Same Point Composable and Nonmalleable Obfuscated Point Functions

Peter Fenteany* Benjamin Fuller†

December 20, 2019

Abstract

A point obfuscator is an obfuscated program that indicates if a user enters a previously stored password. A digital locker is stronger: outputting a key if a user enters a previously stored password. The real-or-random transform allows one to build a digital locker from a composable point obfuscator (Canetti and Dakdouk, Eurocrypt 2008).

Ideally, both objects would be nonmalleable, detecting adversarial tampering. Appending a non-interactive zero knowledge proof of knowledge adds nonmalleability in the common random string (CRS) model.

Komargodski and Yogev (Eurocrypt, 2018) built a nonmalleable point obfuscator without a CRS. We show a lemma in their proof is false, leaving security of their construction unclear. Bartusek, Ma, and Zhandry (Crypto, 2019) used similar techniques and introduced another nonmalleable point function; their obfuscator is not secure if the same point is obfuscated twice. Thus, there was no composable and nonmalleable point function to instantiate the real-or-random construction.

Our primary contribution is a nonmalleable point obfuscator that can be composed any polynomial number of times with the same point (which must be known ahead of time). Security relies on the assumption used in Bartusek, Ma, and Zhandry. This construction enables a digital locker that is nonmalleable with respect to the input password.

As a secondary contribution, we introduce a key encoding step to detect tampering on the key. This step combines nonmalleable codes and seed-dependent condensers. The seed for the condenser must be public and not tampered, so this can be achieved in the CRS model. The password distribution may depend on the condenser's seed as long as it is efficiently sampleable. This construction is black box in the underlying point obfuscation.

Nonmalleability for the password is ensured for functions that can be represented as low degree polynomials. Key nonmalleability is inherited from the class of functions prevented by the nonmalleable code.

Keywords: Digital lockers; Point obfuscation; Virtual black-box obfuscation; Nonmalleable codes; Seed-dependent condensers; Nonmalleability

1 Introduction

Obfuscation hides the implementation of a program from all users of the program. This work is concerned with *virtual black-box obfuscation*, where an obfuscator creates a program that reveals nothing about the program other than its input and output behavior [BGI+01, BGI+12]. Barak et al. showed that a virtual

*Email: peter.fenteany@uconn.edu. University of Connecticut.

†Email: benjamin.fuller@uconn.edu. University of Connecticut.

black-box obfuscator cannot exist for all polynomial time circuits [BGI+01]. However, this leaves open the possibility of virtual black-box obfuscators for interesting classes of programs [CD08, BC10, CRV10, WZ17, BR17].[1]

We focus on *obfuscated point functions* [Can97] and *digital lockers* [CD08] [BC10]. A *point function obfuscator* is an algorithm $\mathsf{lockPoint}(\mathsf{val})$ which outputs a circuit $\mathsf{unlockPoint}_{\mathsf{val}}(\cdot)$. The circuit $\mathsf{unlockPoint}_{\mathsf{val}}(\cdot)$ stores val and indicates when val is inputted to it. An obfuscated point function needs to hide all partial information about val [Can97].

A digital locker obfuscator inputs a value, val, and key, key. The output is a program $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}(\cdot)$ which outputs key if and only if the input is val. Soundness says $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}$ should reveal nothing about val or key if the adversary cannot guess val. Digital lockers have applications in password [Can97] and biometric authentication [CFP+16, ABC+18].

It is possible to compose point functions to build a digital locker using the *real-or-random* construction [CD08]. It works as so: sample a random point r. For each bit of the key, either r (corresponding to a 0 in key) or val (corresponding to a 1) is obfuscated. An obfuscation of val is prepended as a check value. When running the program, if the check obfuscation opens, the user runs the other programs: failures to open correspond to a key bit 0 and successes correspond to a key bit of 1. Crucially, the point function must retain security when val is obfuscated multiple times.

Nonmalleability A desirable property of an obfuscated program is nonmalleability. A *nonmalleable* obfuscator detects if an adversary attempts to tamper the obfuscation into a related program [CV09], where being related is defined by some family of functions $\mathcal{F}$. For example, it is desirable to prevent $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}$ from being mauled to $\mathsf{unlock}_{f(\mathsf{val}),f'(\mathsf{key})}$ for $f, f' \in \mathcal{F}$.

In the random oracle model, designing nonmalleable digital lockers and point functions is easy: For a random oracle RO one outputs the program $\mathsf{RO}(\mathsf{val}) \oplus (\mathsf{key}||\mathsf{RO}'(\mathsf{key}))$, where RO and RO' are two independent random oracles of different output length. Similarly, using general non-interactive zero-knowledge proofs of knowledge (NIZKPoKs) in the common random string (CRS) model one can achieve nonmalleability. For $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}(\cdot)$, appending a NIZKPoK of key and val would prevent the adversary from creating a valid obfuscation for any point related to the inputs.[2]

Komargodski and Yogev constructed a nonmalleable point obfuscator without resorting to these tools [KY18a]. Their construction follows. Let g be a fixed group generator. To obfuscate the point val, the obfuscator computes a random r and outputs

$$\mathsf{lockPoint}(\mathsf{val}) = \left(r, r^{g^{\sum_{i=1}^{4} \mathsf{val}^i}}\right).$$

We observe that nonmalleability of Komargodski and Yogev's scheme relies on an incorrect lemma in a way that is not apparently repairable. We discuss this in detail below.

Bartusek, Ma, and Zhandry [BMZ19] using similar mathematical structure showed a nonmalleable point function using random a, b, c:

$$\mathsf{lockPoint}(\mathsf{val}) = a, g^{a\cdot\mathsf{val}+(\mathsf{val})^2+(\mathsf{val})^3+(\mathsf{val})^4+(\mathsf{val})^5}, b, g^{b\cdot\mathsf{val}+(\mathsf{val})^6}, c, g^{c\cdot\mathsf{val}+(\mathsf{val})^7}.$$

The structure of the group element is similar to Komargodski and Yogev's construction, but with a random scalar in place of "double exponentiation." The terms involving b and c ensure no incorrect

[1] We do not consider indistinguishability obfuscation in this work [GGH+13, GGH+16, SW14, PST14, GLSW15, AJ15].

[2] The adversary can always substitute an obfuscation on an unrelated point. Thus, it is possible to create obfuscations for functions f where $f(\mathsf{val})$ is easy to guess.

point causes the obfuscation to unlock. In both constructions, g is assumed to be fixed; this means the distribution of val may depend on generator g. Bartusek, Ma, and Zhandry [BMZ19] show security based on new Diffie-Hellman variants and show these variants hold in the generic group model, using tools from the auxiliary input generic group model [CDG18].

The natural nonmalleability definition is that, given $\mathsf{unlockPoint}_{\mathsf{val}}$, an adversary can only output the same obfuscation or obfuscations of independent points. The above constructions use a weaker definition. Given an obfuscation $\mathsf{lockPoint}_{\mathsf{val}}$, the adversary is required to output a function f and an obfuscation $\mathsf{lockPoint}_{f(\mathsf{val})}$. That is, the definition requires the adversary to *know* what tampering function they are applying. Both constructions consider f as a polynomial of bounded degree related to the assumed hardness in the DDH assumptions. The definition considers the tampering functions prevented, not what operations are performed by the adversary.

The goal of this work is to construct nonmalleable digital lockers. The real-or-random construction instantiated with nonmalleable point functions would provides nonmalleability over val. Crucially, this construction requires security to hold when the nonmalleable point functions are *composed* though only with the same val. Both previous constructions have issues that prevent incorporation. The proof of nonmalleability for [KY18a] relies on an untrue lemma and the proof does not seem easily repairable, and the construction of [BMZ19] cannot be composed twice or more. We discuss these issues and then introduce our contribution.

[KY18b, Lemma 4.6] is not true Let g be a fixed generator of a group. The version of Komargodski and Yogev published in Eurocrypt 2018 [KY18a] relied on a *fixed generator* power DDH assumption which says for any distribution x with super logarithmic entropy (here the distribution of x can depend on generator g) that

$$g, g^x, g^{x^2}, ..., g^{x^t} \approx_c g, g^{u_1}, g^{u_2}, ..., g^{u_t},$$

for a truly random set of elements $u_1, ..., u_t$. This assumption is used in the proof by assuming that the adversary sees $\sum_{i=1}^{4} u_i$ and arguing they can't predict any linear combinations other than $c\sum_{i=1}^{4} u_i$ for some constant c. However, Bartusek, Ma, and Zhandry [BMZ19] showed that for a fixed generator this assumption cannot be true: x can be drawn from points where most bits of g^x are fixed. As a result, a revised version [KY18b] proposes a revised assumption called entropic power where

$$g, g^x, g^{x^2}, ..., g^{x^t} \approx_c g, g^{z_1}, g^{z_2}, ..., g^{z_t}.$$

Where z_i are independent and have some super logarithmic min-entropy. This assumption does not appear to suffice. In particular, [KY18b, Lemma 4.6] is incorrect as stated. The lemma states it is hard to predict linear combinations of z_i other than $c\sum_{i=1}^{4} z_i$ for any constant $c \in \mathbb{G}$, even knowing $\sum_{i=1}^{4} z_i$. However, even if each z_i has entropy, the value $\sum_{i=1}^{4} z_i$ may uniquely determine each z_i: let z_i vary in the ith quarter of bits and fix the rest of bits to be 0. The attack of [BMZ19] prevents arguing that z_i has any greater amount of entropy.

This does not seem to be an issue of just the proof technique. The point of the entropic power assumption is to switch to an information-theoretic setting where the adversary cannot predict new powers from a linear combination, but bounding the entropy of each z_i may cause all powers to be predictable. Repairing this scheme seems to require a new Diffie-Hellman assumption or a major change in analysis.

[BMZ19] is not composable One might try to compose the construction of Bartusek et al. [BMZ19]. However, this scheme is not secure even when used twice for the same val. The hardness of finding g^x is

the underlying assumption used to show nonmalleability [BMZ19, Assumption 4]. Since the distribution of x may depend on g, one can construct distributions x where g^x is distinguishable from a random group element g^r. If one can find g^x, the scheme can not be secure. If one tries to obfuscate the same point x twice, all the higher order terms can be removed by dividing two instances. That is, given

$$a_1, g_1 = g^{a_1 x + x^2 + x^3 + x^4 + x^5}$$
$$a_2, g_2 = g^{a_2 x + x^2 + x^3 + x^4 + x^5}$$

one can easily compute $g^{(a_1 - a_2)x} = g_1/g_2$, and so we recover

$$g^x = (g_1/g_2)^{(a_1 - a_2)^{-1}}.$$

1.1 Our Contribution

The primary contribution of this work is the first same-point composable nonmalleable point function The composable, nonmalleable point function can instantiate the real-or-random construction providing a nonmalleable digital locker that prevents tampering over val only. This construction is in the standard model.

As a secondary contribution, we introduce a key encoding step to detect tampering on key. The key encoding step allows us to achieve a digital locker that is nonmalleable over both val and key. However, our key encoding step requires a public value that all distributions can depend on. This can be achieved in the common random string (CRS) model. In our construction the distribution of val can depend on the public value. In the CRS model, one can achieve nonmalleability in a non-black box way using non-interactive zero knowledge proofs of knowledge [CV09].

Composable Same Point Nonmalleable Point Function Obfuscation We introduce a new nonmalleable point function that can be safely composed τ times as long as the *same point* is obfuscated each time. The construction builds on the one time scheme of Bartusek et al. [BMZ19]. We include additional randomized powers to prevent the above attack. The construction needs to know the desired composition parameter τ ahead of time. The value τ would be known in the case when a point function is being used to construct a digital locker. Let $\vec{a}, \vec{b}, \vec{c}$ be uniform vectors of length τ. The construction is as follows:

$$\mathsf{lockPoint}(x; \vec{a}, \vec{b}, \vec{c}) \stackrel{def}{=} \begin{bmatrix} \vec{a}, & g^{\sum_{i=1}^{\tau} \vec{a}_i x^i + \sum_{i=\tau+1}^{\tau+5} x^i}, \\ \vec{b}, & g^{\vec{b}_1 x + \sum_{i=2}^{\tau} \vec{b}_i x^{i+\tau+5} + x^{2\tau+5}}, \\ \vec{c}, & g^{\vec{c}_1 x + \sum_{i=2}^{\tau} \vec{c}_i x^{i+2\tau+4} + x^{3\tau+5}}. \end{bmatrix}$$

The intuition for the formation of the first group element is that we need to randomize more powers to prevent the adversary from removing the higher order powers and being able to linear solve for g^x. Since the adversary can create $\tau - 1$ linearly independent pairs, τ randomized powers are necessary. We add a fifth non-randomized power in the $\vec{a}$ term to deal with the additional flexibility created by τ. The crucial step in the proof is showing that some linear system has no interesting solutions, the extra power is to counteract the degree of freedom introduced by τ (see Theorem 3.2).

The intuition for the $\vec{b}$ and $\vec{c}$ terms is similar. Due to our proof technique, we need to randomize different powers for the $\vec{a}$ term, the $\vec{b}$ term, and the $\vec{c}$ term, resulting in the above construction. All terms have a randomized x^1 coefficient so we can reduce to [BMZ19, Assumption 4].

We can instantiate the real-or-random construction with this construction to yield a nonmalleable digital locker that only provides nonmalleability over the locked val.

Detecting Key Tampering Our definition of nonmalleability for digital lockers uses two supplemental algorithms called key verifier and value verifier, denoted as $\mathsf{V}_{\mathsf{key}}$ and $\mathsf{V}_{\mathsf{val}}$ respectively. The value verifier was defined for nonmalleable point functions (Definition 2.2). If the entire process could be controlled by the adversary, they could output a circuit that says the key' is always good. With these in hand, we define the following game (Definition 2.7):

1. Before input to the digital locker the key is encoded, $c = \mathsf{Enc}(\mathsf{key}, \mathsf{val})$ and placed in the digital locker $\mathsf{lock}(\mathsf{val}, c)$ (that doesn't detect tampering of key). Note, we allow c to depend on val.
2. The adversary performs tampering, outputs functions f_{key} and f_{val}, and a new obfuscation unlock'.
3. The adversary wins if
   (a) unlock' passes the value verifier. That is, $\mathsf{V}_{\mathsf{val}}(\mathsf{unlock}') \neq \perp$,
   (b) If $c' \stackrel{def}{=} \mathsf{unlock}'(f_{\mathsf{val}}(\mathsf{val}))$ passes the key verifier. That is, $\mathsf{V}_{\mathsf{key}}(c', f_{\mathsf{val}}(\mathsf{val})) \neq \perp$, and
   (c) The obfuscation is consistent with the provided functions: $c' = f_{\mathsf{key}}(\mathsf{unlock}(f_{\mathsf{val}}(\mathsf{val}))$.

We require the adversary to output its tampering functions as in [KY18a] and [BMZ19]. This is a limitation of the current definition, a stronger definition would keep the adversary from performing any prohibited tampering function, even if they are not *aware* of what function they are applying. Our proof techniques do not seem to extend to this stronger definition. The key verifier algorithm takes val as input since it is available, however, the *mauled* val is provided.

The adversary tampers c, but the definition makes no claim whether a tampering class is easy or hard. Since permutations and one directional bit flips (replacing an obfuscation with one of a random point) are easily computable, the encoding strategy should prevent them. This is possible for our encoding strategy (see below).

As mentioned above, one could prove knowledge (using a NIZKPoK) of just key to prevent modification of this value. Such a method would inherently depend on the underlying point digital locker. Our goal is to avoid general NIZKPoKs. Our strategy is to use a nonmalleable code [DPW10] to ensure an adversary can only tamper to independent values and a seed-dependent condenser [DRV12] to ensure an independent value is unlikely to pass the key verifier. We discuss the approach and alternative tools in Section 4.

Our construction requires `seed` to be public and not tampered. This can be achieved in the CRS model. Importantly, our construction does not assume independence of distributions from the random object. The CRS is only necessary for preventing tampering of key, the real-or-random construction prevents tampering of val in the standard model.

Organization In Section 2, we present definitions. In Section 3, we introduce the composable nonmalleable point function. In Section 4, we present the real-or-random digital locker construction and add the key encoding and verifier.

2 Preliminaries

For random variables X_i over some alphabet $\mathcal{Z}$ we denote by $X = X_1, ..., X_n$ the tuple $(X_1, \ldots, X_n)$. For a set of indices J, X_J is the restriction of X to the indices in J. The *minentropy* of X is $\mathrm{H}_\infty(X) = -\log(\max_x \Pr[X = x])$, and the *average (conditional) minentropy* [DORS08, Section 2.4] of X given Y is

$$\tilde{\mathrm{H}}_\infty(X|Y) = -\log\left(\mathop{\mathbb{E}}_{y \in Y} \max_x \Pr[X = x|Y = y]\right).$$

For a distinguisher D, the *computational distance* between X and Y is $\delta^D(X,Y) = |\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]|$ (we extend it to a class of distinguishers $\mathcal{D}$ by taking the maximum over all distinguishers $D \in \mathcal{D}$). We denote by $\mathcal{D}_s$ the class of randomized circuits which output a single bit and have size at most s. Logarithms are base 2. In general, capitalized letters are used for random variables and the corresponding lowercase letters for their samples. We say that two circuits, C and C', with inputs in $\{0,1\}^\lambda$ are equivalent if $\forall x \in \{0,1\}^\lambda, C(x) = C'(x)$. We denote this as $C \equiv C'$. For a matrix $\mathbf{A}$ let $\mathbf{A}_{i,j}$ denote the entry in the ith row and the jth column. Let $\mathbf{A}_{(\cdot,j)}$ represent the jth column and $\mathbf{A}_{(i,\cdot)}$ represent the ith row.

Definition 2.1. *An ensemble of distributions $\mathcal{X} = \{X_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{X}_\lambda$ is over $\{0,1\}^\lambda$, is* well-spread *if*

1. *It is efficiently and uniformly samplable. That is, there exists a PPT algorithm given 1^λ as input whose output is identically distributed as X_λ.*
2. *For all large enough $\lambda \in \mathbb{N}$, it has super-logarithmic minentropy. Namely, $\mathrm{H}_\infty(X_\lambda) = \omega(\log \lambda)$.*

2.1 Obfuscation Definitions

All obfuscation definitions include a requirement of *polynomial slowdown*, which says the running time should be at most a polynomial factor larger than the original program. Running time of our constructions can be easily verified. For all definitions, we include a tampering function $\mathcal{F}$. The traditional definition can be achieved by taking $\mathcal{F} = \emptyset$. We adapt nonmalleability definitions from Komargodski and Yogev [KY18a]. See Komargodski and Yogev for definitional considerations [KY18a].

Our constructions require that the challenger can recognize a legitimate obfuscation. We call this object a value verifier or $\mathsf{V}_{\mathsf{val}}$. It was called a verifier (without the word value) in [KY18a].

Definition 2.2 (Value Verifier)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathcal{O}$ be a program that takes inputs $x \in \{0,1\}^\lambda$ and outputs a program $\mathcal{P}$. A PPT algorithm $\mathsf{V}_{\mathsf{val}}$ is called a* value verifier *if for all $x \in \{0,1\}^\lambda$, it holds that $\Pr[\mathsf{V}_{\mathsf{val}}(\mathcal{P}) = 1 | \mathcal{P} \leftarrow \mathcal{O}(x)] = 1$, (prob. over the randomness of $\mathsf{V}_{\mathsf{val}}$ and $\mathcal{O}$).*

Our constructions consist of tuples of group elements and strings. The obvious value verifier suffices as long as group elements are recognizable.

Point Obfuscators A point function is a function I_{val}: $\{0,1\}^n \mapsto \{0,1\}$ that outputs 1 on input val and 0 elsewhere. An obfuscator preserves functionality while hiding the point val if val is not provided as input to the program. In this work we consider a version that allows for the same point to be obfuscated multiple times while retaining security.

Definition 2.3 (τ-Same Point Nonmalleable Point Function)**.** *For security parameter $\lambda \in \mathbb{N}$, let $\mathcal{F} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a family of functions, let $\mathcal{X}$ be a family of distributions over $\{0,1\}^\lambda$. A $(\mathcal{F},\mathcal{X})$-non* malleable point function obfuscation $\mathsf{lockPoint}$ *is a PPT algorithm that inputs a point $\mathsf{val} \in \{0,1\}^\lambda$, and outputs a circuit $\mathsf{unlockPoint}$. Let $\mathsf{V}_{\mathsf{val}}$ be a value verifier for $\mathsf{lockPoint}$ as defined in Definition 2.2. The following properties must hold:*

1. ***Completeness**: For all $\mathsf{val} \in \{0,1\}^\lambda$, it holds that*

$$\Pr[\mathsf{unlockPoint}(\cdot) \equiv I_{\mathsf{val}}(\cdot) | \mathsf{unlockPoint} \leftarrow \mathsf{lockPoint}(\mathsf{val})] \geq 1 - \mathtt{ngl}(\lambda),$$

where the probability is over the randomness of $\mathsf{lockPoint}$.

2. ***Soundness:*** *For every PPT $\mathcal{A}$ and any polynomial function p, there exists a simulator $\mathcal{S}$ and a polynomial $q(\lambda)$ such that, for all large enough $\lambda \in \mathbb{N}$, all $\mathsf{val} \in \{0,1\}^\lambda$ and for any predicate $\mathcal{P} : \{0,1\}^\lambda \mapsto \{0,1\}$,*

$$\begin{aligned}&|\Pr[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^\tau) = \mathcal{P}(\mathsf{val})|\{\mathsf{unlockPoint}_i\}_{i=1}^\tau \leftarrow \mathsf{lockPoint}(\mathsf{val})] \\ &- \Pr[\mathcal{S}^{I_{\mathsf{val}}(\cdot)}(1^\lambda) = \mathcal{P}(\mathsf{val})]| \leq \frac{1}{p(\lambda)},\end{aligned}$$

where S is allowed $q(\lambda)$ oracle queries to I_{val} and the probabilities are over the internal randomness of $\mathcal{A}$ and $\mathsf{lockPoint}$, and of $\mathcal{S}$, respectively. Here $I_{\mathsf{val}}(\cdot)$ is an oracle that returns 1 when provided input val and 0 otherwise.

3. ***Nonmalleability*** *For any $X \in \mathcal{X}$, for any PPT $\mathcal{A}$, there exists $\epsilon = \mathtt{ngl}(\lambda)$, such that:*

$$\Pr_{\mathsf{val}\leftarrow X}\left[\mathsf{V}_{\mathsf{val}}(C) = 1, f \in \mathcal{F}, (I_{f(\mathsf{val})} \equiv C) \middle| \begin{array}{c}\{\mathsf{unlockPoint}_i\}_{i=1}^\tau \leftarrow \mathsf{lockPoint}(\mathsf{val}) \\ (C, f) \leftarrow \mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^\tau)\end{array}\right] \leq \epsilon.$$

In the above $\mathsf{unlockPoint}_i$ are τ outputs of $\mathsf{lockPoint}$ on the same input point val and independent randomness. Note that the simulator is still only provided with a single oracle. In usual composition definitions the simulator has τ oracles. Since we consider the same point being obfuscated multiple times, all of these oracles would have the same functionality and can be reduced to a single oracle.

In addition to the above traditional definition of soundness, we'll use two auxiliary definitions of privacy for nonmalleable point functions. These are known as distributional indistinguishability and oracle indistinguishability (both first defined in [Can97]).

Definition 2.4 ((Same Point) Distributional Indistinguishability)**.** *An algorithm $\mathsf{lockPoint}$ is called a good distributional indistinguishable (DI) obfuscator if for any PPT $\mathcal{A}$ with binary output and any well-spread distribution $\mathcal{X}$ over points in $\{0,1\}^\lambda$ then there exists some negligible function ϵ such that*

$$\begin{aligned}&|\Pr_{\mathsf{val}\leftarrow\mathcal{X}}[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^\tau) = 1|\mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val})] \\ &- \Pr_{u\xleftarrow{\$}\{0,1\}^\lambda}[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^\tau) = 1|\mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(u)]| \leq \epsilon.\end{aligned}$$

In the previous definition [BC10] of composable distributional indistinguishability in the *random* case the adversary is provided with τ different random points. Here because we focus on obfuscating multiple copies of the same random point, we require indistinguishability from τ copies of a random point.

Definition 2.5 ((Same point) Oracle Indistinguishability)**.** *An algorithm $\mathsf{lockPoint}$ is a oracle indistinguishable (OI) obfuscator if for any PPT adversary $\mathcal{A}$ and any polynomial function p, there exists a polynomial size family of sets $\{L_\lambda\}_{\lambda\in\mathcal{N}}$ such that for all sufficiently large λ and for all $\mathsf{val} \notin L_\lambda$,*

$$\begin{aligned}&|\Pr[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^\tau) = 1|\mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val})] \\ &- \Pr_{\mathsf{val}'\xleftarrow{\$}\{0,1\}^n}[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^\tau) = 1|\mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val}')]| \leq \frac{1}{p(\lambda)}.\end{aligned}$$

Bitanski and Canetti [BC10] previously showed that τ-distributional indistinguishability implies τ-composable virtual gray box obfuscation (for point functions). They then showed that for $\tau = O(1)$ this implies virtual black box obfuscation. We will need super constant composition. To overcome this barrier we restrict to composition of the same point and directly show their result works for virtual black box obfuscation in this case. These proofs are similar to the original proofs of Canetti [Can97].

Theorem 2.1. *Any τ same point distributional indistinguishable point obfuscator is a τ-composable same point VBB obfuscator.*

Proof. [BC10, Lemma 3.1] shows distributional indistinguishability implies oracle indistinguishability. We need to modify this proof slightly because our versions of distributional indistinguishability and oracle indistinguishability consider the same point.

Lemma 2.1. *Suppose that a point obfuscator* lockPoint *satisfies τ-distributional indistinguishability (for the same point) then it satisfies τ-oracle indistinguishability (for the same point).*

Proof of Lemma 2.1. Consider some binary PPT $\mathcal{A}$ and a polynomial function p. We demonstrate the existence of the family L_λ. Define by X_λ the set of all values $\mathsf{val} \in \mathcal{X}_\lambda$ such that

$$\begin{aligned} &|\Pr[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\tau}) = 1 | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val})] \\ &- \Pr_{\mathsf{val}' \xleftarrow{\$} \{0,1\}^\lambda}[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\tau}) = 1 | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val}')]| > \frac{1}{p(\lambda)}. \end{aligned} \tag{1}$$

Define X_λ^+ as the set of points where Eq (1) is true without the absolute values. Define X_λ^- as the set of points where the negative of Eq (1) is greater than $1/p(\lambda)$. Assume towards a contradiction that $|X_\lambda| = \omega(\mathtt{poly}(\lambda))$. This means that for infinitely many λ it is either true that $|X_\lambda^+|$ or $|X_\lambda^-|$ is super polynomial size. Without loss of generality assume that the first case holds. We construct a well-spread distribution ensemble Z_λ such that $\mathcal{A}$ distinguishes τ obfuscations of points from Z_λ and τ obfuscations of a uniform point. Define the distribution Z_λ as the uniform distribution over X_λ^+. Note that Z_λ is a well spread distribution. Then it must be the case that

$$\begin{aligned} &|\Pr_{\mathsf{val} \leftarrow Z_\lambda}[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\tau}) = 1 | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val})] \\ &- \Pr_{u \xleftarrow{\$} \{0,1\}^\lambda}[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\tau}) = 1 | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(u)]| \geq \frac{1}{p(\lambda)}. \end{aligned}$$

This contradicts the τ-distributional indistinguishability of lockPoint. □

Lemma 2.2. *Suppose that a point obfuscator,* lockPoint *satisfies same point τ-oracle indistinguishability then it satisfies VBB when composed τ times.*

Proof of Lemma 2.2. Let $\mathcal{A}$ be some PPT adversary. Furthermore, suppose that lockPoint satisfies τ-oracle indistinguishability. Define L_λ as the polynomial size set in Definition 2.5. Define the simulator $S_\mathcal{A}$:

1. Query the oracle $\mathcal{I}$ on each point $x \in L_\lambda$. If the oracle returns 1 go to next step and set $\mathsf{val} = x$.
2. If $\mathsf{val} = \perp$ set $\mathsf{val} \xleftarrow{\$} \{0,1\}^\lambda$.
3. Run and output $\mathcal{A}(\{\mathsf{lockPoint}(\mathsf{val})\}_{i=1}^{\lambda})$.

Note that L_λ may differ for each adversary and length λ. Fix some predicate $\mathcal{P}$. We now analyze the quality of $S_\mathcal{A}$ suppose that $\mathsf{val} \in L_\lambda$ then $S_\mathcal{A}$ outputs exactly the distribution $\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\lambda})$ so the

simulation is perfect. In the case when $\mathsf{val} \notin L_\lambda$ by the definition of oracle indistinguishability

$$\begin{aligned}
&\Pr[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\tau}) = \mathcal{P}(\mathsf{val}) | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val})] \\
&- \Pr[\mathcal{S}_{\mathcal{A}}^{I_{\mathsf{val}}(\cdot)}(1^\lambda) = \mathcal{P}(\mathsf{val})] \leq \\
&|\Pr[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{\tau}) = 1 | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val}), \mathsf{val} \notin L_\lambda] \\
- \Pr_{\mathsf{val}' \xleftarrow{\$} \{0,1\}^\lambda} &[\mathcal{A}(\{\mathsf{unlockPoint}_i\}_{i=1}^{t}) = 1 | \mathsf{unlockPoint}_i \leftarrow \mathsf{lockPoint}(\mathsf{val}')]| \leq \frac{1}{p(\lambda)}.
\end{aligned}$$

Thus, in all cases for an arbitrary polynomial $p(\lambda)$ the VBB condition is satisfied. □

The application of Lemma 2.1 and 2.2 proves Theorem 2.1. □

We now present our definition of a nonmalleable digital locker. Our notation for digital lockers adds a key verifier which checks if the key should be accepted. This is analogous to the value verifier in the previous subsection:

Definition 2.6 (Key Verifier)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter and let $n = n(\lambda)$ be a parameter. Let $\mathcal{O}$ be a program that takes inputs $x \in \{0,1\}^\lambda, y \in \{0,1\}^k$ and outputs a program $\mathcal{P}$. A PPT algorithm $\mathsf{V}_{\mathsf{key}}$ (with inputs in $\{0,1\}^{\lambda+n}$ and outputs in $\{0,1\}^k \cup \perp$) for program class $\mathcal{O}$ is called a* key verifier *if it holds that*

$$\Pr[\mathsf{V}_{\mathsf{key}}(x,z) \neq \perp | \mathcal{P} \leftarrow \mathcal{O}(x,y), z \leftarrow \mathcal{P}(x)] = 1,$$

Where the probability is over the randomness of $\mathsf{V}_{\mathsf{key}}$ and $\mathcal{O}$).

Here we note the three different values x, y, z. The value x is the input value, y is the input key, and z as an encoded version of the key. The output of the locker is z which is then checked. There most be value and key independent algorithm that checks z otherwise no manipulation detection is possible. A definition for traditional digital lockers is found in Canetti and Dakdouk [CD08]. Our definition considers tampering on both key and val.

Definition 2.7 (Nonmalleable Digital Locker)**.** *For security parameter $\lambda \in \mathbb{N}$, Let $\mathcal{F} : \{0,1\}^\lambda \to \{0,1\}^\lambda, \mathcal{G} : \{0,1\}^n \to \{0,1\}^n$ be families of functions and $\mathcal{X}$ be a family of distributions over $\{0,1\}^\lambda$. A $(\mathcal{F}, \mathcal{G}, \mathcal{X})$-nonmalleable* digital locker lock *is a PPT algorithm that inputs a point $\mathsf{val} \in \{0,1\}^\lambda$ and string $\mathsf{key} \in \{0,1\}^n$. Let $\mathsf{V}_{\mathsf{val}}$ be a value verifier for lock and let $\mathsf{V}_{\mathsf{key}}$ be a key verifier for lock. The following conditions must be met:*

1. ***Completeness**: For a circuit unlock define the circuit $\mathsf{unlock}'(x) = \mathsf{V}_{\mathsf{key}}(x, \mathsf{unlock}(x))$. For all $\mathsf{val} \in \{0,1\}^\lambda, \mathsf{key} \in \{0,1\}^n$ it holds that*

   $$\Pr[\mathsf{unlock}'(\cdot) \equiv I_{\mathsf{val},\mathsf{key}}(\cdot) | \mathsf{unlock} \leftarrow \mathsf{lock}(\mathsf{val}, \mathsf{key})] \geq 1 - \mathtt{ngl}(\lambda),$$

   where the probability is over the randomness of lock.

2. ***Soundness**: For every PPT $\mathcal{A}$ and any polynomial function p, there exists a simulator $\mathcal{S}$ and a polynomial $q(\lambda)$ such that, for all large enough $\lambda \in \mathbb{N}$, all $\mathsf{val} \in \{0,1\}^\lambda$, all $\mathsf{key} \in \{0,1\}^k$, and for any $\mathcal{P} : \{0,1\}^{\lambda+k} \mapsto \{0,1\}$,*

   $$\left|\Pr[\mathcal{A}(\mathsf{lock}(\mathsf{val}, \mathsf{key})) = \mathcal{P}(\mathsf{val}, \mathsf{key})] - \Pr[\mathcal{S}^{I_{\mathsf{val},\mathsf{key}}}(1^\lambda) = \mathcal{P}(\mathsf{val}, \mathsf{key})]\right| \leq \frac{1}{p(\lambda)},$$

   where S is allowed $q(\lambda)$ oracle queries to $I_{\mathsf{val},\mathsf{key}}$ and the probabilities are over the internal randomness of $\mathcal{A}$ and lock, and of $\mathcal{S}$, respectively. Here $I_{\mathsf{val},\mathsf{key}}$ is an oracle that returns key when provided input val, otherwise $I_{\mathsf{val},\mathsf{key}}$ returns $\perp$.

3. ***Nonmalleability*** *For any distribution $X \in \mathcal{X}$, for any PPT $\mathcal{A}$, for any* $\mathsf{key} \in \{0,1\}^n$*, there exists* $\epsilon = \mathtt{ngl}(\lambda)$ *such that:*

$$\Pr_{\mathsf{val} \leftarrow X}\left[\begin{array}{r|l} \mathsf{V}_{\mathsf{val}}(C) = 1, f \in \mathcal{F}, g \in \mathcal{G}, & \\ y = C(f(\mathsf{val})), & \\ y = g(\mathsf{unlock}_{\mathsf{val},\mathsf{key}}(\mathsf{val})), & \mathsf{unlock}_{\mathsf{val},\mathsf{key}} \leftarrow \mathsf{lock}(\mathsf{val},\mathsf{key}) \\ \mathsf{V}_{\mathsf{key}}(f(\mathsf{val}), y) \neq \perp, & (C,f,g) \leftarrow \mathcal{A}(\mathsf{unlock}_{\mathsf{val},\mathsf{key}}) \\ \exists \alpha \ s.t. \ I_{f(\mathsf{val}),\alpha} \equiv C & \end{array}\right] \leq \epsilon.$$

where at most one of f and g may be the identity function.

If nonmalleability is not a requirement a traditional digital locker can be obtained by outputting $\mathsf{unlock}'(x) = \mathsf{V}_{\mathsf{key}}(x, \mathsf{unlock}(x))$ instead of $\mathsf{unlock}(x)$.

3 A composable nonmalleable point function

In this section, we introduce a new construction of a nonmalleable point function that can be composed as long as the same point is used each time. Our construction draws on ideas from [BMZ19] and is secure under the same assumptions. Their construction is as follows for randomly sampled a, b, c:

$$\mathsf{lockPoint}(\mathsf{val}) = a, g^{a\cdot\mathsf{val}+(\mathsf{val})^2+(\mathsf{val})^3+(\mathsf{val})^4+(\mathsf{val})^5}, b, g^{b\cdot\mathsf{val}+(\mathsf{val})^6}, c, g^{c\cdot\mathsf{val}+(\mathsf{val})^7}.$$

The first group element is the key to nonmalleability, the second two group elements are there to provide correctness. Security of their construction and ours relies on two assumptions (they showed security of these assumptions in the generic group model even if the distribution of val depends on the chosen generator of the group).

Assumption 3.1. *[BMZ19, Assumption 3] Let $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a group ensemble with efficient representation and operations where each $\mathbb{G}_\lambda$ is a group of prime order $p \in (2^\lambda, 2^{\lambda+1})$. We assume that for every $\lambda \in \mathbb{N}$ there is a canonical group (and efficiently computable) and canonical and efficient mapping between the elements of $\{0,1\}^\lambda$ to $\mathbb{G}_\lambda$. Let $\{\mathcal{X}_\lambda\}$ be a family of well-spread distributions over $\{0,1\}^\lambda$. Then for any $\ell = \mathtt{poly}(\lambda)$ for any PPT $\mathcal{A}$:*

$$\left|\Pr[\mathcal{A}(\{k_i, g^{k_i x + x^i}\}_{i\in[2,\ldots,\ell]} = 1] - \Pr[\mathcal{A}(\{k_i, g^{k_i r + r^i}\}_{i\in[2,\ldots,\ell]}\right| = \mathtt{ngl}(\lambda).$$

where $x \leftarrow \mathcal{X}_\lambda, r \leftarrow \mathbb{Z}_{p(\lambda)}, k_i \leftarrow \mathbb{Z}_{p(\lambda)}$.

The second assumption can be proved from Assumption 3.1, see [BMZ19, Lemma 8], and is useful for arguing nonmalleability:

Assumption 3.2. *[BMZ19, Assumption 4] Let $\mathcal{G}$ and $\mathcal{X}_\lambda$ be defined as in Assumption 3.1. Then for any $\ell = \mathtt{poly}(\lambda)$ for any PPT $\mathcal{A}$:*

$$\Pr[g^x \leftarrow \mathcal{A}(\{k_i, g^{k_i x + x^i}\}_{i\in[2,..,\ell]} = \mathtt{ngl}(\lambda).$$

where $x \leftarrow \mathcal{X}_\lambda$ and $k_i \leftarrow \mathbb{Z}_{p(\lambda)}$.

We now introduce our main construction. The intuition behind the construction is to increase the number of randomized powers to deal with the additional constraints on val that the adversary gains by seeing multiple copies; it will be proved secure under Assumptions 3.1 and 3.2.

Construction 3.1. *Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a group ensemble with efficient representation and operations where each $\mathbb{G}_\lambda$ is a group of prime order $p \in (2^\lambda, 2^{\lambda+1})$. We assume that for every $\lambda \in \mathbb{N}$ there is a canonical and efficient mapping between the elements of $\{0,1\}^\lambda$ to $\mathbb{G}_\lambda$. Let g be a generator of the group $\mathbb{G}_\lambda$. For some parameter $\tau \in \mathbb{Z}^+$, let $\vec{a}, \vec{b}, \vec{c} \xleftarrow{\$} \mathbb{G}_\lambda$ be input randomness and define the algorithm* $\mathsf{lockPoint}$ *as:*

$$\mathsf{lockPoint}(\mathsf{val}; \vec{a}, \vec{b}, \vec{c}) \stackrel{def}{=} \begin{bmatrix} \vec{a}, & g^{\sum_{i=1}^{\tau} \vec{a}_i x^i + \sum_{i=\tau+1}^{\tau+5} x^i} \\ \vec{b}, & g^{\vec{b}_1 x + \sum_{i=2}^{\tau} \vec{b}_i x^{i+\tau+4} + x^{2\tau+5}}, \\ \vec{c}, & g^{\vec{c}_1 x + \sum_{i=2}^{\tau} \vec{c}_i x^{i+2\tau+4} + x^{3\tau+5}}. \end{bmatrix}$$

Given a program $\mathsf{unlockPoint}$ *consisting of three vectors $\vec{a}, \vec{b}, \vec{c}$ and group elements g_1, g_2, g_3 and input* val *compute:*

$$g^{\sum_{i=1}^{\tau} \vec{a}_i \mathsf{val}^i + \sum_{i=\tau+1}^{\tau+5} \mathsf{val}^i} \stackrel{?}{=} g_1$$
$$g^{\vec{b}_1 \mathsf{val} + \sum_{i=2}^{\tau} \vec{b}_i \mathsf{val}^{i+\tau+4} + \mathsf{val}^{2\tau+5}} \stackrel{?}{=} g_2$$
$$g^{\vec{c}_1 \mathsf{val} + \sum_{i=2}^{\tau} \vec{c}_i \mathsf{val}^{i+2\tau+4} + \mathsf{val}^{3\tau+5}} \stackrel{?}{=} g_3.$$

If all of these checks pass, output 1*. Otherwise, output* 0*.*

In order to add same point composability, we extend from three scalars to 3τ scalars (while keeping 3 group elements). We note that this scheme is that of [BMZ19] if we let $\tau = 1$.

Lemma 3.1. *For any $\tau = \mathtt{poly}(\lambda)$ Construction 3.1 satisfies completeness.*

Proof. This argument is analogous to the functionality preservation argument in [BMZ19]. The only difference is that polynomials are higher degree due to composition. Fix some point $x \in \mathbb{Z}_{p(\lambda)}$. It suffices to argue that over the randomness of $\mathsf{unlockPoint} \leftarrow \mathsf{lockPoint}(x)$ that the probability that there exists some y such that $\mathsf{unlockPoint}(y) = 1$ is $\mathtt{ngl}(\lambda)$.

Recall that the randomness used to construct $\mathsf{unlockPoint}$ is the vectors $\vec{a}, \vec{b}, \vec{c}$. Fix some $x \in \mathbb{Z}_{p(\lambda)}$. Fix some value $\vec{a}$ and define $\alpha \stackrel{def}{=} \sum_{i=1}^{\tau} x^i + \sum_{i=\tau+1}^{\tau+5} x^i$. For some other value y, since $\mathcal{G}$ is prime order the only way for the first element to match is for $\alpha = \sum_{i=1}^{\tau} y^i + \sum_{i=\tau+1}^{\tau+5} y^i$. Since this is a polynomial of degree $\tau + 5$ there are at most $\tau + 4$ such values y (excluding the original value x). Consider one such value y. Then, consider the polynomial $P(\vec{b}) \stackrel{def}{=} \vec{b}_1(x - y) + \sum_{i=2}^{\tau} \vec{b}_i(x^{i+\tau+4} - y^{i+\tau+4}) + (x^{2\tau+5} - y^{2\tau+5})$. Fix some values of $\vec{b}_i$ for $i = 2, ..., \tau$. Then this is a linear polynomial in $\vec{b}_1$ that is zero with probability at most $1/p(\lambda)$. A similar argument holds for the second check value. Thus, a candidate y is a solution to both equations with probability $1/p(\lambda)^2$. Thus means for a fixed x the probability of one of the y's working is at most $(\tau + 5)/p(\lambda)^2$ by union bound. With a second application of union bound, the probability across all x of some y existing is at most $(\tau + 5)/p(\lambda) = \mathtt{ngl}(\lambda)$ as desired. □

Theorem 3.1. *Suppose that Assumption 3.1 holds. Then for any $\tau = \mathtt{poly}(\lambda)$, Construction 3.1 satisfies virtual black box security (when composed up to τ times).*

Proof. We show that Construction 3.1 satisfies distributional indistinguishability (Definition 2.4). Virtual black box security then follows by Theorem 2.1.

Suppose for the aim of arriving at a contradiction that there exists some well-spread distribution $\mathcal{X}_\lambda$ such that there exists a PPT adversary $\mathcal{A}$ and a polynomial $q(\cdot)$ such that

$$|\Pr[\mathcal{A}(\{\mathsf{lockPoint}(x)\}_{i=1}^{\tau}) = 1] - \Pr[\mathcal{A}(\{\mathsf{lockPoint}(r)\}_{i=1}^{\tau}) = 1]| > \frac{1}{q(\lambda)},$$

where $x \leftarrow X_\lambda$ and $r \leftarrow \mathbb{Z}_{p(\lambda)}$. We then show how to build an adversary $\mathcal{B}$ that breaks Assumption 3.1 (with respect to distribution family $\mathcal{X}_\lambda$) receiving $\ell = 3\tau + 4$ elements (corresponding to a maximum power of $3\tau + 5$). That is, $\mathcal{B}$ will receive $3\tau + 4$ pairs of the form

$$\{k_i, g^{k_i z + z^i}\}_{i \in \{2,\ldots,3\tau+5\}},$$

where z is either distributed according to $\mathcal{X}_\lambda$ or uniformly in $\mathbb{Z}_{p(\lambda)}$. Denote by $\{k_i, g^{h_i}\}_{i=2,\ldots,3\tau+5}$ the received values, defining $h_i = k_i z + z^i$. Then, $\mathcal{B}$ samples three matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ uniformly in $\mathbb{Z}_{p(\lambda)}^{\tau \times (\tau-1)}$.

Our goal is to produce τ obfuscations (either of x or r). $\mathcal{B}$ compute the matrices $\mathbf{A}', \mathbf{B}', \mathbf{C}' \in \mathbb{Z}_{p(\lambda)}^{\tau \times \tau}$ as follows:

$$\mathbf{A}'_{i,j} = \begin{cases} \sum_{\alpha=1}^{\tau-1} \mathbf{A}_{i,\alpha} k_{\alpha+1} + \sum_{\alpha=\tau}^{\tau+4} k_{\alpha+1} & j = 1 \\ \mathbf{A}_{i,j-1} & \text{otherwise.} \end{cases}$$

$$\mathbf{B}'_{i,j} = \begin{cases} \sum_{\alpha=1}^{\tau-1} \mathbf{B}_{i,\alpha} k_{\alpha+\tau+5} + k_{2\tau+5} & j = 1 \\ \mathbf{B}_{i,\alpha-1} & \text{otherwise.} \end{cases}$$

$$\mathbf{C}'_{i,j} = \begin{cases} \sum_{\alpha=1}^{\tau-1} \mathbf{C}_{i,\alpha} k_{\alpha+2\tau+5} + k_{3\tau+5} & j = 1 \\ \mathbf{C}_{i,j-1} & \text{otherwise.} \end{cases}$$

Then $\mathcal{B}$ computes the ith value to be fed into $\mathcal{A}$ as:

$$\mathsf{lockPoint}_i = \begin{cases} \mathbf{A}'_{(i,\cdot)}, & g^{\sum_{j=1}^{\tau-1} \mathbf{A}_{i,j} h_{j+1} + \sum_{j=t}^{\tau+5} h_{j+1}}, \\ \mathbf{B}'_{(i,\cdot)}, & g^{\sum_{j=1}^{\tau-1} \mathbf{B}_{i,j} h_{j+\tau+4} + h_{2\tau+5}}, \\ \mathbf{C}'_{(i,\cdot)}, & g^{\sum_{j=1}^{\tau-1} \mathbf{C}_{i,j} h_{j+2\tau+4} + h_{3\tau+5}}. \end{cases}$$

The above group elements can be formed linearly from the received values $\{k_i, g^{h_i}\}_{i \in [2,\ldots,3\tau+5]}$ and $\mathbf{A}, \mathbf{B}, \mathbf{C}$. For the ith obfuscation, the values produced in the exponent are (omitting the exponential notation):

$$\sum_{j=1}^{\tau-1} \mathbf{A}_{i,j} h_{j+1} + \sum_{j=\tau+1}^{\tau+5} h_j = \left(\sum_{j=1}^{\tau-1} \mathbf{A}_{i,j} k_{j+1} + \sum_{j=\tau}^{\tau+4} k_{j+1} \right) z + \sum_{j=1}^{\tau-1} \mathbf{A}_{i,j} z^{j+1} + \sum_{j=\tau+1}^{\tau+5} z^j,$$

$$\sum_{j=1}^{\tau-1} \mathbf{B}_{i,j} h_{j+\tau+5} + h_{2\tau+5} = \left(\sum_{j=1}^{\tau-1} \mathbf{B}_{i,j} k_{i+\tau+5} + k_{2\tau+5} \right) z + \sum_{j=1}^{\tau-1} \mathbf{B}_{i,j} z^{j+\tau+5} + z^{2\tau+5},$$

$$\sum_{j=1}^{t-1} \mathbf{C}_{i,j} h_{j+2\tau+5} + h_{3\tau+5} = \left(\sum_{j=1}^{\tau-1} \mathbf{C}_{i,j} k_{i+2\tau+5} + k_{3\tau+5} \right) z + \sum_{j=1}^{\tau-1} \mathbf{C}_{i,j} z^{j+2\tau+5} + z^{3\tau+5}.$$

From the above equations, it is apparent that the matrices $\mathbf{A}', \mathbf{B}', \mathbf{C}'$ are consistent with the group elements. Furthermore it is clear for $j > 1$ that the coefficients for z^j are appropriately formed. It remains to show that the 3τ coefficients of z are uniformly random. Denote by ζ_i for $i = 1, ..., 3\tau$ coefficients of z respectively. Let $\mathbf{1}^{i,j}$ represent an all 1 matrix of dimension $i \times j$ and define $\mathbf{0}^{i \times j}$ similarly. Define the matrix of coefficients:

$$\mathbf{D} \stackrel{def}{=} \left(\begin{array}{c|c|c|c|c|c|c} \mathbf{A}_{(\cdot,1)} & \mathbf{A}_{(\cdot,2,\ldots,t-1)} & \mathbf{1}^{\tau\times 5} & \mathbf{0}^{\tau\times\tau-5} & \mathbf{0}^{\tau\times 1} & \mathbf{0}^{\tau\times\tau-1} & \mathbf{0}^{\tau\times 1} \\ \hline \mathbf{B}_{(\cdot,1)} & \mathbf{0}^{\tau\times\tau-1} & \mathbf{0}^{\tau\times 5} & \mathbf{B}_{(\cdot,2,\ldots,t-1)} & \mathbf{1}^{\tau\times 1} & \mathbf{0}^{\tau\times\tau-1} & \mathbf{0}^{\tau\times 1} \\ \hline \mathbf{C}_{(\cdot,1)} & \mathbf{0}^{\tau\times\tau-1} & \mathbf{0}^{\tau\times 5} & \mathbf{0}^{\tau\times\tau-5} & \mathbf{0}^{\tau\times 1} & \mathbf{C}_{(\cdot,2,\ldots,t-1)} & \mathbf{1}^{\tau\times 1} \end{array}\right).$$

The set of values received by the adversary can be described by:

$$\mathbf{D}\begin{bmatrix} k_2 \\ k_2 \\ \ldots \\ k_{3t+5} \end{bmatrix} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \ldots \\ \zeta_{3t} \end{bmatrix}$$

The matrix $\mathbf{D}$ has dimension $3\tau \times 3\tau + 5$. For each coefficient ζ_j to be random it suffices for the matrix $\mathbf{D}$ to have row rank of 3τ. For $\mathbf{D}$ to have rank 3τ it suffices for each $\mathbf{A}||\mathbf{1}, \mathbf{B}|\mathbf{1}, \mathbf{C}||\mathbf{1}$ to have rank of τ. Since each matrix is random this occurs with probability at most $\tau/p = \mathtt{ngl}(\lambda)$. If one these matrices is not full rank, $\mathcal{B}$ aborts and outputs a random value. Conditioning on these matrices being full rank the obfuscation are properly prepared for $\mathcal{A}$. Denote by

$$\mathtt{Disting}_{\mathcal{A}} \stackrel{def}{=} |\Pr[\mathcal{A}(\{\mathsf{lockPoint}(x)\}_{i=1}^{\tau}) = 1] - \Pr[\mathcal{A}(\{\mathsf{lockPoint}(r)\}_{i=1}^{\tau}) = 1]|.$$

Then one has that

$$\begin{aligned} \left|\Pr[\mathcal{B}(\{k_i, g^{k_i x + x^i}\}_{i\in[2,\ldots,3\tau+4]} = 1] - \Pr[\mathcal{B}(\{k_i, g^{k_i r + r^i}\}_{i\in[2,\ldots,3\tau+4]}\right| = \\ \Pr[\mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \text{ not full rank}] + \Pr[\mathbf{A} \wedge \mathbf{B} \wedge \mathbf{C} \text{ full rank}] \cdot \mathtt{Disting}_{\mathcal{A}} = \\ \mathtt{ngl}(\lambda) + (1 - \mathtt{ngl}(\lambda)\frac{1}{q(\lambda)} = \frac{1}{q'(\lambda)} \end{aligned}$$

for some polynomial function $q'(\lambda)$. This completes the proof of Theorem 3.1. □

Theorem 3.2. *Let λ be a security parameter Let $\{\mathcal{X}_\lambda\}$ be a well-spread distribution ensemble and let $m, \tau \in \mathbb{Z}^+$ be parameters that are both $\mathtt{poly}(\lambda)$. Let $\mathcal{F}_{poly}$ be the ensemble of functions f_λ where f_λ is the set of non-constant, non-identity polynomials in $\mathbb{Z}_{p(\lambda)}[x]$ with degree at most m. Suppose that Assumption 3.1 holds for $\ell = m(3\tau + 5)$. Then, the above obfuscator is non-malleable for τ-compositions for $\mathcal{F}_{poly}$ and distribution ensemble $\{\mathcal{X}_\lambda\}$.*

Proof. We look to contradict Assumption 3.2, which follows from Assumption 3.1. Consider a mauling adversary $\mathcal{A}$ that, given τ obfuscations of a point x, can output a new obfuscation of $f(x)$ for $f \in \mathcal{F}_{poly}$. Consider m to be the degree of f. We build an adversary $\mathcal{B}$ which given the ensemble $\{k_i, g^{k_i x + x^i}\}_{i=2\ldots,m(3\tau+5)}$ and access to $\mathcal{A}$ recovers g^x with noticeable probability.

First, we consider the case when $m > 1$. We set up the reduction as so: upon receiving the ensemble $\{k_i, g^{k_i x + x^i}\}_{i=2\ldots,m(3\tau+5)}$, we create τ obfuscations of x as detailed in Theorem 3.1. We send these to $\mathcal{A}$, which returns

$$(f, \vec{a}, \vec{b}, \vec{c}, j_a, j_b, j_c)$$

where $\vec{a}, \vec{b}, \vec{c} \in \mathbb{Z}_{p(\lambda)}^{\tau}$ and $j_a, j_b, j_c \in \mathbb{G}_\lambda$. Define the vector $\vec{l}$ as the coefficients of:

$$\vec{c}_1(f(x)) + \sum_{i=2}^{\tau} \vec{c}_i (f(x))^{i+2\tau+4} + (f(x))^{3\tau+5} = \sum_{i=0}^{m(3\tau+5)} \vec{l}_i x^i.$$

In order for the adversary to succeed, this value must equal the exponent of j_c with noticeable probability. $\mathcal{B}$ computes and returns

$$\left(j_c \left(g^{l_0} \cdot \prod_{i=2}^{m(3\tau+5)} h_i^{l_i} \right)^{-1} \right)^{1/(l_1 - \sum_{i=2}^{m(3\tau+5)} k_i l_i)}.$$

Since $\mathcal{B}$ has properly prepared the set of obfuscations to $\mathcal{A}$, $\mathcal{A}$ returns a valid obfuscation of $f(x)$ with probability at least $1/\mathtt{poly}(\lambda)$. In this case then

$$j_c = g^{l_0 + l_1 x + \ldots + l_{m(3\tau+3)} x^{m(3\tau+5)}}$$

with the same probability. In this case, we see that the value in parenthesis is

$$g^{x(l_1 - \sum_{i=2}^{m(3\tau+5)} k_i l_i)}.$$

Since all l_i, k_i are known, this can be computed unless

$$l_1 - \sum_{i=2}^{m(3\tau+5)} k_i l_i = 0.$$

Since $f(x)$ is of degree m, $l_{m(3\tau+5)}$ must be nonzero. $\mathcal{A}$'s view is independent of $k_{m(3\tau+5)}$. So, the probability that the sum is equal to l_1 is $1/(p(\lambda) - 1)$. So, $\mathcal{B}$ returns the correct value with probability $1/\mathtt{poly}(\lambda) - 1/(p(\lambda) - 1) = 1/\mathtt{poly}(\lambda)$ contradicting Assumption 3.2.

We now consider the case where $m = 1$, or for linear functions f. In this case, we are given the ensemble $\{k_i, g^{k_i x + x^i}\}_{i=2,\ldots,3\tau+5}$. This time, upon receiving

$$(f, \vec{a}, \vec{b}, \vec{c}, j_a, j_b, j_c)$$

from $\mathcal{A}$, $\mathcal{B}$ instead computes the coefficients $\vec{l}$ of

$$\sum_{i=1}^{\tau} \vec{a}_i f(x)^i + \sum_{i=\tau}^{\tau+5} f(x)^i = \sum_{i=0}^{\tau+5} \vec{l}_i x^i$$

as in the nonlinear case. In this case, $\mathcal{B}$ computes and outputs:

$$\left(j_a \left(g^{l_0} \cdot \prod_{i=2}^{\tau+5} h_i^{l_i} \right)^{-1} \right)^{1/(l_1 - \sum_{i=2}^{\tau+5} k_i l_i)}.$$

Because $\mathcal{A}$ outputs the value

$$g^{\sum_{i=1}^{\tau} \vec{a}_i f(x)^\tau + (f(x))^{\tau+1} + \ldots + (f(x))^{\tau+5}}$$

with noticeable probability, $\mathcal{B}$'s computation evaluates to g^x unless

$$l_1 - \sum_{i=2}^{\tau+5} k_i l_i = 0.$$

Let $\mathbf{R}$ be a random $\mathbb{Z}_{p(\lambda)}^{\tau \times \tau-1}$ and let $\mathbf{1}^{\tau \times 5}$ be a $\tau \times 5$ matrix of all 1s. To see that this happens with negligible probability, for the first group element of each obfuscation received the coefficient of x^1 are as follows:

$$\vec{a}_1 = \left[\mathbf{R} | \mathbf{1}^{\tau \times 5}\right] \cdot \begin{pmatrix} k_2 \\ k_3 \\ \dots \\ k_{\tau+5} \end{pmatrix}$$

As shown in the proof of Theorem 3.1 the values of R are uniformly random conditioned on the other values seen by the adversary.

We note that, as all k_i are uniformly chosen, the only information $\mathcal{A}$ learns about $k_{\tau+1}, ..., k_{\tau+5}$ is in the vector $\vec{a}_1$ Furthermore $\mathbf{R}$ is independent of these values. Thus, we can see that $\mathcal{A}$ receives items of the form

$$\vec{a}_{1,j} = \sum_{i=2}^{\tau} k_i \mathbf{R}_{i,j} + \sum_{i=\tau+1}^{\tau+5} k_i.$$

Without loss of generality, we assume that an adversary knows the values $k_2, ..., k_\tau$. To change the obfuscated point they will also need to change the higher order powers $x^{\tau+1}, ..., x^{\delta+5}$. The only value they have seen that involves the values $k_{\tau+1}, ..., k_{\tau+5}$ are terms of the form $c + \left(\sum_{i=1}^{5} k_{\tau+i}\right) x$ for some value c. Since the function is linear, we can represent $f(x) = \alpha x + \beta$. So, the adversary must find α, β, γ such that

$$\sum_{i=0}^{4} (\alpha x + \beta)^{i+\tau+1} = \gamma \sum_{i=0}^{4} x^{i+\tau+1}.$$

Define $\delta = \tau + 1$. We can write the desired linear combination as follows:

$$\begin{bmatrix} \alpha^{4+\delta} \\ \alpha^{3+\delta}\left(\binom{\delta+4}{1}\beta + \binom{\delta+3}{0}\right) \\ \alpha^{2+\delta}\left(\binom{\delta+4}{2}\beta^2 + \binom{\delta+3}{1}\beta + \binom{\delta+2}{0}\right) \\ \alpha^{1+\delta}\left(\sum_{i=0}^{3}\left(\binom{\delta+4-i}{3-i}\beta^{3-i}\right)\right) \\ \alpha^{\delta}\left(\sum_{i=0}^{4}\left(\binom{\delta+4-i}{4-i}\beta^{4-i}\right)\right) \end{bmatrix}^{\intercal} \begin{bmatrix} k_{4+\delta} & 0 & 0 & 0 & 0 \\ 0 & k_{3+\delta} & 0 & 0 & 0 \\ 0 & 0 & k_{2+\delta} & 0 & 0 \\ 0 & 0 & 0 & k_{1+\delta} & 0 \\ 0 & 0 & 0 & 0 & k_\delta \end{bmatrix} = \gamma \begin{bmatrix} k_{4+\delta} \\ k_{3+\delta} \\ k_{2+\delta} \\ k_{1+\delta} \\ k_\delta \end{bmatrix}$$

Substituting, one has that

1. If $\beta = 0$ then this implies $\alpha^{\delta+4} = \alpha^{\delta+3} = \alpha^{\delta+2} = \alpha^{\delta+1} = \alpha^\delta$ which only has solutions if $\alpha = 0$ or $\alpha = 1$. These are both considered trivial solutions.
2. Otherwise, $\gamma = \alpha^{\delta+4}$ (using first equation),
3. $(\delta + 4)\beta + 1 = \alpha$ (using second equation),
4. $(\delta + 4)\beta + 2 = 0$ or $\delta = -5$ (using third equation, relying on $\beta \neq 0$).
5. Assume that $\delta \neq -5$, then $\alpha = -1$ (substitution of third constraint into second equation)

$\mathsf{lock}(\mathsf{val}, \mathsf{key})$:

1. Sample $r \leftarrow \mathbb{Z}_{p(\lambda)}$.
2. Compute $\vec{z}_1 = \mathsf{lockPoint}(\mathsf{val})$.
3. For $i = 1$ to n:
   (a) If $\mathsf{key}_i = 1$, set $\mathsf{unlockPoint}_{i+1} = \mathsf{lockPoint}(\mathsf{val})$.
   (b) Else, set $\vec{z}_{i+1} = \mathsf{lockPoint}(r)$.
4. Output $\vec{z}$.

$\mathsf{unlock}(\{z_i\}_{i=1}^{n+1}, \mathsf{val})$:

1. If $\mathsf{unlockPoint}_1(\mathsf{val}) = \perp$ output $\perp$.
2. Initialize $\mathsf{key} = \vec{0}$.
3. For $i = 1$ to n:
   (a) If $\mathsf{unlockPoint}_{i+1}(\mathsf{val}) \neq \perp$ set $\mathsf{key}_i = 1$.
4. Output key.

Figure 1: Nonmalleable digital locker preventing tampering over only val

6. $\gamma = (-1)^\delta$ (substitution of α in first equation). Note that $\gamma = 1$ corresponds to no tampering. Thus, we consider $\gamma = -1$.
7. $\delta \equiv -5$ or $\delta \equiv -6$ (solving fourth equation using prior constraints) and thus $\tau \equiv -6$ or $\tau \equiv -7$.

We note that since $\tau = \mathtt{poly}(\lambda)$ for large enough λ one can be sure that $\tau \not\equiv \{-6, -7\} \mod |\mathbb{G}_\lambda|$. So, the only functions that $\mathcal{A}$ can maul to are the constant and identity functions, neither of which are in $\mathcal{F}_{poly}$. This means that $\mathcal{A}$ returns a solution where $l_1 - \sum_{i=2}^{\tau+5} k_i l_i = 0$. with negligible probability. So, with non-negligible probability, $\mathcal{B}$ can break Assumption 3.2. □

4 Nonmalleable digital lockers

We now use the nonmalleable point function from Construction 3.1 to construct a nonmalleable digital locker that does not prevent any tampering over the stored key. We use the well known real-or-random construction of digital-lockers [CD08]. The basic real or random construction is in Figure 1. We do not argue security of this basic construction, as long as $\mathsf{lockPoint}$ is $n + 1$ same point composable then this construction provides a digital locker that provides nonmalleability over val. The argument is the same as in [CD08] with the worst case for nonmalleability being when all of key is 1 since this provides the adversary with $n + 1$ obfuscations of val.

4.1 Detecting tampering over key

With the ability to instantiate the real or random construction with nonmalleable point functions, we turn to detecting tampering over the encoded key. As mentioned in the introduction, this construction requires a public object that all parties can depend on (as long as the distribution is efficiently sampleable) which can be achieved in the CRS model. However, the construction is black box in the underlying digital locker (unlike a construction from NIZKs).

We combine nonmalleable codes and seed-dependent condensers to check if the adversary tampers over the key value. We use the locked point val as input to a seed-dependent condenser as part of the value encoded in the nonmalleable code. If the adversary tampers to an *independent value*, they are unlikely to match the output of the condenser on the real val. We introduce these tools and then our

construction. We first present the notion of nonmalleable codes, introduced by Dziembowski, Pietrzak, and Wichs [DPW10].

Definition 4.1. *A pair of algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *is called a* coding scheme *if* $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n$ *is randomized and* $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k \cup \perp$ *is deterministic and for each* $s \in \{0,1\}^k$ *it holds that* $\Pr[\mathsf{Dec}(\mathsf{Enc}(s)) = s] = 1$.

Definition 4.2. *A coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is called* $(\epsilon_{nmc}, s_{nmc}, \mathcal{F})$-nonmalleable *if for each* $f \in \mathcal{F}$ *and each* $s \in \{0,1\}^k$*, there exists a distribution* $D_f()$ *over* $\{\{0,1\}^k, \mathtt{same}\}$ *that is efficiently samplable given oracle access to* f *such that the following holds:*

$$\delta^{s_{nmc}}(\{c \leftarrow \mathsf{Enc}(s); \overline{c} \leftarrow f(c), \overline{s} = \mathsf{Dec}(\overline{c}) : \textit{Output } \overline{s}\}, \quad \{\tilde{s} \leftarrow D_f, \ \textit{Output } s \textit{ if } \tilde{s} = \mathtt{same} \textit{ else } \tilde{s}\}) \leq \epsilon_{nmc}.$$

Seed-dependent condensers were introduced by Dodis, Ristenpart, and Vadhan [DRV12]. Their goal is similar to a traditional randomness extractor, except the output only has to be statistically close to a distribution with minentropy. Importantly, it is possible to construct condensers where the adversary is allowed to output the chosen distribution after seeing the seed.

Definition 4.3. *Let* $\mathtt{cond} : \{0,1\}^\lambda \times \{0,1\}^d \to \{0,1\}^\alpha$ *be a* (k, k', s, ϵ) seed-dependent condenser *if for all probabilistic adversaries of size at most* s *who take a random seed* $\mathsf{seed} \leftarrow U_d$ *and output a distribution* $X_{\mathsf{seed}} \leftarrow \mathcal{A}(\mathsf{seed})$ *of entropy* $\mathrm{H}_\infty(X|\mathsf{seed}) \geq k$*, then for the joint distribution* (X, U_d) *over* X_{seed} *arising from a random* $\mathsf{seed} \leftarrow U_d$*, there exists a distribution* Y *such that* $\tilde{\mathrm{H}}_\infty(Y|U_d) \geq k'$ *such that* $\Delta((Y, U_d), (\mathtt{cond}(X; U_d), U_d)) \leq \epsilon$.

Dodis, Ristenpart, and Vadhan showed that seed-dependent condensers can be constructed using collision resistant hash functions. Furthermore, this construction works for $\epsilon = 0$. That is, the output has entropy instead of being close to a distribution with entropy. For our construction, we will require $k' = \omega(\log \lambda)$.

The encoding We now present the construction. Instead of directly *locking* the value key we instead lock the value

$$c = \mathsf{Enc}(\mathsf{key} || \mathtt{cond}(\mathsf{val}; \mathsf{seed})),$$

where Enc is the encoding function for a nonmalleable code and $\mathtt{cond}$ is a seed dependent condenser. Notionally, the nonmalleable code prevents tampering to *independent* points and the condenser detects if the adversary tampers to an independent point.

Construction 4.1. *Let* $(\mathsf{lock}', \mathsf{unlock}')$ *be defined as in Figure 1. Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a coding scheme where* $\mathsf{Enc} : \{0,1\}^{k+\alpha} \to \{0,1\}^n$*. Let* $\mathtt{cond} : \{0,1\}^\lambda \times \{0,1\}^d \to \{0,1\}^\alpha$ *be a seed-dependent condenser. Define the algorithms* $(\mathsf{lock}', \mathsf{unlock}', \mathsf{V}_{\mathsf{key}})$ *as in Figure 2.*

However, security of this construction is not straightforward as we are using nonmalleable codes in a nonstandard way. In a nonmalleable code, the adversary specifies the tampering function before seeing any information about c. In our setting, the adversary sees obfuscations that have c embedded before deciding how to tamper. The crucial part to our argument is that the set of obfuscations is pseudorandom condition on c and $s \overset{def}{\mathtt{cond}}(\mathsf{val}; \mathsf{seed})$. If an adversary is able to tamper substantially better given obfuscations of val from some entropic distribution than with uniformly random val we can check whether they tampered properly and use this to break distributional indistinguishability.

Theorem 4.1. *Let* $\lambda \in \mathbb{N}$ *be a security parameter and let* $\{0,1\}^\lambda$ *be the domain. Let* $(\mathsf{lockPoint}, \mathsf{unlockPoint})$ *be a* $(n+1)$*-same point composable and* $\mathcal{F}_{single}$*-nonmalleable.*

$\mathsf{lock}'(\mathsf{val}, \mathsf{key})$,
input in $\{0,1\}^{\lambda+k}$:

1. Compute $z = \mathtt{cond}(\mathsf{val}, \mathsf{seed})$.
2. Compute $y = \mathsf{Enc}(\mathsf{key}||z)$.
3. Output $\mathsf{lock}(\mathsf{val}, y)$.

$\mathsf{unlock}'(\mathsf{val}') \stackrel{def}{=} \mathsf{unlock}(\mathsf{val}')$

$\mathsf{V}_{\mathsf{key}}(\mathsf{val}', y)$,
input in $\{0,1\}^{\lambda+n}$:

1. Compute $z = \mathtt{cond}(\mathsf{val}', \mathsf{seed})$.
2. Run decode $\mathsf{key}' = \mathsf{Dec}(y)$.
3. If $\mathsf{key}'_{k\ldots k+n} \neq z$ output $\perp$.
   Else output $\mathsf{key}'_{0,\ldots,k-1}$.

Figure 2: Nonmalleable digital locker preventing tampering over both val and key. A seed of a seed-dependent condenser must be public and global.

1. *Suppose for any $s = \mathtt{poly}(\lambda)$ there exists $\mu, \beta = \omega(\log \lambda)$ such $\mathtt{cond} : \{0,1\}^\lambda \times \{0,1\}^d \to \{0,1\}^\alpha$ is a $(\mu, \beta, s, 0)$-seed-dependent condenser.*
2. *Let $\mathsf{seed} \leftarrow \{0,1\}^d$ be a public parameter.*
3. *$X \stackrel{def}{=} X(\mathsf{seed})$ be an s-samplable distribution so $\tilde{\mathrm{H}}_\infty(X|\mathsf{seed}, \mathtt{cond}(\mathsf{seed}, X)) \geq \beta$.*[3]
4. *Let a description of $\mathbb{G}_\lambda$, a generator g for $\mathbb{G}_\lambda$ and $\mathsf{seed} \leftarrow \{0,1\}^d$ be system parameters.*
5. *Let $\mathcal{F}_{nmc}$ be a function class. Suppose for any $s_{nmc} = \mathtt{poly}(\lambda)$ there exists $\epsilon_{nmc} = \mathtt{ngl}(\lambda)$ such that $(\mathsf{Enc}, \mathsf{Dec})$ is an $(\epsilon_{nmc}, s_{nmc}, \mathcal{F}_{nmc})$ nonmalleable code.*

Then $(\mathsf{lock}', \mathsf{unlock}')$ in Construction 4.1 and Figure 4.1 is point nonmalleable for $\mathcal{F}_{single}$ and key nonmallable for $\mathcal{F}_{nmc}$. In particular, $(\mathsf{lock}', \mathsf{unlock}')$ is a $(\mathcal{F}_{single}, \mathcal{F}_{nmc}, \mathcal{X})$-nonmalleable digital locker.

Before proving Theorem 4.1 we make some notes about the construction and alternative approaches.

Choice of nonmalleable code We recommend using a nonmalleable code that detects at least permutations and $1 \to 0$ bit tampers, such as [AGM+15a, AGM+15b], as these transforms are otherwise easily computable in polynomial time. One concern about nonmalleable codes is that the adversary is *necessarily* restricted to low complexity classes, not including the code's encoding and decoding functions. Note, the construction encodes and decodes the code "in the clear" while the adversary is tampering "in the exponent."

Recently, Kiayias et al. [KLT18] introduced *nonmalleable codes with manipulation detection*. Here, the adversary has low probability of producing any codeword $\tilde{c}$ that successfully decodes. Kiayias et al. constructed a nonmalleable code with manipulation detection, unfortunately, their standard model result requires logarithmic length symbols. Furthermore, the primitive cannot exclude constant functions. It is an open problem if this primitive can simplify our construction.

Alternative tools Constructions using nonmalleable extractors [DW09, CRS14] or one-way hashes [BCFW09, BFS11, CQZ+16] may be possible. However, they are not immediate, we use the primitive of nonmalleable hashes to illustrate. A nonmalleable hash function is a family of functions $h \in \mathcal{H}$ such that an adversary given $h(x)$ (sampled $h \leftarrow \mathcal{H}$ and x from some distribution) cannot find $h(f(x))$ for f in some

[3] In the previous sections, we consider X that have worst case min-entropy. However, if $\tilde{\mathrm{H}}_\infty(X|\mathsf{seed}, \mathtt{cond}(\mathsf{seed}, X)) \geq \beta$ for some $\beta = \omega(\log \lambda)$ then there exists some $\beta' = \omega(\log \lambda)$ such that with $\Pr_{\mathsf{seed}}[\mathrm{H}_\infty(X|\mathsf{seed}, \mathtt{cond}(\mathsf{seed}, X)) \geq \beta'] \geq 1 - \mathtt{ngl}(\lambda)$. Thus, this change does not effect the set of distributions assumed to be secure in Assumption 3.1.

function class $\mathcal{F}$. Several of these works claim to be "standard model" but all require h is random and not tampered by the adversary. One could append a nonmalleable hash, obfuscating $\mathsf{key}' = \mathsf{key}||h(\mathsf{key}||\mathsf{val})$. The concerns are:

1. This approach assumes that the function instance h is assumed to be independently sampled from key and val. In our approach, the public randomness required is for $\mathtt{seed}$ of the condenser, and the distribution of val (and key) can depend on this value.
2. Non malleable hashes are analyzed with the adversary only knowing the output value $h(x)$. It is not clear that security would hold in the presence of multiple correlated obfuscations.

Similar issues arise with nonmalleable extractors [DW09, CRS14].

4.2 Proof of Theorem 4.1

Completeness and soundness follow from the underlying properties of nonmalleable point function. (The only additional issue with correctness is if in the real or random construction the sampled value $r = \mathsf{val}$ this occurs with negligible probability for any input val.)

Since the point function satisfies VBB when composed $n+1$ times it also satisfies $(n+1)$-distributional indistinguishability [Var10, Theorem 3.18]. Suppose that ϵ_{pr} is the negligible function for the distributional indistinguishability condition and ϵ_{nm} is the negligible function for the nonmalleable condition of the point function. We will show that the overall malleability denoted by ϵ is at most

$$\epsilon \leq \epsilon_{pr} + \epsilon_{nm} + 2^{-\beta} + \epsilon_{nmc}.$$

Let $\mathcal{M}$ be a mauling adversary for the overall digital locker. Note that for f output by $\mathcal{M}$ the probability that $f \in \mathcal{F}_{single}$ and f is not the identity is at most ϵ_{nm}. Suppose not, then there is some key^* where this is the case. One can construct a mauling adversary $\mathcal{M}'$ for lock by:

1. Sampling a random seed,
2. Computing $z = \mathtt{cond}(\mathsf{val}, \mathsf{seed})$, $y = \mathsf{Enc}(\mathsf{key}^*||z)$,
3. Running the real or random construction with $\mathsf{key} \stackrel{def}{=} z$ and val,
4. Outputting $\mathcal{M}$'s outputs when provided with the real-or-random values.

Thus, if the probability that $\mathcal{M}$ succeeds in breaking the nonmalleability condition is greater than ϵ this implies that it outputs f that is the identity and $g \in \mathcal{F}_{nmc}$ and not the identity with probability greater than $\epsilon - \epsilon_{nm} = \epsilon_{pr} + 2^{-\beta} + \epsilon_{nmc}$.

We now show that this contradicts the security of the nonmalleable code. The key difference between traditional application of nonmalleable codes is that the adversary is provided with a set of values $\{\mathsf{lockPoint}_i\}_{i=1}^n$ that are correlated with the encoded value c before deciding how to tamper. We show that this is not a problem with the following argument:

1. The values $\{\mathsf{lockPoint}_i\}_{i=1}^n$ are pseudorandom conditioned on the encoded value c. (This is the case in the real or random construction.)
2. There is a mechanism to check if the decoded value $\tilde{s} = \mathsf{Dec}(c')$ is good. Here we use matching the output of the seed dependent condenser.

3. If the adversary is provided with random obfuscations (uncorrelated to c) and replaces c with some new value c', by security of the nonmalleable code, c' depends on c with low probability and if c' is independent of c, $\tilde{s} = \mathsf{Dec}(c')$ passes the test with low probability.
4. If the adversary tampers and passes the test with higher probability with the pseudorandom values, we can use this to break the pseudorandomness condition.

To make the above intuition formal we define an adaptive tampering experiment as follows for random variables $\mathcal{LP}, C, \mathcal{CD}$ and binary predicate $\mathtt{Test}$:

Experiment $\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F}_{nmc},\mathcal{LP},C,\mathcal{CD},\mathcal{A},\mathtt{Test}}$:
Sample $(obf, c, cond) \leftarrow (\mathcal{LP}, C, \mathcal{CD})$
Sample $f \leftarrow \mathcal{A}(obf)$.
If $f \notin \mathcal{F}_{nmc}$ output 0.
If $f(c) = c$ output 0.
If $\mathtt{Test}(f(c), cond)$ output 1.
Else output 0.

We define $\mathcal{LP}, C, \mathcal{CD}, \mathtt{Test}$ as

- $\mathcal{LP}$: the set of obfuscations output by the real or random construction,
- $\mathcal{CD}$ the output of the seed dependent condenser on val.
- C: the encoded codeword $c = \mathsf{Enc}(\mathsf{key}||\mathtt{cond}(\mathsf{val};\mathsf{seed}))$.
- $\mathtt{Test}$: if $\mathsf{Dec}(c)_{k\ldots k+\alpha} \stackrel{?}{=} z$.

We now show that in the above experiment the adversary can't tamper substantially better than when specifying the tampering function with no information.

Theorem 4.2. *Let $\mathcal{CD} \in \{0,1\}^\alpha$ be a distribution such that $\mathrm{H}_\infty(\mathcal{CD}) \geq \beta$. Let $(\mathsf{Enc}, \mathsf{Dec})$ be a $(\epsilon_{nmc}, s_{nmc}, \mathcal{F})$ nonmalleable code and $\mathsf{Enc} : \{0,1\}^{k+\alpha} \to \{0,1\}^n$. Let $\mathsf{key} \in \{0,1\}^k$, define the distribution C_{key} by sampling $cond \leftarrow \mathcal{CD}$ and computing $\mathsf{Enc}(\mathsf{key}||cond)$. For inputs c and $cond$, define $\mathtt{Test}(c, cond) = 1$ if and only if $\mathsf{Dec}(y)_{k\ldots(k+\alpha)} = cond$. Suppose that*

1. *For all $f \in \mathcal{F}$ it is possible to compute f in a circuit of size at most $s_{\mathcal{F},\mathtt{eval}}$.*
2. *It is possible to evaluate $\mathtt{Test}$ using a circuit of size at most $|\mathtt{Test}|$ and $s_{nmc} > |\mathtt{Test}|$.*
3. *For a function f it is possible to check if $f \in \mathcal{F}$ in size at most $s_{\mathcal{F},\mathtt{check}}$. Furthermore, this check is correct with probability 1.*
4. *$\mathcal{LP}$ be an distribution over $\mathcal{M}$ that is correlated to C_{key} and $\mathcal{CD}$ and let R be a sampleable distribution independent of of $\mathcal{LP}, C_{\mathsf{key}}, \mathcal{CD}$ such that*

$$\delta^{\mathcal{D}_{s_{pr}}}((\mathcal{LP}, C_{\mathsf{key}}, \mathcal{CD}), (R, C_{\mathsf{key}}, \mathcal{CD})) \leq \epsilon_{pr}.$$

Then for all $\mathcal{A}$ of size s it holds that

$$\Pr\left[\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}} = 1\right] \leq 2^{-\beta} + \epsilon_{nmc} + \epsilon_{pr}.$$

Here $s = \min\{s_{pr} - |\mathtt{Test}| - s_{\mathcal{F},\mathtt{eval}} - s_{\mathcal{F},\mathtt{check}}, s_{nmc}\}$.

Before proving Theorem 4.2, we show how Theorem 4.1 follows from Theorem 4.2. Let $\mathcal{A}$ be some adversary. Let $W = W(\mathsf{seed})$ be some efficiently distribution such that

$$\tilde{\mathrm{H}}_\infty(W|\mathsf{seed}, \mathtt{cond}(\mathsf{seed}, W)) \geq \beta.$$

Define the random variables $\mathcal{LP} \stackrel{def}{=} \mathsf{lock}'(W, C)$. With overwhelming probability, the conditional distribution $\mathcal{LP}|C, \mathcal{CD}$ is in the class $\mathcal{X}$. By $(n+1)$-distributional indistinguishability of the point function one has that for any PPT $\mathcal{A}$,

$$\delta^{\mathcal{D}_{s_{pr}}}((\mathcal{LP}, C_{\mathsf{key}}, \mathcal{CD}), (R, C_{\mathsf{key}}, \mathcal{CD})) \leq \epsilon_{pr}.$$

Thus, the assumptions of Theorem 4.2 are satisfied. One can then conclude that

$$\Pr\left[\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}} = 1\right] \leq 2^{-\beta} + \epsilon_{nmc} + \epsilon_{pr}.$$

It remains to show that for all $\mathcal{A}, \mathsf{key}$

$$\Pr_{\mathsf{val}\leftarrow X}\left[\begin{array}{r|l} \mathcal{V}(C) = 1, f \in \mathcal{F}, & \\ g \in \mathcal{G}, y = C(f(\mathsf{val})), & \\ y = g(\mathsf{unlock}(f(\mathsf{val}))), & (C, f, g) \leftarrow \mathcal{A}(\mathsf{lock}(\mathsf{val}, \mathsf{key})) \\ \mathsf{V}_{\mathsf{key}}(f(\mathsf{val}), y) \neq \perp, & \\ \exists \alpha \text{ s.t. } I_{f(\mathsf{val}),\alpha} \equiv C & \end{array}\right] \leq 2^{-\beta} + \epsilon_{nmc} + \epsilon_{pr},$$

when f is the identity function and g is not. Towards the sake of arriving at a contradiction, suppose that there exists some $\mathcal{A}$ that wins the above game with probability greater than $\epsilon' = 2^{-\beta} + \epsilon_{nmc} + \epsilon_{pr}$. Note that $\mathtt{Test}$ outputs 1 with the same probability as $\mathsf{V}_{\mathsf{key}}$ outputting a value other than $\perp$. If the above is true, it must hold for some key^*. Thus, we can build an adversary $\mathcal{A}'$ that breaks the conditions of Theorem 4.2 for the distribution $\mathcal{LP} \stackrel{def}{=} \mathsf{lock}(\mathsf{val}, \mathsf{key})$ where val is sampled from X by:

1. Receiving lock
2. Running $\mathcal{A}$ on input lock.
3. Receive outputs C, f, g.
4. Output g. (Corresponding to a tampering function on c.)

This adversary $\mathcal{A}'$ is a valid adversary for $\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}}$. This completes the proof of how Theorem 4.1 follows from Theorem 4.2. We now proceed to the proof of Theorem 4.2.

Proof of Theorem 4.2. We begin by defining a standard nonmalleable code experiment with a simulator for a function f defined by a distribution $D_f(\cdot)$:

Experiment $\mathbf{Exp}^{\mathrm{sim}}_{f,\mathcal{CD},D_f}$:
Sample $\tilde{s} \leftarrow D_f(\cdot), cd \leftarrow \mathcal{CD}$
If $\tilde{s} = \mathtt{same}$ output 0.
If $\mathtt{Test}(\tilde{s}, cd) = 1$ output 1.
Else output 0.

Lemma 4.1. *Suppose that* $\mathrm{H}_\infty(\mathcal{CD}) \geq \beta$*, for any* f*,* $\Pr[\mathbf{Exp}^{\mathrm{sim}}_{f,Z,D_f}(k) = 1] \leq 2^{-\beta}$.

Proof of Lemma 4.1. We note that whenever $\tilde{s} = \texttt{same}$ the output of the experiment is 0. Thus, we can restrict our attention to cases when $\tilde{s} \neq \texttt{same}$. Then $\Pr[\mathcal{CD} = \tilde{s}_{k-\alpha \ldots k}] \leq 2^{-\mathrm{H}_\infty(\mathcal{CD})} = 2^{-\beta}$. This completes the proof of Lemma 4.1. □

We will now argue that the adversary in the adaptive adversary does not perform substantially better than in the simulated experiment. We use a hybrid argument with two intermediate games, $\mathbf{Exp}^1_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}}$ and $\mathbf{Exp}^2_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}}$. In moving from $\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}}$ to $\mathbf{Exp}^1_{\mathcal{F},R,Y,Z,\mathcal{A}}$ we will replace the distribution $\mathcal{LP}$ with a distribution R that is uncorrelated to $C, \mathcal{CD}$. In $\mathbf{Exp}^2_{f,C,\mathcal{CD}}$ we will eliminate the distribution R as input and move from the adversary picking a function to defining the experiment for a particular function f. Finally in moving to $\mathbf{Exp}^{\text{sim}}_{f,\mathcal{CD},D_f}(k)$ we will rely on the hardness of nonmalleable codes. The two experiments are described formally below.

Experiment $\mathbf{Exp}^1_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}}$:	**Experiment** $\mathbf{Exp}^2_{f,C,\mathcal{CD}}$:
Sample $(c, cd) \leftarrow (C, \mathcal{CD})$	Sample $(c, cd) \leftarrow (C, \mathcal{CD})$
Sample $r \leftarrow R$.	If $f(c) = c$ output 0.
Sample $f \leftarrow \mathcal{A}(r)$.	Output $\texttt{Test}(f(c), cd)$.
If $f \notin \mathcal{F}$ output 0.	
If $f(y) = y$ output 0.	
Output $\texttt{Test}(f(c), cd)$.	

We now show each of these games are computationally close.

Lemma 4.2. *Suppose that*

1. *For each $f \in \mathcal{F}$, the function f is computable in size at most $s_{\mathcal{F}}$.*
2. *For f it is possible to correctly check $f \in \mathcal{F}$ in size $s_{\mathcal{F},\texttt{check}}$.*
3. *That $\delta^{\mathcal{D}_{s_{pr}}}((\mathcal{LP}, C_{\mathsf{key}}, \mathcal{CD}), (R, C_{\mathsf{key}}, \mathcal{CD})) \leq \epsilon_{pr}$.*

Then for $\mathcal{A}$ of size at most $s_{pr} - |\texttt{Test}| - s_{\mathcal{F},\texttt{eval}} - s_{\mathcal{F},\texttt{check}}$,

$$\left|\Pr\left[\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}}(k) = 1\right] - \Pr[\mathbf{Exp}^1_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}}(k) = 1]\right| \leq \epsilon_{pr}.$$

Proof of Lemma 4.2. Suppose not. That is, suppose that there exists an $\mathcal{A}$ of size at most $s_{pr} - |\texttt{Test}| - s_{\mathcal{F},\texttt{eval}} - s_{\mathcal{F},\texttt{check}}$ such that

$$\left|\Pr\left[\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}}(k) = 1\right] - \Pr[\mathbf{Exp}^1_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}}(k) = 1]\right| > \epsilon_{pr}.$$

Then the following program D (of size at most s_{pr}) is a distinguisher for $((\mathcal{LP}, C, \mathcal{CD})$ and $(R, C, \mathcal{CD}))$:

1. On input p, c, cd.
2. Run $f \leftarrow \mathcal{A}(p)$.
3. If $f \notin \mathcal{F}$ or $f(c) = c$ output 0.
4. Else output $\texttt{Test}(f(c), cd)$.

That is,

$$\begin{aligned}&|\Pr[D(\mathcal{LP}, C, \mathcal{CD}) = 1] - \Pr[D(R, C, \mathcal{CD}) = 1]| \\ &= \left|\Pr[\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F},\mathcal{LP},C,\mathcal{CD},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{1}_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}} = 1]\right| > \epsilon_{pr}.\end{aligned}$$

This contradicts the indistinguishability of $(\mathcal{LP}, C, \mathcal{CD})$ and $(R, C, \mathcal{CD})$ and completes the proof of Lemma 4.2. □

Lemma 4.3. *There exists some $f \in \mathcal{F}$ such that for any $\mathcal{A}$ (here $\mathcal{A}$ need not be computationally bounded):*

$$\Pr[\mathbf{Exp}^{2}_{f,C,\mathcal{CD}}(k) = 1] \geq \Pr[\mathbf{Exp}^{1}_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}}(k) = 1].$$

Proof of Lemma 4.3. First we consider the circuits $\mathcal{A}$ that always output $f \in \mathcal{F}$. Given any $\mathcal{A}$ that outputs an $f \notin \mathcal{F}$ we can design another $\mathcal{A}'$ that runs $f \leftarrow \mathcal{F}$ and simply outputs a fixed $f' \in \mathcal{F}$ whenever $f \notin \mathcal{F}$. This $\mathcal{A}'$ does not perform worse in $\mathbf{Exp}^1$ than $\mathcal{A}$.

Now consider some $\mathcal{A}$ that always outputs functions $f \in \mathcal{F}$. There is a distribution $D_{\mathcal{A}}$ that outputs exactly the distribution that is output by $\mathcal{A}$. Note that this distribution is independent of y. Note that

$$\Pr[\mathbf{Exp}^{1}_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}} = 1] = \sum_{f \in \mathcal{F}} \Pr\left[D_{\mathcal{A}} = f\right] \Pr_{(c,cd) \leftarrow (C,\mathcal{CD})}\left[f(c) \neq c \wedge \mathtt{Test}(c, cd) = 1\right].$$

Now suppose that for all $f \in \mathcal{F}$,

$$\Pr[\mathbf{Exp}^{2}_{f,C,\mathcal{CD}}(k) = 1] < \Pr[\mathbf{Exp}^{1}_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}}(k) = 1].$$

Then one has

$$\begin{aligned}\Pr[\mathbf{Exp}^{1}_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}} = 1] &= \sum_{f \in \mathcal{F}} \Pr\left[D_{\mathcal{A}} = f\right] \Pr_{(c,cd) \leftarrow (C,\mathcal{CD})}\left[f(c) \neq c \wedge \mathtt{Test}(c, cd) = 1\right] \\ &= \sum_{f \in \mathcal{F}} \Pr\left[D_{\mathcal{A}} = f\right] \Pr[\mathbf{Exp}^{2}_{f,C,\mathcal{CD}} = 1] \\ &< \sum_{f \in \mathcal{F}} \Pr\left[D_{\mathcal{A}} = f\right] \Pr[\mathbf{Exp}^{1}_{\mathcal{F},R,C,\mathcal{CD},\mathcal{A}} = 1] \\ &= \Pr[\mathbf{Exp}^{1}_{\mathcal{F},C,\mathcal{CD},\mathcal{A}} = 1]\end{aligned}$$

This is a contradiction and completes the proof of Lemma 4.3. □

Before showing distinguishability of the last two games we consider the definition of nonmalleable codes where the encoded secret is drawn from a distribution instead of considering a single point:

Lemma 4.4. *Let $(\mathsf{Enc}, \mathsf{Dec})$ be a $(\epsilon_{nmc}, s_{nmc})$-nonmalleable code for functions in $\mathcal{F}$. Then for any distribution Z over points in $\{0,1\}^k$ it holds that*

$$\begin{aligned}\delta^{\mathcal{D}_{s_{nmc}}}\big(&(\{c \leftarrow \mathsf{Enc}(Z); \overline{c} \leftarrow f(c), \overline{s} = \mathsf{Dec}(\overline{c}) : \textit{Output } \overline{s}\}, Z), \\ &(\{\tilde{s} \leftarrow D_f, \ \textit{Output } Z \textit{ if } \tilde{s} = \mathtt{same} \textit{ else } \tilde{s}\}, Z)\big) \\ &\leq \epsilon_{nmc}.\end{aligned}$$

Proof of Lemma 4.4. Suppose not, that is there exists some Z for which the statement is not true. In particular, there must be some $z \in Z$ where $\Pr[Z = z] > 0$ such that there exists $\mathcal{D}_{s_{nmc}}$,

$$\begin{aligned}\delta^{\mathcal{D}_{s_{nmc}}}(\,&(\{c \leftarrow \mathsf{Enc}(z); \overline{c} \leftarrow f(c), \overline{s} = \mathsf{Dec}(\overline{c}) : \text{Output } \overline{s}\}, z)\,,\\ &(\{\tilde{s} \leftarrow D_f,\ \text{Output } z \text{ if } \tilde{s} = \texttt{same} \text{ else } \tilde{s}\}, z)\\ &> \epsilon_{nmc}.\end{aligned}$$

This contradicts security of the nonmalleable code. □

With this distributional version of security for nonmalleable codes we can turn to indistinguishability of the last two games.

Lemma 4.5. *For every $f \in \mathcal{F}$, if $s_{nmc} \geq |Test|$ then*

$$\left|\Pr[\mathbf{Exp}^{\text{sim}}_{f,\mathcal{CD},D_f} = 1] - \Pr[\mathbf{Exp}^{2}_{f,C,\mathcal{CD}} = 1]\right| \leq \epsilon_{nmc}.$$

Proof of Lemma 4.5. Suppose not, that is suppose that there exists some $f \in \mathcal{F}$ such that

$$\left|\Pr[\mathbf{Exp}^{\text{sim}}_{f,\mathcal{CD},D_f}(k) = 1] - \Pr[\mathbf{Exp}^{2}_{f,C,\mathcal{CD}}(k) = 1]\right| > \epsilon_{nmc}.$$

Then we have a distinguisher D (of size $|\texttt{Test}|$) for the distributional version of nonmalleable code security guarantee 1) input $\tilde{s}, z$ and 2) Compute $\texttt{Test}(\tilde{s}, z)$. This completes the proof of Lemma 4.5. □

Combining Lemmas 4.1, 4.2, 4.3 and 4.5 completes the proof of Theorem 4.2 and completes the proof of Theorem 4.1. □

Acknowledgements

This work was funded in part by a grant from Comcast Inc, by NSF Grant CNS 1849904, and ONR Grant N00014-19-1-2327. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19020700008. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Governement. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

The authors thank Luke Demarest, Pratyay Mukherjee, Alex Russell, and Mayank Varia for their helpful feedback. Special thanks to James Bartusek, Fermi Ma, and Mark Zhandry for discussing their work and its compositional properties.

References

[ABC+18] Quentin Alamélou, Paul-Edmond Berthier, Chloé Cachet, Stéphane Cauchie, Benjamin Fuller, Philippe Gaborit, and Sailesh Simhadri. Pseudoentropic isometries: A new framework for fuzzy extractor reusability. In *AsiaCCS*, 2018.

[AGM+15a] Shashank Agrawal, Divya Gupta, Hemanta K Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology – CRYPTO*, pages 538–557. Springer, 2015.

[AGM+15b] Shashank Agrawal, Divya Gupta, Hemanta K Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In *Theory of Cryptography Conference*, pages 375–397. Springer, 2015.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology – CRYPTO*, pages 308–326. Springer, 2015.

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In *Advances in Cryptology–CRYPTO 2010*, pages 520–537. Springer, 2010.

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 524–541. Springer, 2009.

[BFS11] Paul Baecher, Marc Fischlin, and Dominique Schröder. Expedient non-malleability notions for hash functions. In *Cryptographers Track at the RSA Conference*, pages 268–283. Springer, 2011.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in Cryptology–CRYPTO*, pages 1–18. Springer, 2001.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. *Journal of the ACM (JACM)*, 59(2):6, 2012.

[BMZ19] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. In *Advances in Cryptology – CRYPTO*, 2019.

[BR17] Zvika Brakerski and Guy N Rothblum. Obfuscating conjunctions. *Journal of Cryptology*, 30(1):289–320, 2017.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology-CRYPTO'97*, pages 455–469. Springer, 1997.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *Advances in Cryptology–EUROCRYPT 2008*, pages 489–508. Springer, 2008.

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *Annual International Cryptology Conference*, pages 693–721. Springer, 2018.

[CFP+16] Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam Smith. Reusable fuzzy extractors for low-entropy distributions. In *Advances in Cryptology–Eurocrypt 2016*, pages 117–146. Springer, 2016.

[CQZ+16] Yu Chen, Baodong Qin, Jiang Zhang, Yi Deng, and Sherman SM Chow. Non-malleable functions and their applications. In *IACR International Workshop on Public Key Cryptography*, pages 386–416. Springer, 2016.

[CRS14] Gil Cohen, Ran Raz, and Gil Segev. Nonmalleable extractors with short seeds and applications to privacy amplification. *SIAM Journal on Computing*, 43(2):450–476, 2014.

[CRV10] Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *Theory of Cryptography Conference*, pages 72–89. Springer, 2010.

[CV09] Ran Canetti and Mayank Varia. Non-malleable obfuscation. In Omer Reingold, editor, *Theory of Cryptography*, pages 73–90, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, volume 2010, page 1st. Citeseer, 2010.

[DRV12] Yevgeniy Dodis, Thomas Ristenpart, and Salil Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In *Theory of Cryptography Conference*, pages 618–635. Springer, 2012.

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 601–610. ACM, 2009.

[DY13] Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. In *Theory of Cryptography Conference*, pages 1–22. Springer, 2013.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE, 2013.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 151–170. IEEE, 2015.

[KLT18] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Non-malleable codes for partial functions with manipulation detection. In *Advances in Cryptology – CRYPTO*, 2018.

[KY18a] Ilan Komargodski and Eylon Yogev. Another step towards realizing random oracles: Non-malleable point obfuscation. In *Advances in Cryptology – EUROCRYPT*, pages 259–279. Springer, 2018.

[KY18b] Ilan Komargodski and Eylon Yogev. Another step towards realizing random oracles: Non-malleable point obfuscation. Cryptology ePrint Archive, Report 2018/149, 2018. Version 20190226:074205, `https://eprint.iacr.org/2018/149`.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology – CRYPTO*, pages 500–517. Springer, 2014.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 2014.

[Var10] Mayank Harshad Varia. *Studies in program obfuscation*. PhD thesis, Massachusetts Institute of Technology, 2010.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 600–611. IEEE, 2017.