scheme, which utilizes blockchain and smart contracts [9]. However, this approach necessitates game companies to implement their loot box function using smart contracts, resulting in the disclosure of the function's source code once it is uploaded to the blockchain. In addition, this approach uses the timestamp as its random source, which is predictable to the players. Consequently, players can exploit this predictability to increase their chances of winning.

Ideally, if game companies are willing to make their loot box algorithm open-source, curious players would have the opportunity to audit the source code or manually run it to verify if the actual probabilities match the stated probabilities. However, in reality, since loot box algorithms are considered trade secrets by game companies, they often take measures to protect and conceal the source code from the public. Moreover, it is almost impossible to compel game companies to open-source their loot box algorithms from a regulatory standpoint.

As a result, our main goal is to enable players to verify the loot box probability with minimal information disclosure of the underlying opening function. Additionally, it is crucial to ensure that the game company cannot manipulate the randomness used in the loot box opening process, in order to prevent intentional decreases in the winning probability

doi:10.6342/NTU202301027

of certain players.

In this work, we propose two protocols: the probability verification protocol and the loot box opening protocol. The probability verification protocol allows players to verify the winning probability of loot boxes using publicly verifiable random sources. On the other hand, the loot box opening protocol establishes a mechanism for the game servers and players to agree on a random input, ensuring that neither party can manipulate the outcome.

This paper makes the following contributions:

1. We investigated the problem of loot box probability disclosure, and examine current regulations surrounding loot boxes.
2. We propose the probability verification protocol and the loot box opening protocol as solutions to verify the loot box probability statement without disclosing the source code.
3. We optimize the HeadStart beacon by using a cryptographic accumulator, which reduces the complexity of verification to constant with respect to the number of contributions.
4. We implement and evaluate the probability verification protocol and the loot box opening protocol, showing that our protocols are efficient and practical.

doi:10.6342/NTU202301027

Chapter 2 Background

In this section, we will cover the following topics. Firstly, we will examine the different regulations on loot boxes worldwide, with a particular focus on the new "gacha regulation" recently implemented in Taiwan in 2023. Secondly, we will provide a high-level introduction to the public randomness beacon (PRB). Lastly, we will introduce a new cryptographic primitive called functional commitment (FC).

2.1 Regulations on loot box around the world

As of now, several countries have implemented regulations concerning loot boxes. However, the existing regulatory approaches differ significantly. Certain countries classify loot boxes as a form of gambling, while others only mandate the disclosure of probabilities [38].

Belgium In 2018, the Belgian Gaming Commission stated that loot box mechanisms are considered gambling under current legislation and has effectively prohibited loot boxes by enforcing gambling laws. Game companies that implement paid loot boxes without a gambling license may face criminal prosecution. However, a recent study showed that 82% among the 100 highest-grossing Belgian iPhone games had loot box features [37].

doi:10.6342/NTU202301027

China In 2017, the People's Republic of China (PRC) is the first country that legally mandates game companies to disclose the probabilities of receiving randomized loot box rewards [39].

Japan In 2012, the Consumer Affairs Agency (CAA) in Japan banned the "complete gacha", which involves the player needing to collect a series of items before having the chance to obtain a specific rare item. The requirement for disclosing probabilities is achieved through self-regulation in Japan. In 2016, the Computer Entertainment Supplier's Association (CESA) announced a guideline for gacha probability and pricing, which obligated its members to comply with these guidelines. A study on Japanese game players' attitudes towards loot box probability statements [21] revealed that despite regulations and self-regulating guidelines, the majority of players did not trust the probability announcement.

Netherlands In 2018, a report by the Netherlands Gambling Authority [3] found that loot boxes offering tradable items are illegal and, hence, the underlying games cannot be sold without an appropriate license.

South Korea South Korea has previously adopted a self-regulatory approach to loot box games through the leadership of the Korean Association of the Game Industry (K-GAMES). However, recently in February 2023, the National Assembly of South Korea collaborated with K-GAMES to pass a new amendment that mandates loot box probability disclosure [28].

Apart from national regulations on loot box games, Apple app store and Google play

doi:10.6342/NTU202301027

store and also have some policies on loot box games.

Apple App Store In 2017, Apple changed their developer guidelines. According to App Store Review Guidelines, "Apps offering loot boxes or other mechanisms that provide randomized virtual items for purchase must disclose the odds of receiving each type of item to customers prior to purchase." [2].

Google Play Store In 2019, Google announced their new policy to loot box, stating that "Apps and games offering mechanisms to receive randomized virtual items from a purchase including, but not limited to, loot boxes must clearly disclose the odds of receiving those items in advance of, and in close and timely proximity to, that purchase." [20].

2.2 Regulation on loot box in Taiwan

In June 2021, an online player named Paul started a petition on a public policy participation platform to promote the Taiwanese government to formulate regulations on "Gacha" mechanics. The petition received support from 6,560 people [29]. Additionally, there have been several incidents of unfair loot box practices. In the following, we will discuss two such incidents.

In September 2021, a popular Taiwanese streamer named Dinter accused the game company behind "Lineage M" of manipulating the probability of loot boxes in the online game [1]. Prior to this incident, the game company had stated that the probability of obtaining a particular item in the Taiwanese version was equivalent to that of the Korean version, which was 10%. However, despite spending more than 4 million TWD, Dinter

doi:10.6342/NTU202301027

only succeeded 11 times out of 475 attempts, resulting in a success rate of approximately 2.3%.

Upon receiving Dinter's appeal, the Fair Trade Commission initiated an investigation. The investigation revealed that, based on the game company's internal communication records, the actual probability of obtaining the item in question was 5%, which was clearly inconsistent with their previous claim. Consequently, the commission imposed a fine of 2 million TWD on the game company, citing violation of the Fair Trade Act. This incident also promoted public discussion on the revision of loot box regulation.

In addition to Dinter's incident, an anonymous player of "Arena of Valor" claimed to be an employee of the game's company and revealed that the probability of obtaining items from loot boxes differed for each player, depending on how much they had spent in the game [26]. The player also presented modified source code as evidence to support his claim. The anonymous player's disclosure led to widespread discussion and highlighted the growing distrust between players and the game company.

Following the incidents and extensive discussions between players, game companies, and the government, the Consumer Protection Committee announced revisions to regulations in 2022. The revisions included the following points [14]:

1. Game companies should disclose the probability of randomized virtual item
2. The winning probability should be clearly defined
3. The scope of winning probability disclosure should be clearly defined
4. The manner in which winning probability is disclosed should be clearly defined

After discussing the regulations on loot boxes and considering recent events world-

doi:10.6342/NTU202301027

wide, we will now explore two important cryptographic tools that are critical building blocks of our protocol.

2.3 Public randomness beacon (PRB)

A public randomness beacon is a service that produces and publishes **unpredictable** and **bias-resistant** randomness at fixed intervals. The main goal of a randomness beacon is to provide a reliable source of randomness that cannot be predicted or manipulated by any party, including the service provider.

We surveyed several publicly-verifiable randomness beacon constructions, including public verifiable secret sharing (PVSS) based [10] [31] [11], verifiable random function (VRF) based [18], BFT state machine replication based [4], homomorphic encryption based [12] and verifiable delay function (VDF) based [30] [24]. We opted to use the HeadStart randomness beacon [24] because it is a participatory scheme, enabling the public to contribute their randomness and verify its inclusion in the process. We also made some modifications to reduce the complexity regarding the number of contributions, which is better suited to the requirements of the loot box scenario. Furthermore, a HeadStart randomness beacon can be utilized as a shared source of randomness across various loot box games. This enhances usability, as players need only contribute once and can use the following randomness for multiple loot box games.

PRB can be utilized as a random source to generate test data for loot box probability verification. The main benefit is that it guarantees unpredictable and bias-resistance, which means the game companies cannot manipulate or predict the test data to bias the winning probability of the underlying function.

doi:10.6342/NTU202301027

2.4 Functional Commitment

A functional commitment scheme allows a committer to make a commitment to a secret function f, and subsequently prove that $y = f(x)$ for public x and y while keeping all other details about f concealed [7]. In the loot box scenario, functional commitment makes it possible for game companies to demonstrate that the loot box opening procedure is being applied uniformly and correctly for all individuals.

In 2021, Boneh et al. proposed an efficient functional commitment construction. This functional commitment is constructed using some special kinds of zk-SNARKs (Marlin [13] and Plonk [17]) as a building block and is extended with a "proof of function relation" to create a succinct functional commitment. We provide a formal definition of this construction in more detail in section 3.4.

In addition to its succinctness, functional commitment can compel game companies to disclose all the input parameters used in the loot box algorithm, and guarantees that each player is treated fairly with the same algorithm. This feature addresses the concerns raised by players about the game company potentially using different algorithms for Korean and Taiwanese players [1], as well as the controversy that the game company was secretly using players' consumption amounts as a hidden input [26].

doi:10.6342/NTU202301027

Chapter 3 Preliminaries

In this section, we will introduce the cryptographic primitives that we utilize in our loot box verification protocols. We first introduce the verifiable delay function (VDF) and cryptographic accumulator. Then, we define the notion of public randomness beacon (PRB). Finally, we describe how we leverage a cryptographic accumulator to improve one of the PRB construction known as HeadStart.

3.1 Verifiable Delay Function

The Verifiable Delay Function (VDF) was formally defined in [5]. It is a cryptographic function that is moderately difficult to compute, yet its results can be efficiently verified. A VDF consists of the following three algorithms:

- $\texttt{VDF.Setup}(1^\lambda, T) \to \mathbf{pp}$: Given the security parameter 1^λ and a time parameter T, outputs public parameters **pp**. This should be a randomized algorithm.
- $\texttt{VDF.Eval}(x) \to (y, \pi)$: Given an input x, outputs the result y along with a proof π and must run in parallel time T. This should be a deterministic algorithm.
- $\texttt{VDF.Verify}(x, y, \pi) \to \{0, 1\}$: Given input x, claimed output evaluation value y, and proof π, outputs decision bit $\{0, 1\}$. The Verify algorithm, which is much faster

doi:10.6342/NTU202301027

than Eval, must run in total time log (T).

In addition, a VDF must satisfy the following properties:

1. **Correctness**: The algorithm `VDF.Verify`(x, y) must return 1 if (y, π) is actually generated by `VDF.Eval`(x).

2. **Soundness**: For any polynomial time adversary $\mathcal{A}$ who generates (x, y, π), the probability of `VDF.Verify`$(x, y, \pi) = 1$ but $y \neq$ `VDF.Eval`(x) is negligible.

3. σ-**sequentiality**: There is no randomized algorithm with $poly(T, \lambda)$ parallel processors can evaluate `VDF.Eval` with non-negligible probability before time $\sigma(T)$.

With σ-sequentiality, we can ensure that no one is able to obtain the result of `Eval` before $\sigma(T)$ from the start. Although a perfect VDF would achieve $\sigma(T) = T$, such a requirement is unrealistic. In practice, it suffices to use a VDF that ensures $\sigma(T)$ is approximately $T - o(T)$, or even $\sigma(T) = T - \epsilon T$ with a small ϵ.

In practice, we choose to use Wesolowski's VDF [35], which is based on the hidden order group. More precisely, this VDF construction can be reduced to "adaptive root assumption", which means that the adversary cannot compute the order of **any** non-trivial element in the group.

3.2 Cryptographic Accumulator

A cryptographic accumulator scheme generates a value that accumulates a set of values and creates a proof for each value in the set. With the accumulated value and the proof, anyone can verify if a particular value belongs to the set.

doi:10.6342/NTU202301027

Formally speaking, for a set of n values $\mathbb{X} = \{x_1, ..., x_n\}$, a cryptographic accumulator consists of the following algorithms:

- $\texttt{ACC.Setup}(1^\lambda, n) \rightarrow \mathbf{pp}$: Given the security parameter λ and upper limit of set size, return public parameters **pp**.

- $\texttt{ACC.Eval}(\mathbf{pp}, \mathbb{X}) \rightarrow x_{acc}$: Given public parameters and a set $\mathbb{X}$, returns an accumulated value x_{acc} for the set $\mathbb{X}$.

- $\texttt{ACC.CreateWit}(\mathbf{pp}, x_{acc}, x_i, \mathbb{X}) \rightarrow w_i$: Given public parameters, accumulated value x_{acc}, a set $\mathbb{X}$ and an element x_i in $\mathbb{X}$, returns a witness w_i if $x_i \in \mathbb{X}$, otherwise, returns $\perp$.

- $\texttt{ACC.Verify}(\mathbf{pp}, x_{acc}, x_i, w_i) \rightarrow \{0, 1\}$: Given public parameters, accumulated value x_{acc}, a value x_i, and a witness w_i, returns 1 if x_i is indeed accumulated in x_{acc}, otherwise, returns 0.

Several accumulator schemes have been proposed, and the Merkle tree is one of the simplest among them [27]. However, when the size of the underlying set is large, the verification complexity of the Merkle tree's inclusion proof, which is $O(\log n)$, may not be ideal. On the other hand, a new accumulator construction using a hidden order group can achieve a verification complexity of $O(1)$ [6]. Since this construction uses the same technique introduced by Wesolowski [35], it also needs the "adaptive root assumption" of the underlying group.

doi:10.6342/NTU202301027

3.3 Public Randomness Beacon

A public randomness beacon (PRB) is a service that generates and publishes unpredictable, bias-resistant, and publicly verifiable random values at regular intervals.

For ease of describing the use of PRB in the following sections, we model PRB as follows:

- `PRB.Setup`$(1^\lambda, I) \rightarrow$ **pp**: Given the security parameter λ and beacon interval I, outputs public parameters **pp**. This should be a randomized algorithm.
- `PRB.Contribute`(**pp**, x): Given a public parameter **pp**, and a local randomness x, this algorithm inserts x to the upcoming randomness beacon outputs.
- `PRB.Eval`$(\textbf{pp}, \{x_1, ..., x_n\}) \rightarrow r$: Given **pp**, a set of randomness contributions $\{x_1, ..., x_n\}$, outputs r as the outcome of this beacon interval.
- `PRB.Verify`$(\textbf{pp}, x_i, r) \rightarrow \{0, 1\}$: Given **pp**, a randomness contribution x_i and randomness outcome r, outputs decision bit $\{0, 1\}$.

In addition, a public randomness beacon should possess the following properties:

1. **Unpredictability**: Before `PRB.Eval` returns, it is computationally infeasible to predict the output of `PRB.Eval`
2. **Bias-Resistance**: There is no adversary that can manipulate the output of `PRB.Eval` for its own advantage.
3. **Verifiability**: Honest contributors can verify the output is unpredictable and bias-resistant.

doi:10.6342/NTU202301027

As we surveyed in section 2.3, there are many constructions of PRB with different assumptions and different cryptographic primitives. In particular, we choose the HeadStart beacon [24], which allows any contributors to verify the unpredictability and boas-resistance without other assumptions. We also make some improvement to reduce the complexity regarding the number of contribution. In the origin HeadStart beacon, the verification complexity is $O(L \times polylog(T) + \log C)$, where L is the number of beacon outcomes after your contribution and T is the period of each beacon outcome. However, in the loot box scenario, the number of contributions may be quite large since there might be various loot box events, and we can also use the same PRB across different games. We further reduce the complexity to $O(L \times polylog(T))$ by using cryptographic accumulator to replace the Merkle tree used in HeadStart beacon.

3.4 Functional Commitment

As defined in [7], a functional commitment scheme enables a committer to commit to a secret function and provides proofs of evaluations of that function at specific points.

In this paper, we briefly revise the notation used in [7], and make the evaluation protocol non-interactive using the Fiat-Shamir heuristic.

- $\texttt{FC.Setup}(1^\lambda, N) \to \mathbf{pp}$: Given the security parameter λ and upper limit of gates number N, outputs public parameters $\mathbf{pp}$. This should be a randomized algorithm.

- $\texttt{FC.Commit}(\mathbf{pp}, f, r) \to c$: Given a public parameter $\mathbf{pp}$, a secret function f, and randomness r, outputs a commitment c to f. This should be a deterministic algorithm.

doi:10.6342/NTU202301027

- $\texttt{FC.Eval}(\textbf{pp}, f, r, x, y) \to \pi$: Given **pp**, a function f, another randomness r, evaluation point x, and claimed evaluation value y, outputs a proof π that convince verifier that $f(x) = y$.

- $\texttt{FC.Verify}(\textbf{pp}, c, x, y, \pi) \to \{0, 1\}$: Given **pp**, a function f, a commitment c, evaluation point x, claimed evaluation value y, and a proof π, outputs decision bit $\{0, 1\}$.

Following the definition in [7], we briefly restate the notation and switch to the non-interactive version of functional commitment.

A **secure** functional commitment should possess the following properties:

1. **Binding**: It is computationally infeasible to find distinct functions f_1, f_2 such that $\texttt{FC.Commit}(\textbf{pp}, f_1, r_1) = \texttt{FC.Commit}(\textbf{pp}, f_2, r_2)$ for some **pp**, r_1 and r_2.

2. **Hiding**: For commitments c_1, c_2 derived from two distinct functions, it is computationally indistinguishable between c_1 and c_2.

3. **Completeness**: For all commitment c generated by `FC.Eval`, the verification `FC.Verify` always return 1.

4. **Evaluation zero-knowledge**: The proof π reveals nothing other than the evaluation $f(x) = y$.

5. **Knowledge soundness**: A valid evaluation proof can only be produced by provers possessing knowledge of the secret function f.

6. **Evaluation Binding**: It is computationally infeasible to construct valid evaluation proofs for different evaluation values $y_1 \neq y_2$ on the same input x.

doi:10.6342/NTU202301027

Boneh et al. [13] introduced two constructions of general arithmetic circuit functional commitment from preprocessing zk-SNARK [7], namely Marlin and Plonk [17]. In our scenario, the **Hiding** property is the most important since it ensures that the functional commitment scheme does not disclose any information regarding the secret function, which is frequently considered a trade secret.

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

Chapter 4 Problem Formulation

This section formally defines the abstraction of our problem, along with the assumptions we made and the threat model. After that, we formally define the objectives of our protocols.

We denote the loot box opening function as $f(r, \texttt{others})$, where r represents the random source and `others` denotes the input parameters excluding the random source.

Formally speaking, let R be the random space, $\mathbb{O}$ be the domain space for `others`, and {0,1} denote the event of a player winning the loot box. The loot box opening function is defined by

$$f : R \times \mathbb{O} \to \{0, 1\}$$

When the game company stated that their winning probability is p_0, we test whether the probability is **greater than** p_0, for both the domain R and $\mathbb{D}$.

$$\Pr\left[f(r, \texttt{others}) = 1 \,\middle|\, \begin{array}{l} r \in R \\ \\ \texttt{others} \in \mathbb{O} \end{array} \right] \geq p_0$$

Specific loot box mechanisms may have different probabilities for different `others` input parameters. In this case, we should conduct separate testing for each input param-

doi:10.6342/NTU202301027

eter. For example, some games with a "guarantee mechanism" use `count` as an input parameter representing the number of attempts made by the player. As `count` gets higher, the probability increases linearly to 100% when it reaches the "guaranteed count". For such a loot box design, we can do the probability testing for each `count` to verify whether the actual probability is equal to or higher than the claimed probability.

In the following context, we refer to the game player as the client and the game company as the server.

4.1 Assumptions

In our protocols, we made the following assumptions:

- The server does not abort, even though it knows the outcome before responding to the client.
- There is a publicly accessible append-only bulletin board that allows everyone to read and write data.
- Neither the server nor the client can predict or bias the randomness sampled by the other party.
- There is an authenticated communication channel between the server and the client
- There is at least one honest contributor in the public randomness beacon.

doi:10.6342/NTU202301027

4.2 Threat Model

In the probability verification protocol, the adversary's objective is to deceive the client by announcing a false probability statement higher than the actual probability. The adversary can also contribute to the public randomness beacon, but they cannot manipulate or predict the randomness contribution of all the clients.

On the other hand, in the loot box opening protocol, the adversary's objective is to decrease the winning probabilities of the clients. They may attempt to choose from different setups but cannot manipulate or predict the randomness generated by the clients.

Since we assume an authenticated communication channel between the server and the client, all the messages are non-repudiable. This means that if the server responds with a fake result and the client fails to verify the result, a third party can verify that the server is indeed sending a fake response.

4.3 Objectives

The objectives of our protocol are as follows:

1. **Correctness & Soundness**: The probability verification protocol succeeded if and only if the actual winning probability of f is greater than or equal to p.
2. **Public Verifiability**: The probability verification protocol allows anyone to verify the correctness of the probability statement.
3. **Individual Verifiability**: The loot box opening protocol allows the client to verify that the winning probability is not biased by the server.

doi:10.6342/NTU202301027

4. **Input Transparency**: The opening function f should use only transparent input parameters, meaning that there should be no hidden inputs involved.

5. **Algorithmic Hiding**: The protocol should only reveal the evaluation points of f and should not disclose any other information about f.

doi:10.6342/NTU202301027

Chapter 5 Proposed Protocols

This section proposes two protocols: a probability verification protocol and a loot box opening protocol. The probability verification protocol allows players to verify the winning probability of loot boxes using publicly variable random sources. On the other hand, the loot box opening protocol establishes a mechanism for the game servers and players to agree on a randomness input, ensuring that neither party can manipulate the outcome.

5.1 Probability Verification Protocol

We have proposed four methods for verifying the loot box probability statement. Each method can meet different criteria for verifiability, depending on the assumptions and trade-offs involved.

5.1.1 Probability Verification Through Public Randomness Beacon

In this verification method, the public randomness beacon serves as the source of randomness to generate test data. Players who are interested or skeptical about the test data can provide their own random source to the public randomness beacon and then verify it.

doi:10.6342/NTU202301027

We have divided this verification process into four phases, and we describe each phase in detail below. For a summary, please refer to Figure 5.4.

Setup In this phase, the server first runs the setup algorithm for each cryptographic primitive, including `PRB.Setup` and `FC.Setup`. Secondly, the server designates the number of beacon intervals, denoted by n, indicating that it will utilize the random output generated after n intervals from the moment it publishes the commitment. Thirdly, the server designates how the randomness is mapped to test data, denoted as mapping function M. This mapping function takes public randomness r as input and generates the corresponding test data as output. Formally,

$$M(r) \to \{(r_1, o_1), ..., (r_m, o_m)\}$$

where $r_i \in R$ and $o_i \in \mathbb{O}$ for $1 \leq i \leq m$.

Last, the server commits the loot box opening function f to a commitment c by $\texttt{FC.Commit}(\mathbf{pp}, f, r) \to c$, where **pp** is generated by `FC.Setup` and r is sampled from random. Subsequently, the server publishes the public parameters, the interval number n, the mapping function M, and the commitment c on the bulletin board. For a concise flowchart, please refer to Fig. 5.1.

Randomness Contribution In this phase, the clients examine the validity of the setup data published by the server, including public parameters, n, M, and c. Then, interested or skeptical clients can sample their local randomness and contribute through `PRB.Contribute`. It is important to note that clients who have already contributed their local randomness can skip this step, as the PRB protocol ensures unpredictability and bias-resistance for such

doi:10.6342/NTU202301027

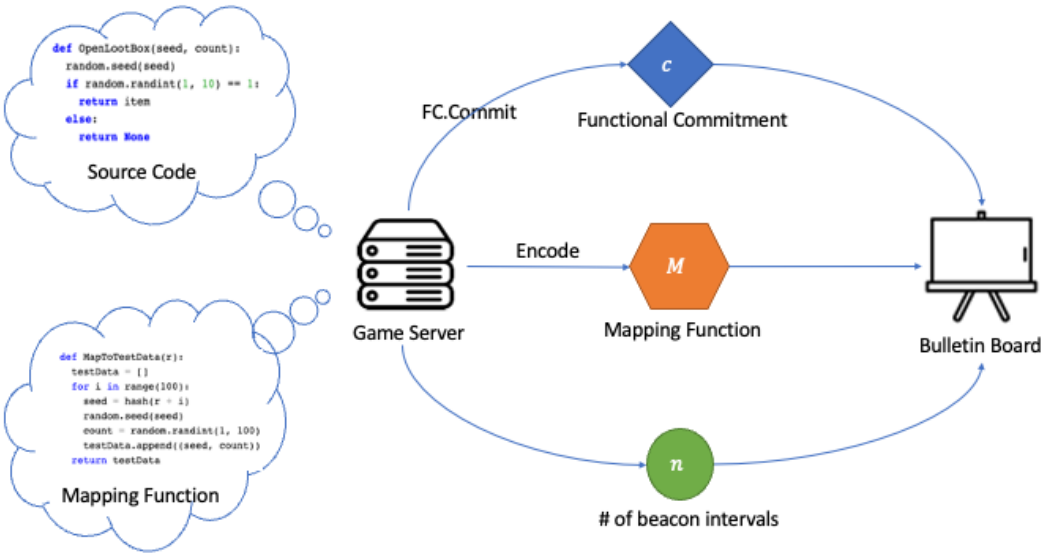

Figure 5.1: The flowchart of setup phase in probability verification protocol

clients. We illustrate this phase in Fig. 5.2.

Evaluation In this phase, the server first gets the randomness outcome r of the n^{th} randomness beacon counted from the setup phase. Secondly, the server uses the mapping function M to obtain test data, i.e.

$$M(r) \to \{(r_1, o_1), ..., (r_m, o_m)\}$$

where $r_i \in R$ and $o_i \in \mathbb{O}$ for $1 \leq i \leq m$.

Then, the server computes the outputs of the function f and evaluation proofs

$$f(r_i, o_i) \to y_i$$

$$\texttt{FC.Eval}(\mathbf{pp}, f, (r_i, o_i), y_i) \to \pi_i$$

for $1 \leq i \leq m$.

doi:10.6342/NTU202301027

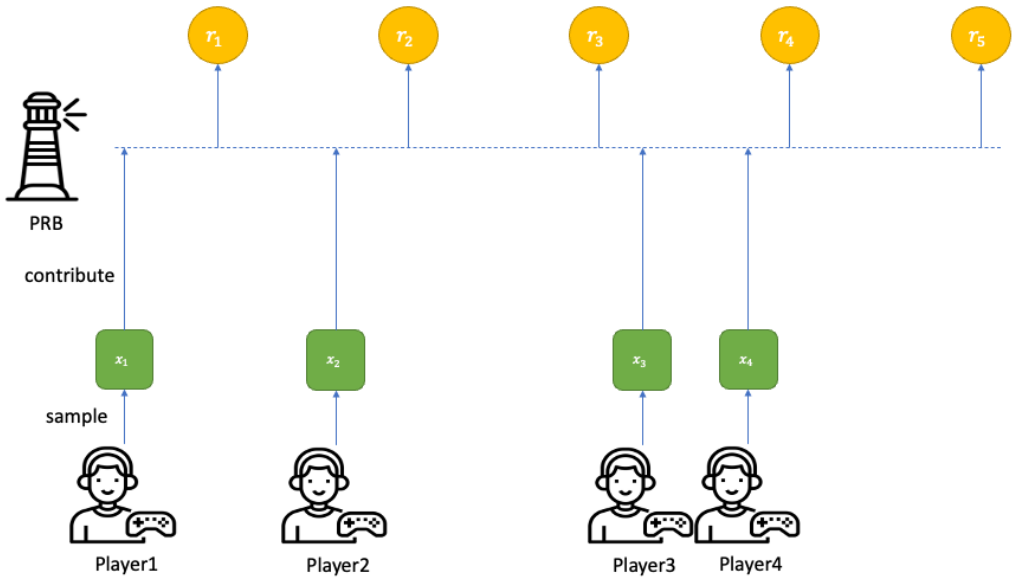

Figure 5.2: The flowchart of randomness contribution phase in probability verification protocol

Last, the server publishes the evaluation and proofs $(y_1, \pi_1), ..., (y_m, \pi_m)$ on the bulletin board.

Verification In this phase, the client first retrieves the randomness r from the randomness beacon and invokes `PRB.Verify` to verify the validity of the randomness outcome. If the verification fails, the client aborts. Secondly, the client maps r to test data using the mapping function M, and obtains $(r_1, o_1), ..., (r_m, o_m)$. Then, the client retrieves the evaluations from the bulletin board and verifies them via

$$\texttt{FC.Verify}(c, (r_i, o_i), y_i, \pi_i) \stackrel{?}{=} 1$$

for $1 \leq i \leq m$

Last, the client can use those verified evaluations to determine the accuracy of the probability statement. We illustrate the last two phases in Fig. 5.3 and summarize the whole protocol in Fig. 5.4.

doi:10.6342/NTU202301027

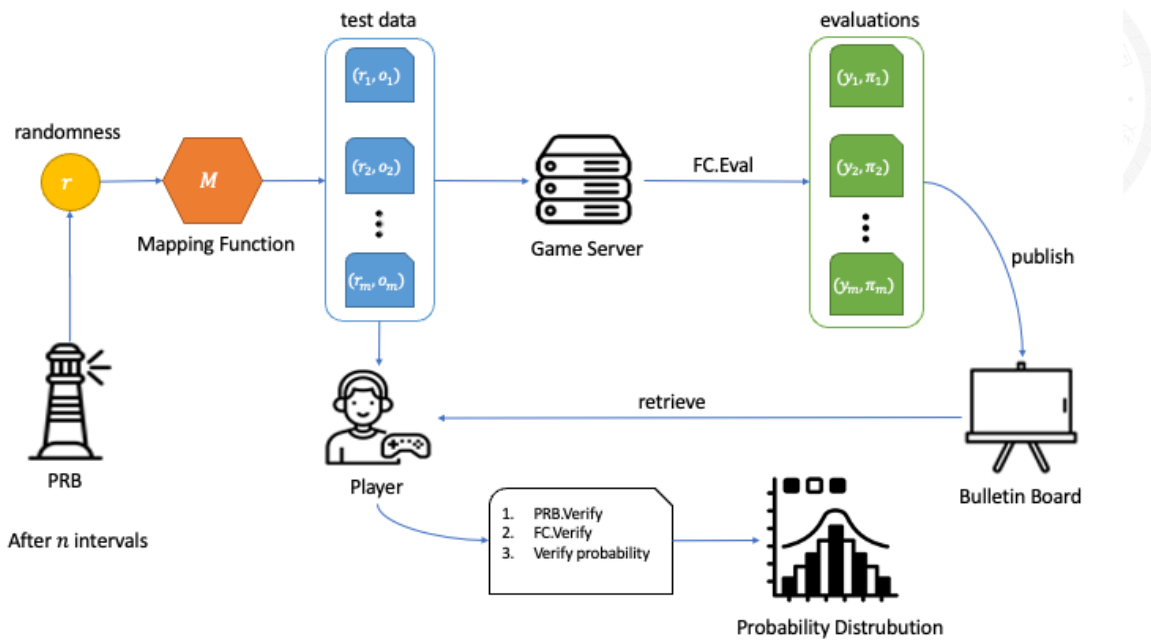

Figure 5.3: The flowchart of evaluation and probability verification phase in probability verification protocol

5.1.2 Probability Verification Through Dry-Run API

Although a public randomness beacon can be used to sample test data, the `others` parameter cannot be customized for each player. For example, suppose that `others` includes the amount of money a player has been paid, such as 100. In this case, there may not be any test data that maps `others` to 100, making it impossible for the player to verify the probability for the exact same situation.

One possible solution to address this issue is to introduce a "dry-run API" that allows players to test loot box openings without spending real money. Dry-run API provides the same functionality as the loot box opening protocol (details will be discussed in section 5.2) but without requiring any payment for the opening requests. We illustrate this dry-run API verification in Fig. 5.5.

The reason this approach works is that functional commitment ensures that the dry-run and real APIs both use the same function. If the functional commitment were not used,

doi:10.6342/NTU202301027

Protocol 1

Setup

- Setup the parameters of cryptographic primitives, including public randomness generation and functional commitment.
- The server designates the number of beacon intervals to be counted from the present time, denoted as n.
- The server designates the mapping function M, which maps the public randomness to the test data.
- The server commits f to a functional commitment, denoted as c.
- The server publishes the public parameters, the interval number n, the mapping function M, and the commitment c on the bulletin board.

Randomness Contribution

- The clients examine the validity of the setup data published by the server, including public parameters, n, M, and c
- The clients sample their local randomness and contribute through `PRB.Contribute`.

Evaluation

- After n beacon intervals, the server obtains the outcome of the n^{th} randomness beacon, denoted as r.
- The server maps r to m test data

$$M(r) \rightarrow \{(r_1, o_1), ..., (r_m, o_m)\}$$

  where $r_i \in R$ and $o_i \in \mathbb{O}$ for $1 \leq i \leq m$.

- The server evaluates the function f on the test data and creates the evaluation proofs, denoted by $(y_1, \pi_1), ..., (y_m, \pi_m)$. After that, publish them on the bulletin board.

Probability Verification

- The client retrieves the randomness r, and invokes `PRB.Verify`. If the verification fails, the client aborts.
- The client maps r to test data $(r_1, o_1), ..., (r_m, o_m)$, and verifies the evaluation via $\texttt{FC.Verify}(c, (r_i, o_i), y_i, \pi_i)$ for $1 \leq i \leq m$. If the verification fails, the client aborts.
- The verified evaluation can now be used by the client to determine the accuracy of the probability statement.

Figure 5.4: Protocol 1: Probability Verification Through Public Randomness Beacon

doi:10.6342/NTU202301027

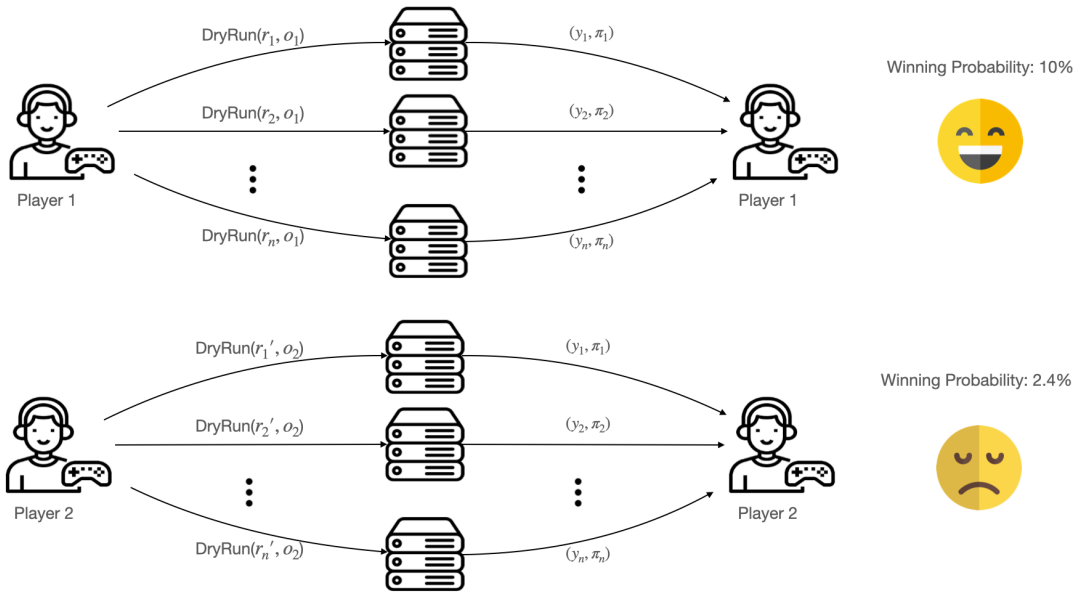

Figure 5.5: The flowchart of probability verification through dry-run API

the server could potentially use different functions for the dry-run API, leading the client to believe in a higher probability statement that may not be accurate.

In practice, allowing too many dry-runs can risk exposing information about the function f. While players previously had to spend a significant amount of money to gather information about the function, it now comes at no cost. The number of attempts allowed for players is determined by the game companies, who must find a trade-off between transparency and the need to protect their trade secrets.

5.1.3 Probability Verification Through Public Bulletin Board

Both of the verification methods mentioned above rely on the game companies, which may have motivations to reduce the number of test data. To address this issue, we propose a method that utilizes the power of the crowd. This method allows the players to upload their loot box opening records onto a public bulletin board, from which other players can download the records, verify them and calculate the probability distribution.

doi:10.6342/NTU202301027

This approach is similar to that of the 3rd-party verification platform we survey in [19]. However, we can programmatically verify the authenticity of the uploaded records using `FC.Verify`. This approach not only requires much less storage space but also guarantees 100% authenticity of the data.

Unfortunately, the issue of selective data uploading cannot be completely resolved, as servers can impersonate clients and choose to upload only advantageous records. Similarly, clients can also selectively upload records with lower winning probabilities.

However, this approach can still be a viable option for players who wish to collect more information from the public and are willing to accept some degree of inaccurate or false data.

5.1.4 Probability Verification Through Third-Party Auditing

While the three aforementioned verification methods are publicly-verifiable, they are still subject to server restrictions. These restrictions may include reducing the number of test data or dry-runs, or selectively uploading winning records to the bulletin board. There is inherently a trade-off between transparency and trade secrets.

By utilizing a trusted third-party auditor, such as the Fair Trade Commission or a company specializing in code review, game companies could confidentially transfer their source code to the auditor for a thorough review to verify the accuracy of the probability statement. If the verification process is successful, the auditor can then sign the functional commitment c, which enables clients to ensure that the server is using the same function that was reviewed and signed by the auditor. In practice, game companies may request multiple auditors to sign the commitment, allowing clients to choose the auditor they trust

doi:10.6342/NTU202301027

the most.

In conclusion, utilizing a trusted third-party auditor can result in a higher level of accuracy when verifying probability statements, while still allowing game companies to retain more trade secrets.

5.2 Loot Box Opening Protocol

Once the client has completed the probability verification protocol and determined the actual winning probability p of f, the objective of the loot box opening protocol is to provide individual verifiability. This means that during the result verification phase, each client can confirm that the server has not biased the winning probability in any way.

The protocol comprises three phases: setup, evaluation, and result verification. Each phase is described in detail in the following paragraphs. Alternatively, a summary of the protocol can be found in Figure 5.6, and a flow chart can be found in Figure 5.7.

Setup In the setup phase, the server samples a randomness α_0 and generates a hash chain the length N.

$$\{\alpha_i | \alpha_i \leftarrow H^i(\alpha_0), 1 \leq i \leq N\}$$

where H is a cryptographic hash function.

The server then sends the last element α_N of the hash chain to the client.

Evaluation In the evaluation phase, the client first samples a local randomness β and prepares input parameters `others`, which may include metadata about the player such as

doi:10.6342/NTU202301027

game level, number of loot box openings, etc. The client then sends $(\beta, \texttt{others})$ to the server.

Assuming that the last opened hash chain position is α_i, the server uses $\alpha_{i-1} \parallel \beta$ as the random source. Namely, the server evaluates

$$f(\alpha_{i-1} \parallel \beta, \texttt{others}) \to y$$

The server then generates the evaluation proof:

$$\texttt{FC.Eval}(\mathbf{pp}, (\alpha_{i-1} \parallel \beta, \texttt{others}), y) \to \pi$$

Last, the server sends back (α_{i-1}, y, π) to the client.

Result Verification After receiving (α_{i-1}, y, π) from the server, the client first verifies the validity of α_{i-1}:

$$H(\alpha_{i-1}) \stackrel{?}{=} \alpha_i$$

Then, the client verifies the validity of the evaluation proof:

$$\texttt{FC.Verify}(\mathbf{pp}, (\alpha_{i-1} \parallel \beta, \texttt{others}), y, \pi) \stackrel{?}{=} 1$$

If both of the above verification pass, the client can be confident that the server has not biased the winning probability of this loot box opening.

In this protocol, we use a hash chain to ensure that the server cannot manipulate the randomness after the setup phase. On the other hand, using a hash chain enables the server to provide randomness unpredictable from the client, which prevents clients with

doi:10.6342/NTU202301027

Loot Box Opening Protocol

Setup

- The server samples a randomness α_0, and generate a hash chain

$$\{\alpha_i | \alpha_i \leftarrow H^i(\alpha_0), 1 \leq i \leq N\}$$

  The server sends the last element α_N to the client.

Evaluation

- The client generates the local randomness β, prepares the input data `others`, and sends both to the server.
- Assuming the last opened hash chain position is α_i, the server uses $\alpha_{i-1} \parallel \beta$ as the random source. Namely, the server evaluates

$$f(\alpha_{i-1} \parallel \beta, \texttt{others}) \rightarrow y$$

- The server also computes `FC.Eval`, and obtain the evaluation proof π.
- The server sends back (α_{i-1}, y, π) to the client.

Result Verification

- The client verifies the hash chain via $H(\alpha_{i-1}) \stackrel{?}{=} \alpha_i$
- The client verifies the evaluation proof via `FC.Verify`.

Figure 5.6: Loot Box Opening Protocol

knowledge of the function from biasing the probability.

One practical concern is that the server may need to repeat the setup phase if the hash chain is exhausted. While increasing N can reduce the likelihood of this happening, it also leads to higher space usage. To address this, the server can choose to store only the initial element α_1 and compute $\alpha_i \leftarrow H^i(\alpha_0)$ on demand when a client requests a loot box opening. Another approach is to save a subset of breakpoints, such as $\{\alpha_1, \alpha_{101}, \alpha_{201}, \dots\}$, to balance computational and storage requirements.

doi:10.6342/NTU202301027

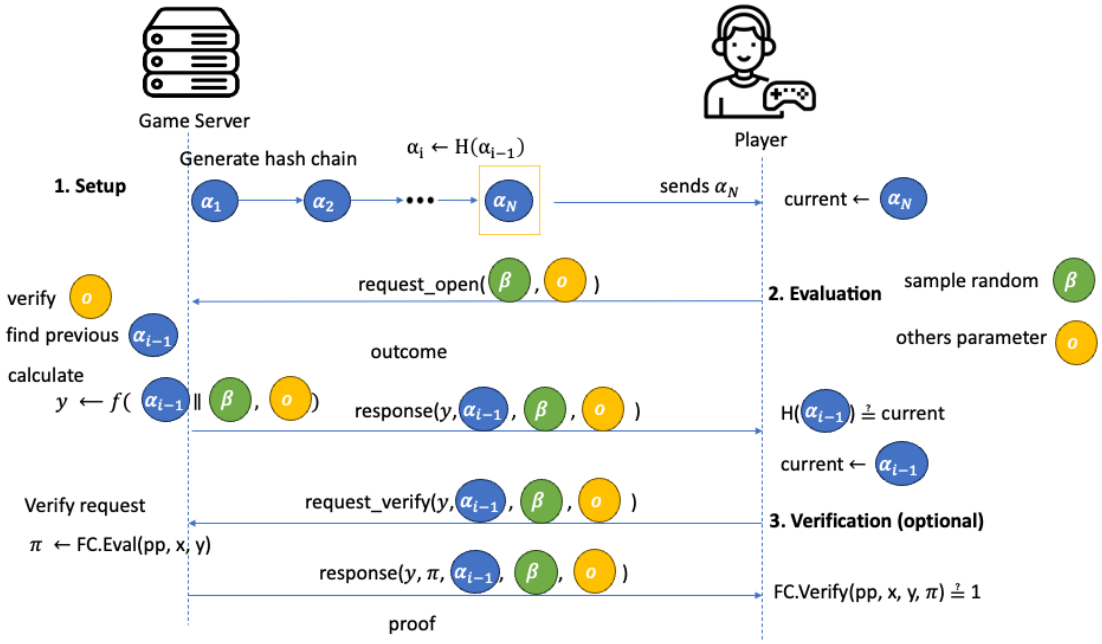

Figure 5.7: The flowchart of loot box opening protocol

doi:10.6342/NTU202301027

Chapter 6 Discussion

6.1 Hypothesis test

To test whether the probability statement is true given some evaluation of test data, we can utilize the concept of a hypothesis test. First, we specify a null hypothesis H_0 and an alternative hypothesis H_1. In our scenario, the null hypothesis is defined as the condition where the claimed probability holds true, while the alternative hypothesis is defined as the condition where the claimed probability does not hold true.

$$H_0 : p \geq p_0$$

$$H_1 : p < p_0$$

Given a significance level α and the results of a hypothesis test on the test data, our goal is to calculate the probability of observing the test results under the assumption that the null hypothesis is true. If this probability is less than the chosen significance level α, we reject the null hypothesis. This implies that the sample data provides strong support for the alternative hypothesis, which suggests that the actual winning probability is very likely to be lower than the claimed probability.

doi:10.6342/NTU202301027

From the central limit theorem, the sample mean of i.i.d random variables will converge to a normal distribution. Assume that the loot box results are i.i.d binomial distribution. The sample mean we have the following approximation:

$$X = \frac{X_1 + ... + X_n}{n} \sim N(p_0, \sqrt{\frac{p_0(1-p_0)}{n}})$$

where X_i is the random variable of each sample result for each $1 \leq i \leq n$. To calculate the critical value when we reject the null hypothesis, we can use the z-score table to find z_α such that

$$Pr(\hat{p} \leq p_0 - z_\alpha \sqrt{\frac{p_0(1-p_0)}{n}}) < \alpha$$

where $\hat{p}$ is the winning probability of sample data. We call $p_L = p_0 - z_\alpha \sqrt{\frac{p_0(1-p_0)}{n}}$ the critical value for the this hypothesis test.

For example, assume the claimed probability $p_0 = 0.3$, sample size $n = 200$, significance level $\alpha = 5\%$, then $z_\alpha = 1.645$ by looking up to the z-score table. We calculate the critical value

$$p_L = p_0 - z_\alpha \sqrt{\frac{p_0(1-p_0)}{n}} = 0.3 - 1.645\sqrt{\frac{0.3 * 0.7}{200}} = 0.247$$

In other words, if the sample winning probability $\hat{p} \leq p_L = 0.247$, we reject the null hypothesis, which means we have strong support that the claimed probability is wrong.

6.2 Determine the number of test inputs

To determine the proper number of test data (sample size), we can utilize statistical power analysis. Given the null hypothesis and the alternative hypothesis defined in the

doi:10.6342/NTU202301027

previous section, we now estimate the false-negative probability given an instance of the alternative hypothesis. Assuming the real winning probability is $p_1 < p_0$, we calculate the probability of rejecting the null hypothesis, called statistical power $(1 - \beta)$. Usually, we expect the statistical power to be greater than 80%, i.e. $\beta < 20\%$. To achieve this, we use the z-score table again to find z_β corresponding to $1 - \beta$. Precisely, we find n satisfying the following inequality:

$$p_1 + z_\beta \sqrt{\frac{p_1(1-p_1)}{n}} \leq p_L = p_0 - z_\alpha \sqrt{\frac{p_0(1-p_0)}{n}}$$

where L.H.S is of the accumulative probability $1 - \beta$ of the sample mean random variable with Bernoulli distribution of probability p_1.

For example, assume the claimed probability $p_0 = 0.3$, actual probability $p_1 = 0.2$, $\alpha = 0.05$, $\beta = 0.2$. We first look up the z-score table and find $z_\alpha = 1.645$, $z_\beta = 0.845$. The above inequality becomes:

$$0.2 + 0.845 \sqrt{\frac{0.2 * 0.8}{n}} \leq 0.3 - 1.645 \sqrt{\frac{0.3 * 0.7}{n}}$$

$$n \geq 115.2$$

In other words, if the sample size is larger than 115.2, we have more than 80% probability to reject the null hypothesis given the actual probability $p_1 = 0.2$ and claimed probability 0.3. In real-world circumstances, a consensus can be reached to select a specific value for p_1, such as $p_1 = 0.9p_0$. This consensus can be established through the efforts of self-regulation associations, government intervention, or the collaboration of the player community.

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

Chapter 7 Security Analysis

7.1 Correctness & Soundness

Assuming that the server provides an adequate amount of test inputs according to the calculation discussed in Section 6.2, the correctness of the protocol is guaranteed with a high probability, as indicated by the parameter β.

On the other hand, by the properties of PRB, it is guaranteed that the test inputs is sampled from unpredictable and bias-resistant random sources. By the **Evaluation Binding** property of FC, it is guaranteed that the server is infeasible to find different evaluation outcomes for a single input. As a result, the soundness is satisfied after the client successfully rejects the null hypothesis we discussed in Section 6.1.

7.2 Public Verifiability

Every player in the public can contribute their randomness to PRB and verify the unpredictability and bias-resistance of the randomness outcome after n beacon intervals. Additionally, every player from the public can also verify the correctness of FC evaluations. Thus, the probability verification protocol satisfies public verifiability.

doi:10.6342/NTU202301027

7.3 Individual Verifiability

Using a hash chain as a random source can prevent the server from manipulating the randomness to its own advantage since the hash function is collision-resistant. On the other hand, since we assume that the server cannot predict or bias the randomness sampled by the client and the client generates its randomness β every time it does the evaluation, we can ensure that the randomness input $\alpha_{i-1} \parallel \beta$ is unpredictable and unbiased by the server. As a result, after the verification of the loot box opening protocol, it is guaranteed that the server does not bias the winning probability.

7.4 Input Transparency

Input transparency means that there is no hidden input to the loot box opening function. This property is followed by the evaluation binding property of functional commitment, where it is guaranteed that given input x, it is infeasible to function distinct evaluation values $y_1 \neq y_2$ that satisfy the verification of $f(x) = y_1$ and $f(x) = y_2$. In order to verify the evaluation proof, the client must also possess knowledge of all input parameters of the loot box opening function. This ensures transparency for all input parameters involved in the process.

7.5 Algorithmic Hiding

Algorithmic hiding is achieved by the hiding and evaluation zero-knowledge properties of functional commitment. In Section 3.4, we gave the full definition of these two

doi:10.6342/NTU202301027

properties. In short, hiding guarantees the indistinguishability of commitment c, and evaluation zero-knowledge guarantees that the evaluation reveals nothing but an evaluation point. In our probability verification protocol, we only reveal m evaluation points, where m is the number of test inputs, specified by the server. It is up to the server to trade off between evaluation leakage and reducing the probability of type-II error. Overall, our protocol guarantees minimal information leakage, i.e. it achieves algorithmic hiding.

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

Chapter 8 Evaluation

In this section, we sketch our implementation of the probability verification protocol and the loot box opening protocol. We begin by examining the computational complexity of our implementation, followed by a comparison of the execution times across various setups. Our experiments were conducted on a Macbook Pro 2021 with the M1 Pro chip.

8.1 Implementation

We implement KZG10 [22] as a special case of a functional commitment scheme. KZG10 is, in fact, a polynomial commitment scheme and serves as a submodule in the construction of [7]. While a polynomial commitment scheme meets all the definitions outlined in section 3.4, it only supports f to be a polynomial. Supporting general circuits in the functional commitment is considered a future work in our current development since the construction in [7] does not provide their implementation.

The source code of our implementation can be accessed on Github [34]. Specifically, we implement our protocols using Python and import the module py_ecc implemented by Ethereum [16] to support elliptic curve operations.

doi:10.6342/NTU202301027

8.2 Complexity Analysis

The complexity of generating the commitment of KZG10 is $O(t)$, where t is the degree of the underlying polynomial. The complexity of generating evaluation proof is also $O(t)$. Finally, the verification complexity is $O(1)$, which consists of two elliptic curve pairing operations.

8.3 Execution Time Analysis

We first analyze the execution times of setup, evaluation, and verification phases in our probability verification protocol over different polynomial degrees. We set sample size = 30, i.e. there are 30 test instances in this experiment. The result is shown in Figure 8.1. As we analyzed in the previous section, the setup and evaluation grows linearly with the polynomial degree and the verification time is constant.

The polynomial degree is about 3 times the number of gates for general circuit construction [17]. Our experiment shows that the evaluation time is practical given a polynomial of degree 100-200 (gate number 30-60).

On the other hand, we also measure the execution time for different sample sizes. With fixed polynomial degree 150, the result is shown in Figure 8.2. As expected, the execution time for both evaluation and verification grows linearly with the sample size.

In conclusion, the execution time of the implemented algorithm is practical for polynomials with degrees ranging from 100 to 200 (30-60 gates for general circuits). For larger degrees, since the execution time grows linearly it can still be considered acceptable de-

doi:10.6342/NTU202301027

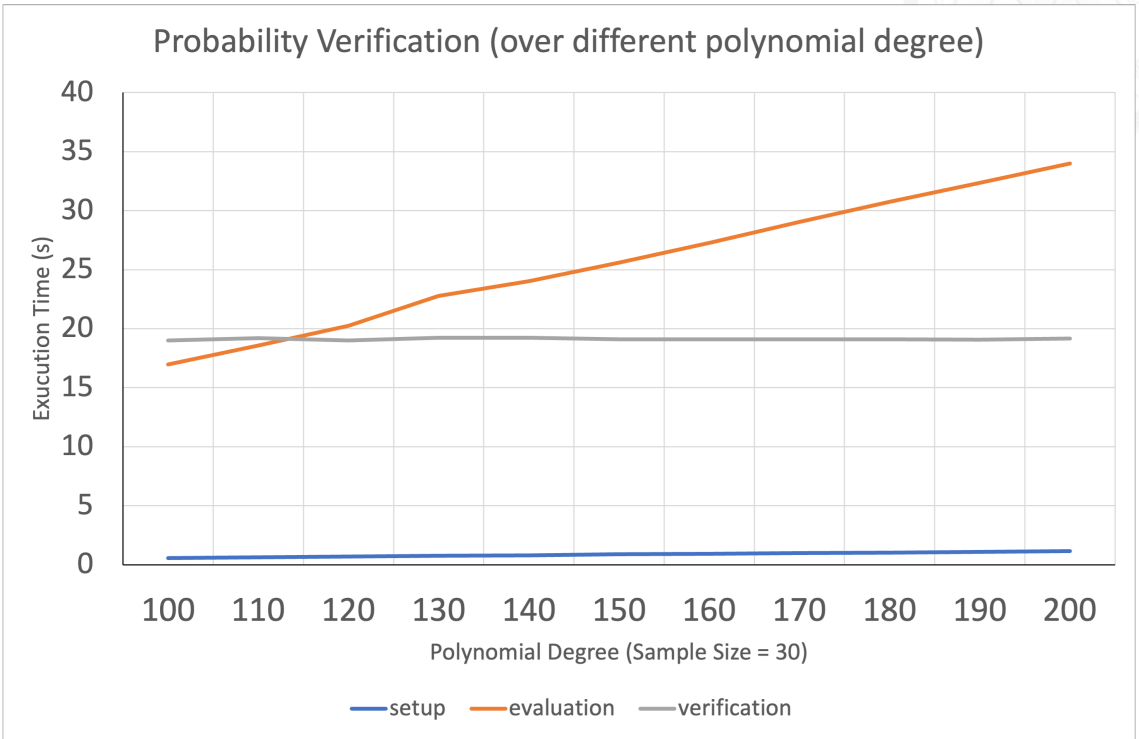

Figure 8.1: The execution time of probability verification protocol over different polynomial degrees

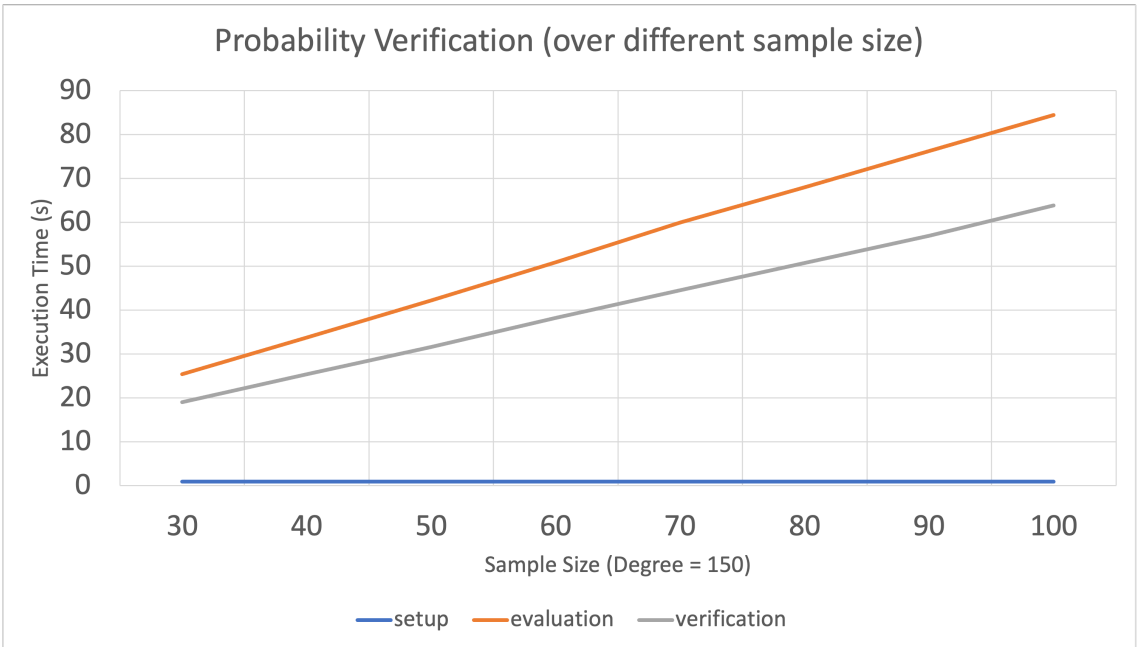

Figure 8.2: The execution time of probability verification protocol over different sample sizes

doi:10.6342/NTU202301027

pending on the specific requirements and constraints of the application.

doi:10.6342/NTU202301027

Chapter 9 Conclusion

In this work, we proposed the probability verification protocol and the loot box opening protocol. The probability verification protocol leverages the public randomness beacon as a verifiable random source while using the functional commitment to verify the correctness of the evaluation result without disclosing the underlying source code. The loot box opening protocol allows the server and the client to agree on a random input, ensuring that neither party can manipulate the outcome. We also introduce hypothesis test and statistical power analysis. Hypothesis testing allows us to determine the truthfulness of the probability statement based on the significance level and the evaluation results of sample inputs. On the other hand, statistical power analysis helps us determine the optimal sample size needed to achieve the desired level of statistical power. Lastly, evaluate our protocols with different settings, demonstrating their efficiency and practicality.

doi:10.6342/NTU202301027

doi:10.6342/NTU202301027

References

[1] Anny. Understand in one article the legal battle between dinter, gamania, and the enactment of gacha regulation. https://www.inside.com.tw/article/30725-DinTer-Gamania-Lineage-M-Lawfare, 2023.

[2] Apple. App store review guidelines. https://developer.apple.com/app-store/review/guidelines/, 2023.

[3] N. G. Authority. Study into loot boxes: A treasure or a burden? Amsterdam, Netherlands, 10, 2018.

[4] A. Bhat, N. Shrestha, Z. Luo, A. Kate, and K. Nayak. Randpiper–reconfiguration-friendly random beacons with quadratic communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 3502–3524, 2021.

[5] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I, pages 757–788. Springer, 2018.

[6] D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. In Advances in Cryptology–CRYPTO

doi:10.6342/NTU202301027

2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39, pages 561–586. Springer, 2019.

[7] D. Boneh, W. Nguyen, and A. Ozdemir. Efficient functional commitments: How to commit to a private function. Cryptology ePrint Archive, 2021.

[8] G. A. Brooks and L. Clark. Associations between loot box use, problematic gaming and gambling, and gambling-related cognitions. Addictive behaviors, 96:26–34, 2019.

[9] A. Carvalho. Bringing transparency and trustworthiness to loot boxes with blockchain and smart contracts. Decision Support Systems, 144:113508, 2021.

[10] I. Cascudo and B. David. Scrape: Scalable randomness attested by public entities. In Applied Cryptography and Network Security: 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings 15, pages 537–556. Springer, 2017.

[11] I. Cascudo, B. David, O. Shlomovits, and D. Varlakov. Mt. random: Multi-tiered randomness beacons. Cryptology ePrint Archive, 2021.

[12] A. Cherniaeva, I. Shirobokov, and O. Shlomovits. Homomorphic encryption random beacon. Cryptology ePrint Archive, 2019.

[13] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. Marlin: Preprocessing zksnarks with universal and updatable srs. In Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39, pages 738–768. Springer, 2020.

doi:10.6342/NTU202301027

[14] T. E. Y. Department of Consumer Protection. Disclosing loot box odds to protect gamers' interests. https://cpc.ey.gov.tw/en/4212D8C5A29ACA5F/61e3c731-23e9-41af-abd0-d12f2912e31c, 2022.

[15] A. Drummond and J. D. Sauer. Video game loot boxes are psychologically akin to gambling. Nature human behaviour, 2(8):530–532, 2018.

[16] Ethereum. Elliptic curve crypto in python. https://github.com/ethereum/py_ecc/, 2023.

[17] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, 2019.

[18] D. Galindo, J. Liu, M. Ordean, and J.-M. Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In 2021 IEEE European Symposium on Security and Privacy (EuroS&P), pages 88–102. IEEE, 2021.

[19] N. T. U. o. S. Gamelab and Technology. Third-party probability verification platform. https://gacha.gamelab.com.tw/, 2023.

[20] Google. Policy center. https://support.google.com/googleplay/android-developer/topic/9858052?hl=en, 2023.

[21] A. Hiramatsu. A research of social game users' attitude to" gacha" probability announcement. In 2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI), pages 115–120. IEEE, 2019.

doi:10.6342/NTU202301027

[22] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16, pages 177–194. Springer, 2010.

[23] S. Kristiansen and M. C. Severin. Loot box engagement and problem gambling among adolescent gamers: Findings from a national survey. Addictive Behaviors, 103:106254, 2020.

[24] H. Lee, Y. Hsu, J.-J. Wang, H. C. Yang, Y.-H. Chen, Y.-C. Hu, and H.-C. Hsiao. Headstart: Efficiently verifiable and low-latency participatory randomness generation at scale. Network and Distributed System Security Symposium (NDSS), 2022.

[25] W. Li, D. Mills, and L. Nower. The relationship of loot box purchases to problem video gaming and problem gambling. Addictive behaviors, 97:27–34, 2019.

[26] LiLi0719. Internal news - the real situation about gacha probability.“. https://forum.gamer.com.tw/C.php?bsn=30518&snA=47974, 2021.

[27] R. C. Merkle. A digital signature based on a conventional encryption function. In Advances in Cryptology—CRYPTO’ 87: Proceedings 7, pages 369–378. Springer, 1988.

[28] E. Obedkov. South korea passes new amendment on loot box probability disclosure. https://gameworldobserver.com/2023/02/28/south-korea-loot-boxes-probability-disclosure-law, 2023.

[29] Paul. Promotion of taiwan’s online game gacha regulation. https://join.gov.tw/idea/detail/ee5dd8b8-bdeb-4d5e-8315-bb0601169d68, 2021.

doi:10.6342/NTU202301027

[30] P. Schindler, A. Judmayer, M. Hittmeir, N. Stifter, and E. Weippl. Randrunner: Distributed randomness from trapdoor vdfs with strong uniqueness. 2021.

[31] P. Schindler, A. Judmayer, N. Stifter, and E. Weippl. Hydrand: Efficient continuous distributed randomness. In 2020 IEEE Symposium on Security and Privacy (SP), pages 73–89. IEEE, 2020.

[32] R. Valentine. Digital games spending reached $127 billion in 2020. https://www.gamesindustry.biz/digital-games-spending-reached-usd127-billion-in-2020/, 2020.

[33] M. von Meduna, F. Steinmetz, L. Ante, J. Reynolds, and I. Fiedler. Loot boxes are gambling-like elements in video games with harmful potential: Results from a large-scale population survey. Technology in Society, 63:101395, 2020.

[34] J. J. Wang. Loot box verification protocols implemented in python. https://github.com/WangJ509/LootBoxVerification, 2023.

[35] B. Wesolowski. Efficient verifiable delay functions. In Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38, pages 379–407. Springer, 2019.

[36] L. Y. Xiao. Regulating loot boxes as gambling? towards a combined legal and self-regulatory consumer protection approach. Interactive Entertainment Law Review, 4(1):27–47, 2021.

[37] L. Y. Xiao. Breaking ban: Belgium's ineffective gambling law regulation of video game loot boxes. Collabra: Psychology, 9(1), 2023.

doi:10.6342/NTU202301027

[38] L. Y. Xiao, L. L. Henderson, R. K. Nielsen, and P. W. Newall. Regulating gambling-like video game loot boxes: A public health framework comparing industry self-regulation, existing national legal approaches, and other potential approaches. Current Addiction Reports, 9(3):163–178, 2022.

[39] L. Y. Xiao, L. L. Henderson, Y. Yang, and P. W. Newall. Gaming the system: suboptimal compliance with loot box probability disclosure regulations in china. Behavioural Public Policy, pages 1–27, 2021.

doi:10.6342/NTU202301027