Nonmalleable Digital Lockers for Efficiently Sampleable Distributions

Peter Fenteany* Benjamin Fuller†

May 29, 2019

Abstract

An obfuscated program reveals nothing about its design other than its input/output behavior. A digital locker is an obfuscated program that outputs a stored cryptographic key if and only if a user enters a previously stored password. A digital locker provides an adversary no information on either with high probability. An ideal digital locker also detects if an adversary mauls one obfuscation into a valid obfuscation of related values. Such a primitive is achievable in the random oracle model or using general purpose non-interactive zero knowledge proofs of knowledge (NIZKPoK) in the common random string (CRS) model. However, there are no known standard model constructions of nonmalleable digital lockers.

Komargodski and Yogev (Eurocrypt, 2018) constructed a simpler primitive: a nonmalleable point function. Security relies on variants of the strong and power DDH assumptions.

This work describes the first nonmalleable digital locker without resorting to generic NIZKPoKs. This construction is built in three steps:

1. We construct a nonmalleable digital locker for short keys from any sufficiently nonmalleable point function.
2. We introduce a new primitive called stocky lockers that augments this primitive to retain security if composed with the same password and maintain pseudorandomness. We show Komargodski and Yogev's construction suffices to build stocky lockers under the average-case variants of the same assumptions. This composed construction is nonmalleable with respect to the password.
3. We extend to polynomial length keys that additionally provides nonmalleability over the stored key. This extension combines the digital locker for short keys, nonmalleable codes, and seed-dependent condensers. The seed for the condenser must be public and not tampered.

The third step can be achieved in the CRS model. The password distribution can depend on the condenser's seed as long as it is efficiently sampleable. We view this as a midpoint in removing the CRS. Prior works on nonmalleable hashes/one-way functions/extractors also assume a public random object that is not tampered by the adversary. The other steps in our construction do not require a CRS.

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

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

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

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

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

Digital Lockers Our focus is on *digital lockers*. 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. Privacy says $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}$ should reveal nothing about val or key if the adversary cannot guess val. A digital locker is also known as a multi-bit point obfuscation [CD08, BC10]. Digital lockers have applications in passwords [Can97] and biometric authentication [CFP+16, ABC+18].

A simpler object to construct is a *point function* $\mathsf{unlockPoint}_{\mathsf{val}}$. Upon creation, $\mathsf{unlockPoint}_{\mathsf{val}}$ stores val, and from then on indicates when val exactly is inputted to it. An obfuscated point function only needs to hide val [Can97]. It is possible to compose point functions to build a digital locker (if the point function retains security when composed) [CD08]. Traditionally, digital lockers have been constructed using the *real-or-random* construction. It works as so: For each bit of the key, either a random point (corresponding to a 0 in key) or val (corresponding to a 1) is obfuscated producing $\mathsf{unlockPoint}_{y_i}$. When running the program, the user runs each $\mathsf{unlockPoint}_{y_i}$, and the program will either fail when a random point was obfuscated or succeed when the true point was obfuscated.

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 $\mathsf{unlockPoint}_{\mathsf{val}}$ from being mauled to $\mathsf{unlockPoint}_{f(\mathsf{val})}$. 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 $\mathsf{RO}(\mathsf{key})$ is truncated.

While such nonmalleable objects are obvious in some models, Komargodski and Yogev constructed the first nonmalleable point obfuscator in the standard model [KY18]. Their construction follows. Let g be a fixed group generator. To obfuscate the point val, the obfuscator computes a random r and outputs

$$\mathcal{O}(\mathsf{val}) = (r, r^{g^{h(\mathsf{val})}}), \text{ where } h(z) = z^4 + z^3 + z^2 + z.$$

Note the nonstandard use of double exponentiation, $g^{h(\mathsf{val})}$ is first computed and interpreted as an exponent for the base group in which r resides. The function h is designed specifically to prevent mauling. Bartusek, Ma, and Zhandry [BMZ18] also showed a nonmalleable point function using random a, b, c:

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

Both constructions rely on strong non-standard variants of the Decisional Diffie-Hellman (DDH) assumption [DH76]. Bartusek, Ma, and Zhandry [BMZ18] show security of their assumptions in the auxiliary input generic group model [CDG18]. In both constructions, g is assumed to be fixed; this means that the distribution val may depend on generator g. For an obfuscated point val, these constructions detect if the

adversary creates an obfuscation of $f(\mathsf{val})$ from an obfuscation of val for polynomials f of degree related to the assumed hardness in the DDH assumptions.

The goal of this work is to construct nonmalleable digital lockers. A first attempt applies the real-or-random construction using nonmalleable point functions like those above. However, that approach does not ensure nonmalleability over the bits of key. It is easy to permute bits of key by reordering group elements and to set individual bits of key to 0 by replacing a group element by a random group element. Furthermore, these constructions are not known to be composable, and they require strong DDH variants for one-time security. Consider a case when a user encrypts their files with key. If the adversary can create $\mathsf{lock}_{\mathsf{val},\mathsf{key}'}(\cdot)$, the user may use some cryptographic key that is known to the adversary or susceptible to cryptanalytic attack [BCM11].

1.1 Our Contribution

We construct nonmalleable digital lockers in three steps:

Section 3 We show nonmalleable point obfuscators can be converted to *nonmalleable digital lockers* with a single bit output b (in Theorem 3.1). Let b be the bit key, we obfuscate $\mathsf{lockPoint}(2 \cdot \mathsf{val} + b)$. Opening proceeds by running $\mathsf{unlockPoint}$ with both values of b. This object is nonmalleable for val as long as the tampering family $\mathcal{F}$ is closed under multiplication by $2, 2^{-1}$ and addition and subtraction of 1 (Theorem 3.1). We introduce this new construction for two reasons:

1. With the real-or-random construction, an adversary can easily force an obfuscation of a random point. This ensures that each point has only two points that evaluate to useful outputs.
2. It may be possible to extend the construction to longer keys. The real-or-random model cannot be extended to other alphabets. In Section 5, we present a construction that supports log bits in a pair of group elements.

Section 4 Ideally, one could compose a single bit digital locker to build a full digital locker. Unfortunately, obfuscation is not generically composable. Instead, we introduce a new type of nonmalleable digital locker we call *stocky lockers*. A stocky locker augments a nonmalleable digital locker with a short output with the following properties:

1. It is composable (soundness and nonmalleability hold) as long as the same val is used as input in each invocation.
2. A vector of obfuscations of val is indistinguishable from a vector of pseudorandom values.

Stocky lockers make no attempt to provide nonmalleability over key.

We show the construction of Komargodski and Yogev [KY18] directly yields a 1-bit stocky locker. The core of the argument is showing nonmalleability for val when the adversary is given multiple copies of $\mathsf{stockLock}(\mathsf{val}, 0)$ and $\mathsf{stockLock}(\mathsf{val}, 1)$.

Unfortunately, the construction of Bartusek et al. [BMZ18] is not composable for the same val, as all the higher order terms can be removed by dividing two instances. That is, given $a_1, g_1 = g^{a_1 x + x^2 + x^3 + x^4 + x^5}$ and $a_2, g_2 = g^{a_2 x + x^2 + x^3 + x^4 + x^5}$ one can easily compute $g^{(a_1 - a_2)x} = g_1/g_2$ and $g^x = (g_1/g_2)^{(a_1 - a_2)^{-1}}$. The hardness of finding g^x is the underlying assumption used to show nonmalleability [BMZ18, Assumption 4]. It is an open question if the construction of Bartusek et al. can be modified to be self-composable.

As mentioned above, we present a second stocky locker in Section 5 under similar assumptions. This construction supports logarithmic length keys. Nonmalleability for this construction also relies on a

power type DDH assumptions. The essence of the nonmalleability argument is arguing that an unseen val lies in a linearly independent space so it is hard to predict. However, since the adversary may have access to more points (corresponding to different key) the points they "see" form a higher dimensional space. To overcome this problem in our new construction we replace h with a new function where for each val, different values of key lie in a low dimensional subspace. Let $\tau = |\mathsf{key}|$, the new h is:

$$h(\mathsf{val}, \mathsf{key}) = \left(\sum_{i=\tau}^{4+\tau} (\mathsf{val}+2)^i\right) + \left(\sum_{i=0}^{\tau-1} \mathsf{key}_j \cdot (\mathsf{val}+2)^i\right).$$

Section 6 We present a generic construction of a nonmalleable digital locker from a stocky locker. This construction ensures nonmalleability over both val and key. This step uses several cryptographic tools and requires a public random object. We make no requirement that the distribution of key or val is independent of this object but it must be public, random, and not modified. These guarantees can be achieved in the common random string (CRS) model. We remark on the model below and then explain the used cryptographic techniques.

Remark: A nonmalleable digital locker can be constructed from a digital locker using generic non-interactive zero-knowledge proofs of knowledge (NIZKPoK) in the common random string (CRS) model. Boldyreva et al. considered this idea to build a nonmalleable hash function [BCFW09]. A nonmalleable hash function is a family of functions $h \in \mathcal{H}$ such that an adversary given $h(x)$ (sampled $h \leftarrow \mathcal{H}$) cannot find $h(f(x))$ for f in some function class $\mathcal{F}$. Subsequent constructions removed general purpose non-interactive zero-knowledge [BFS11] [CQZ+16]. Several of these works claim to be "standard model" but all require h is random and not tampered by the adversary. Similar issues arise with nonmalleable extractors [DW09, CRS14].

Importantly, our construction does not assume independence of distributions from the random object or any programmability in the proof. Furthermore, nonmalleable digital lockers have more stringent security definitions than nonmalleable hashes (such as leaking no partial information). We view our construction as a midpoint towards removing the CRS. We also stress the CRS is only necessary for preventing tampering of key. *The stocky locker construction is in the standard model.*

Key authentication Composing stocky lockers enable a construction of a digital locker that places each bit of key in a separate stocky locker. This approach does not detect mauling of key. One could prove knowledge of key to keep the adversary from modifying the program. Alternatively, one could append a nonmalleable hash, obfuscating $\mathsf{key}' = \mathsf{key}||h(\mathsf{key}||\mathsf{val})$. However, there are two concerns with this approach:

1. This approach assumes that the function instance h is assumed to random and independently sampled from key and val.
2. Any information leaked by $h(\mathsf{key}||\mathsf{val})$ makes it more difficult to show security of the digital locker.

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 authenticate. In more detail:

1. We compute $\mathsf{cond}(\mathsf{val}; \mathsf{seed})$ where cond is a seed-dependent condenser. A seed-dependent condenser has a high entropy output even if the distribution over val is adversarially dependent on the randomness seed. Seed-dependent condensers for efficiently sampleable distributions are instantiable

using collision-resistant hash functions [DRV12, Theorem 4.1]. We note that val can be correlated to seed as long as it is efficiently sampleable. Importantly, seed does need to be randomly sampled and be publicly known.

We then compute $s = \mathsf{key}||\mathtt{cond}(\mathsf{val};\mathsf{seed})$. An adversary outputting any fixed $\tilde{s}$ that does not depend on the received obfuscated programs is unlikely to match the output of the condenser.

2. To ensure that an adversary is limited to "independent" tampering, we use a nonmalleable code. Let $\mathcal{F}$ be some function class. A nonmalleable code is a pair Enc and Dec where for functions $f \in \mathcal{F}$ the value $\tilde{s} = \mathsf{Dec}(f(\mathsf{Enc}(s)))$ is independent of s. We set

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

In a nonmalleable code, the adversary specifies the tampering function before seeing any information about $\mathsf{Enc}(s)$. In our setting, the adversary sees obfuscations correlated to $\mathsf{Enc}(s)$ before deciding how to tamper. We show that nonmalleable codes can be used in a nonstandard way where the tampering function is chosen after seeing the obfuscated values.

nonmalleable codes provide guarantees when the adversary tampers "obliviously." We show a technical result that the adversary's success probability when given the obfuscations does not deviate from oblivious tampering. This also provides obvious modularity in choosing a code.

Constructions using nonmalleable extractors [DW09, CRS14] or one-way hashes [BCFW09, BFS11, CQZ+16] (in place of the nonmalleable code) seem possible. In our approach, the only public randomness required is for $\mathtt{seed}$ of the condenser. Using a nonmalleable extractor or hash seems to require a second public value (the extractor seed or hash function description). Furthermore, security definitions for these primitives do not consider the case when the distribution of val depends on the description of the function.

Bitwise nonmalleable codes We recommend using a nonmalleable code that detects at least permutations [AGM+15a] [AGM+15b]. This is important for ensuring nonmalleability of key, as permutations 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."

Nonmalleable codes with manipulation detection 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. (Clearly, the class of tampering functions cannot contain constant functions.) Kiayias et al. constructed a nonmalleable code with manipulation detection, but their construction requires each symbol of the code to come from a polynomial-sized alphabet or equivalently for each symbol to have logarithmic length. We show in Section 5 a new stocky locker with multiple output bits. With this strengthened object our construction does not need the seed-dependent condenser. However, Kiayias et al.'s construction does not exclude functions which are efficiently computable against our construction such as permutations. If stronger nonmalleable codes with manipulation detection are discovered, we recommend using such codes. Such a result would yield a fully nonmalleable digital locker in the standard model.

Open Questions We present two main open questions resulting from this work. The first is whether stocky lockers can be generically construction from nonmalleable point functions. The second is whether the CRS can be removed from the full construction.

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

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

The *statistical distance* between random variables X and Y with the same domain is

$$\Delta(X, Y) = \frac{1}{2}\sum_x |\Pr[X = x] - \Pr[Y = x]|.$$

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 $C \equiv C'$.

Definition 2.1. *An ensemble of joint distributions $(\mathcal{X}, \mathcal{Y}) = \{X_\lambda, Y_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{X}_\lambda$ is over $\{0,1\}^\lambda$, is* average case 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, Y_λ).*
2. *For all large enough $\lambda \in \mathbb{N}$, it has super-logarithmic conditional minentropy. Namely, $\tilde{\mathrm{H}}_\infty(X_\lambda|Y_\lambda) = \omega(\log \lambda)$.*

3 Nonmalleable Obfuscation Definitions

All obfuscation definitions include a requirement of *polynomial slowdown*, which is omitted for space considerations. 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 directly adapt the definition of nonmalleability from Komargodski and Yogev. We do note their definition requires the adversary that is performing the mauling to output the mauling function f. See Komargodski and Yogev for definitional considerations [KY18].

The following definitions use *virtual grey-box obfuscation*, which means the simulator is allowed unbounded time but a limited number of queries. Bitanski and Canetti [BC10] showed virtual grey-box obfuscation and virtual black-box obfuscation are equivalent for digital lockers and point functions. Frequently in this work, we use distributional versions of soundness, which are also equivalent for point functions and digital lockers [Can97].

Our constructions require that the challenger can recognize a legitimate obfuscation.

Definition 3.1 (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 $\mathcal{V}_\mathcal{O}$ for program class $\mathcal{O}$ is called a* verifier *if all $x \in \{0,1\}^\lambda$, it holds that $\Pr[\mathcal{V}_\mathcal{O}(\mathcal{P}) = 1 | \mathcal{P} \leftarrow \mathcal{O}(x)] = 1$, (prob. over the randomness of $\mathcal{V}$ and $\mathcal{O}$).*

Our constructions consist of tuples of group elements and strings. As such, the obvious verifier suffices.

Point Obfuscators A point function is a function I_{val}: $\{0,1\}^n \mapsto \{0,1\}$ that outputs 1 on input x and 0 elsewhere. An obfuscator preserves functionality while hiding the point val if val is not provided as input to the program.

Definition 3.2 (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 $\mathcal{V}$ be a verifier for $\mathsf{lockPoint}$ as defined in Definition 3.1. The following properties must hold:*

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

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

   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 (possibly inefficient) 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 $\mathcal{P} : \{0,1\}^\lambda \mapsto \{0,1\}$,*

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

   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 $I_x(\cdot)$ is an oracle that returns 1 when provided input x and 0 otherwise.

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

$$\Pr_{\mathsf{val} \leftarrow X} \left[\mathcal{V}(C) = 1, f \in \mathcal{F}, (I_{f(\mathsf{val})} \equiv C) | (C, f) \leftarrow \mathcal{A}(\mathsf{lockPoint}(\mathsf{val})) \right] \leq \epsilon.$$

Single Bit Digital Locker We introduce a new primitive in order to create nonmalleable digital lockers. We will build a (nonmalleable) single bit digital locker: $I_{\mathsf{val},b}$: $\{0,1\}^n \mapsto \{\perp, 0, 1\}$. Here $I_{\mathsf{val},b}(\mathsf{val}) = b$ and $I_{\mathsf{val},b}(\mathsf{val}') = \perp$ for all other points $\mathsf{val}' \neq \mathsf{val}$.

Definition 3.3 (Single Bit Digital Locker)**.** *For security parameter $\lambda \in \mathbb{N}$, let $\mathcal{F} : \{0,1\}^{\lambda-1} \to \{0,1\}^{\lambda-1}$ be a family of functions, let $\mathcal{X}$ be a family of distributions over $\{0,1\}^{\lambda-1}$. A $(\mathcal{F}, \mathcal{X})$-nonmalleable* single bit digital locker lock *is a probabilistic polynomial-time algorithm that inputs a point $\mathsf{val} \in \{0,1\}^{\lambda-1}$ and bit $b \in \{0,1\}$ and outputs a circuit unlock. Let $\mathcal{V}$ be a verifier for lock as defined in Definition 3.1. The following conditions must be met:*

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

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

   where the probability is over the randomness of lock.

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

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

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

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

$$\Pr_{\mathsf{val} \leftarrow X}\left[\mathcal{V}(C) = 1, f \in \mathcal{F}, (I_{f(\mathsf{val}),0} \equiv C \vee I_{f(\mathsf{val}),1} \equiv C) | (C, f) \leftarrow \mathcal{A}(\mathsf{lock}(\mathsf{val}))\right] \leq \epsilon.$$

As stated in the introduction, we present a new generic construction of a nonmalleable single bit digital locker from a nonmalleable point function (assuming a rich enough tampering class).

Theorem 3.1. *Let* $\mathsf{lockPoint}$ *be $(\mathcal{F}_{point}, \mathcal{X})$-nonmalleable point function obfuscator where $\mathcal{X}$ is the set of all X such that $\mathrm{H}_\infty(X) \geq \omega(\log \lambda)$. Then for $\mathsf{lock}(x, b) = \mathsf{lockPoint}(x||b)$ is a $(\mathcal{F}_{point}, \mathcal{X})$-nonmalleable single bit digital locker if $\mathcal{F}_{point}$ is closed under multiplication by $2, 2^{-1}$ and addition and subtraction of 1.*

Proof. We separately argue completeness, soundness, and nonmalleability. We substitute val for x for legibility.

Completeness First note that $2x + b$ is injective in both variables. Completeness immediately follows.

Soundness Note no distribution is assumed on the bit b. Fix some $b \in \{0,1\}$. Let $\mathcal{Z}_\lambda$ be the set of distributions Z over $\{0,1\}^\lambda$ where $\mathrm{H}_\infty(Z) = \omega(\log \lambda)$ and similarly define the set of distributions $\mathcal{X}_{\lambda-1}$ as the set of distributions X over $\{0,1\}^{\lambda-1}$ where $\mathrm{H}_\infty(X) = \omega(\log \lambda)$. Lastly, define the set of distributions $\mathcal{Y} = \{2X + b | X \in \mathcal{X}_{\lambda-1}\}$ where we understand $2X + b$ to be a distribution created by sampling $x \leftarrow X$ and computing $2x + b$. Then $\mathcal{Y} \subseteq \mathcal{Z}_\lambda$. That is, for each distribution that lock is intended to secure it is contained in the set of distributions that $\mathsf{lockPoint}'$ is intended to secure.

We show soundness by contradiction. First assume that the construction described $\mathsf{lockPoint}(x)$ is sound. That is, for all $c \in \mathbb{Z}^+$ there exists some $\lambda_{p,c}$ such that for $\lambda > \lambda_{p,c}$ for all PPT $\mathcal{A}_p$, there exists a simulator $\mathcal{S}_p$ such that:

$$\left|\Pr_{x \in \{0,1\}^\lambda}[\mathcal{A}_p(\mathsf{lockPoint}(x)) = 1] - \Pr_{x \in \{0,1\}^\lambda}[\mathcal{S}_p^{I_x}(1^\lambda) = 1]\right| \leq \frac{1}{\lambda^c}.$$

In addition, suppose $\mathsf{lock}(x, b) = \mathsf{lockPoint}(2x + b)$ is insecure, that is, there exists some $c \in \mathbb{Z}^+$ such that for all $\lambda_{c,dl}$ there exists a $\lambda > \lambda_{c,dl}$ such that there exists a PPT $\mathcal{A}_{dl}$ where for all S_{dl} using a polynomial number of queries

$$\left|\Pr_{x \in \{0,1\}^{\lambda^*}}[\mathcal{A}_{dl}(\mathsf{lock}(x, b)) = 1] - \Pr_{x \in \{0,1\}^\lambda}[\mathcal{S}_{dl}^{I_{x,b}}(1^{\lambda^*}) = 1]\right| > \frac{1}{\lambda^c}.$$

For a fixed $c \in \mathbb{Z}^+$, denote by $\lambda_{\max,c} = \max\{\lambda_{p,c}, \lambda_{dl,c}\}$. There must exist some $\lambda > \lambda_{c,max}$ where the distance between $\mathcal{A}_p$ and S_p is less than $1/\lambda^c$ while there exists some $\mathcal{A}_{dl}$ such that for all $\mathcal{S}_{dl}$ the

input : Oracle access to $I_{x,b}$
output: $\mathcal{P}(x, b)$

Initialize $\mathcal{S}_p$.
Receive query y from $\mathcal{S}_p$, send $x_{0,\ldots\lambda-2}$ to $I_{x,b}$.
If response b, check if $x_{\lambda-1} = b$, if so return 1.
Otherwise **return** $\perp$ to S'.
Output S_{dl}'s output.

Algorithm 1: Construction of $\mathcal{S}_{dl}$ from $\mathcal{S}_p$

distance between $\mathcal{A}_{dl}$ and $\mathcal{S}_{dl}$ is greater than $1/\lambda^c$. Denote by $\mathcal{A}^*_{dl}$ one such adversary, that is for $\mathcal{A}^*_{dl}$ and all S_{dl} making a polynomial number of queries their statistical distance is at least $1/\lambda^c$. With the goal of arriving at a contradiction we use $\mathcal{A}^*_{dl}$ to construct an adversary $\mathcal{A}^*_p$ for $\mathsf{lockPoint}$ that cannot be effectively simulated.

This adversary $\mathcal{A}^*_p$ works as follows, on input $\mathsf{lockPoint}(x)$ where $x \in \{0,1\}^\lambda$, $\mathcal{A}^*_p$ initializes $\mathcal{A}^*_{dl}$ with input $\mathsf{lockPoint}(x)$. Note this is equivalent to initializing $\mathcal{A}^*_{dl}$ with input $\mathsf{lock}(x', b) = \mathsf{lockPoint}(x)$ for $x = x'||b$. Any predicate $\mathcal{P}(x'||b) = \mathcal{P}(x)$ output by $\mathcal{A}^*_{dl}$ is a valid predicate on on x. So, this predicate can be immediately output. Thus, $\mathcal{A}^*_{dl}$ implies the existence of an $\mathcal{A}^*_p$ that outputs 1 with the same probability. That is,

$$\Pr[\mathcal{A}^*_p[(\mathsf{lockPoint}(x)) = 1] = \Pr[\mathcal{A}^*_{dl}(\mathsf{lock}(x', b)) = 1].$$

By assumption for each $\mathcal{A}_p$ there exists $\mathcal{S}_p$ such that for $\lambda > \lambda_{\max,c}$,

$$\left| \Pr_{x\in\mathcal{Z}^\lambda}[\mathcal{A}_p(\mathsf{lockPoint}(x)) = \mathcal{P}(x)] - \Pr_{x\in\mathcal{Z}^\lambda}[\mathcal{S}_p^{I_x(\cdot)}(1^\lambda) = \mathcal{P}(x)] \right| \leq \frac{1}{\lambda^c}.$$

This $\mathcal{S}_p$ implies the existence of an $\mathcal{S}_{dl}$ that computes any predicate $\mathcal{P}$ on (x', b). $\mathcal{S}_{dl}$ initializes S_p and for every query from S_p it drops the last bit and asks its oracle the query. It only returns a match to S_p if the returned bit matches the last bit of S_p's query. A formal description is in Algorithm 1. So, the existence of $\mathcal{S}_p$ directly leads to the simulator $\mathcal{S}_{dl}$ that computes a 1 with the same probability. That is,

$$\Pr[\mathcal{S}_p^{I_x}(1^\lambda) = 1] = \Pr[\mathcal{S}_{dl}^{I_{x,b}}(1^\lambda) = 1].$$

These three equations allow us to state:

$$\left| \Pr_{x\leftarrow X}[\mathcal{A}^*_{dl}(\mathsf{lock}(x, b)) = 1] - \Pr_{x\leftarrow X}[\mathcal{S}_{dl}^{I_{x,b}}(1^{\lambda^*}) = 1] \right| \leq \frac{1}{\lambda^c}.$$

This is a contradiction and completes the argument of soundness.

Nonmalleability We proceed by contradiction assuming that there exists some ensemble X'_λ where $\mathrm{H}_\infty(X_\lambda) = \omega(\log\lambda)$, some $\mathcal{A}_{dl}$, and some bit $b \in \{0,1\}$ such that there exists some c where

$$\Pr_{x'\leftarrow X'_\lambda}\left[\begin{array}{c}(\mathsf{lock}', f) \leftarrow \mathcal{A}_{dl}(\mathsf{lock}(x', b)) \\ \mathcal{V}(\mathsf{lock}') = 1, f \in \mathcal{F}, (I_{f(x'),0} \equiv \mathsf{lock}' \vee I_{f(x'),1} \equiv \mathsf{lock}')|\end{array}\right] > 1/\lambda^c$$

We build an mauling adversary for the original obfuscation $\mathsf{lockPoint}$. We consider the ensemble of distributions $Z_\lambda = X_\lambda||b$, that is the distribution of X_λ with the bit b appended. Note that this is a valid ensemble for the obfuscator $\mathsf{lockPoint}$. We design $\mathcal{A}_p$ as follows:

1. Receive $\mathsf{lockPoint}(x)$ as input (equivalently receive $\mathsf{lock}(x', b)$ where $x = 2x' + b$).
2. Initialize $\mathcal{A}_{dl}$ with $\mathsf{lock}(x', b)$.
3. Receive lock', f.
4. Set $f_0(x) = 2^{-1}(x - b)$ and $f_2(x) = 2 * (x)$. Compute $f_3 = f_2 \circ f \circ f_0$.
5. Flip $r \xleftarrow{\$} \{0,1\}$. Set $f' = f_3 + r$.
6. Output lock', f'.

$\mathcal{A}_p$ is trying to produce a tampering function on $x \in \{0,1\}^\lambda$, while $\mathcal{A}_{dl}$ is tampering on x' contained in $x'||b$. The manipulations f_0 and f_2 are to create a valid tampering function on x ignoring the last bit. Secondly, $\mathcal{A}_p$ output a new $x^* = f(x)||b^*$ where $b^* \in \{0,1\}$. Since $\mathcal{A}_p$ does not know if the last bit of x^* it outputs each possibility with probability $1/2$. Together these facts allow us to conclude:

$$\begin{aligned}
&\Pr_{x \leftarrow Z_\lambda}\left[\begin{array}{c}(\mathsf{lockPoint}', f) \leftarrow \mathcal{A}_p(\mathsf{lockPoint}(x)) \\ \mathcal{V}(\mathsf{lockPoint}') = 1, f \in \mathcal{F}, (I_{f(x)} \equiv \mathsf{lockPoint}')\end{array}\right] \\
&\quad = \Pr[R = r] \Pr_{x' \leftarrow X_\lambda}\left[\begin{array}{c}(\mathsf{lockPoint}', f) \leftarrow \mathcal{A}_p(\mathsf{lockPoint}(x')) \\ \mathcal{V}(\mathsf{lockPoint}') = 1, f \in \mathcal{F}, (I_{f(x'),r} \equiv \mathsf{lockPoint})\end{array}\right] \\
&\quad = \frac{1}{2} \Pr_{x' \leftarrow X_\lambda}\left[\begin{array}{c}(\mathsf{lock}', f) \leftarrow \mathcal{A}_{dl}(\mathsf{lock}(x', b) \\ \mathcal{V}(\mathsf{lock}') = 1, f \in \mathcal{F}, (I_{f(x'),0} \equiv \mathsf{lock}' \vee I_{f(x'),1} \equiv \mathsf{lock}')\end{array}\right] \\
&\quad > \frac{1}{2\lambda^c}.
\end{aligned}$$

This contradicts the nonmalleability of $\mathsf{lockPoint}$ and completes the proof of nonmalleability. This completes the proof of Theorem 3.1.

□

3.1 Stocky Lockers and full nonmalleable digital lockers

We now present our two main building blocks, an augmented digital locker for short keys that we call a stocky locker and a full digital locker. We introduce the digital locker first as the (nonmalleable) definition has appeared before in Canetti and Dakdouk [CD08].

Definition 3.4 (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 $\mathcal{V}$ be a verifier for $\mathsf{lockPoint}$. The following conditions must be met:*

1. ***Completeness**: 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 (possibly inefficient) 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 $\mathsf{key} \in \{0,1\}^n$, and for any $\mathcal{P} : \{0,1\}^{\lambda+n} \mapsto \{0,1\}$,*

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

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:*

$$\Pr_{\mathsf{val} \leftarrow X}\left[\mathcal{V}(C) = 1, f \in \mathcal{F}, g \in \mathcal{G} \wedge (I_{f(\mathsf{val}),g(\mathsf{key})} \equiv C | (C,f,g) \leftarrow \mathcal{A}(\mathsf{lock}(\mathsf{val}))\right] \leq \epsilon.$$

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

Stocky Lockers A stocky locker is a digital locker with a short key, that is composable with the same val, provides nonmalleability over val, and produces obfuscations that are indistinguishable from obfuscations of random points.

Definition 3.5 (Stocky Locker)**.** *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 $(\mathcal{F}, \mathcal{X}, \rho, \tau)$-nonmalleable* stocky locker stockLock *is a probabilistic polynomial-time algorithm that inputs a point $\mathsf{val} \in \{0,1\}^\lambda$ and string $\mathsf{key} \in \{0,1\}^\tau$ and outputs a circuit* stockUnlock*. Let $\mathcal{V}$ be a verifier for* lock*. The following conditions must be met:*

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

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

where the probability is over the randomness of stockLock*.*

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

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

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

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

$$\Pr_{\mathsf{val} \leftarrow X}\left[\mathcal{V}(C) = 1, f \in \mathcal{F}, \exists y \in \{0,1\}^\tau, I_{f(\mathsf{val}),y} \equiv C | (C,f) \leftarrow \mathcal{A}(\{\mathsf{stockLock}(\mathsf{val},\mathsf{key}_i)\}_{i=1}^\rho)\right] \leq \epsilon.$$

4. ***Pseudorandomness*** *Let U_λ be the uniform distribution. For any distribution $X \in \mathcal{X}$, for all $x \in X$, $\{\mathsf{key}_i \in \{0,1\}^\tau\}_{i=1}^\rho$ and any PPT D, there exists $\epsilon = \mathtt{ngl}(\lambda)$ such that:*

$$\Pr_{x \leftarrow X}[D(\mathsf{lock}(x,\mathsf{key}_i) = 1] - \Pr_{y_i \xleftarrow{\$} U_\lambda^\rho}[D(\mathsf{lock}(y_i,\mathsf{key}_i) = 1] \leq \epsilon.$$

4 Building Stocky Lockers from Komargodski and Yogev [KY18]

In this section, we show that Komargodski and Yogev's point function obfuscator suffices to build a stocky locker. Our assumptions are slightly modified to use average minentropy. That is, we assume the adversary may have some side information Y that is correlated with val. This modification is made because the distribution of val that depends on a seed of a seed-dependent condenser still has average minentropy conditioned on the output (see Section 6).

Construction 4.1. *Let $\lambda \in \mathbb{N}$ be a security parameter and let $\{0,1\}^{\lambda-1}$ be the domain. Let $\mathcal{F}_{t,\mathtt{poly}} \stackrel{def}{=} \{f : \{0,1\}^{\lambda-1} \to \{0,1\}^{\lambda-1}\}_{\lambda\in\mathbb{N}}$ the ensemble of all functions that can be computed by polynomials of degree at most t except constant and identity functions. 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 a group $\mathbb{G}_{5\lambda}$. For known generator $g \in \mathbb{G}_{5\lambda}$, Our single bit digital locker gets an element $\mathsf{val} \in \{0,1\}^{\lambda-1}, b \in \{0,1\}$ and randomness $r \in \mathbb{G}_{5\lambda}$. It then outputs:*

$$\mathsf{lock}(\mathsf{val}, b; r) \stackrel{def}{=} \left(r, r^{g^{(2\mathsf{val}+b)^4+(2\mathsf{val}+b)^3+(2\mathsf{val}+b)^2+(2\mathsf{val}+b)}}\right).$$

Given a program unlock consisting of two group elements r', y' for test password val', the user computes:

$$\mathsf{unlock}(\mathsf{val}', 0) = \left((r')^{g^{(2\mathsf{val}')^4+(2\mathsf{val}')^3+(2\mathsf{val}')^2+(2\mathsf{val}')}} \stackrel{?}{=} y'.\right)$$
$$\mathsf{unlock}(\mathsf{val}', 1) = \left((r')^{g^{(2\mathsf{val}'+1)^4+(2\mathsf{val}'+1)^3+(2\mathsf{val}'+1)^2+(2\mathsf{val}'+1)}} \stackrel{?}{=} y'.\right)$$

If $\mathsf{unlock}(\mathsf{val}', b)$ outputs 1 (for either $b = 0$ or $b = 1$), then the user outputs b. Otherwise, $\perp$ is the output.

Throughout our discussion we use $h(z) = z^4 + z^3 + z^2 + z$ as shorthand for the polynomial being computed in the exponent. We now introduce the relevant hardness assumptions. We consider an ensemble of groups with efficient operations where $\mathbb{G}_\lambda$ is a group of prime order $p \in (2^\lambda, 2^{\lambda+1})$.

The first assumption we will use is the t-strong vector DDH introduced by Bitanski and Canetti [BC10]. This strengthens the DDH assumption in two ways. First, the exponents are not drawn from uniform powers but rather from a well spread distribution. Second, the adversary is given multiple samples $x_1, \ldots, x_\ell$, each from correlated distributions $\mathcal{X}_1, \ldots, \mathcal{X}_t$. The only guarantee is on the marginal distribution of each $\mathcal{X}_i$; nothing is guaranteed about the joint distribution. As an example, each distribution could be identically distributed. Here we introduce a new variant of this assumption where each distribution has average minentropy. Note this change is largely syntactic, as the probability of the worst case minentropy being a $\omega(\log\lambda)$ factor lower than the average minentropy is negligible (by [DORS08, Lemma 2.2a]).

Assumption 4.1 (t-Strong Average Vector DDH)**.** *Let $t = \mathtt{poly}(\lambda)$ be a parameter and let $\mathbb{G}_\lambda$ be an ensemble of groups with efficient representation and operations. We say that the* t-strong average vector decision Diffie-Hellman assumption *holds if for any vector X, Y (we elide the subscript λ) where X is a vector in $(\mathbb{Z}_p^*)^t$ and each X_i is average-case well-spread (conditioned on Y), it holds that for every $s_{sec} = poly(\lambda)$ there exists some $\epsilon = \mathtt{ngl}(\lambda)$ such that:*

$$\delta^{s_{sec}}\left((g_1, g_1^{x_1}, y_1, ..., g_t, g_t^{x_t}, y_t), (g_1, g_1^{u_1}, y_1, ..., g_t, g_t^{u_t}, y_t)\right) \leq \epsilon.$$

Where $((x_1, y_1), ..., (x_t, y_t)) \leftarrow (X, Y)$ and u_i is sampled uniformly from $\mathbb{Z}_p^*$.

The second assumption we will consider is a variant of the power DDH assumption. The t-strong entropic power DDH assumption (introduced by Komargodski and Yogev [KY18]) says that increasing powers of a single element x are indistinguishable from a coordinate wise well-spread distribution.

Assumption 4.2 (t-Strong Entropic Power DDH)**.** *The* t-strong average DDH assumption *is said to hold for an ensemble of groups* $\mathbb{G}_\lambda$ *with associated generator* g *if for any average-case well-spread distribution ensemble* X, Y *(we elide the subscript* λ*), there exists a family of independent distributions* $Z_{1,y}, ..., Z_{t,y}$ *giving rise to independent distributions* (Z_i, Y) *that are average-case well-spread (Definition 2.1), such that for any* $s_{sec} = \mathtt{poly}(\lambda)$ *there exists* $\epsilon = \mathtt{ngl}(\lambda)$ *such that*

$$\delta^{s_{sec}}((g, g^X, g^{X^2}, ..., g^{X^t}, Y), (g, g^{Z_1}, g^{Z_2}, ..., g^{Z_t}, Y)) \leq \epsilon.$$

where the distribution (X, Y) *is jointly sampled and* Z_i *are sampled conditioned on* $y \leftarrow Y$*.*

We now show that Construction 4.1 is a stocky locker. There are two things that could go wrong when an adversary receives single bit digital lockers on correlated points: 1) seeing obfuscations with $b = 0$ and $b = 1$ may allow the adversary to maul, and 2) having multiple samples of points under different randomness may break privacy. To detect tampering, we show security against an adversary that obtains $g^{h(2\mathsf{val})}$ and $g^{h(2\mathsf{val}+1)}$. That is, we don't rely on the generators r_i in providing any protection against tampering. This argument has two parts (relying on the entropic power assumption): First, predicting polynomials of degree greater than 1 predicts an unseen power and can thus distinguish between powers and random group elements. Then, predicting linear polynomials corresponds to finding a value that is outside the linear span of provided values (which is information theoretically hard in the random case). To show that soundness and pseudorandomness are preserved, we rely on the t-strong vector DDH assumption.

Theorem 4.1. *Suppose that*

1. *The* n*-strong vector DDH assumption holds,*
2. *The* $4t$*-entropic power DDH assumption holds,*
3. *The selected prime* $p \notin \{2, 3, 5, 11, 17\}$*. (As* $\mathbb{G}_\lambda$ *increases these primes will never be selected.)*
4. X *is a distribution over* $\{0,1\}^{\lambda-1}$ *such that* $\mathrm{H}_\infty(X) = \omega(\log \lambda)$*.*

Then Construction 4.1 is a $(\mathcal{F}, X, n, 1)$*-nonmalleable stocky locker, where* $\mathcal{F} = \{f | f$ *is a polynomial,* $\deg(f) \leq t\}$ *(excluding constant polynomials and the identity polynomial).*

Proof. We separately consider completeness, soundness, pseudorandomness and nonmalleability under composition. We substitute x for val for legibility.

Completeness This is a direct extension of completeness for underlying point function and the generic transform that is shown in Theorem 3.1. (The crucial fact is that $g^{h(2x+b)}$ is a one-to-one function.)

Soundness and Pseudorandomness Define the random variables $\vec{X} \stackrel{def}{=} \{X_i = (X||\mathsf{key}_i)\}_{i=1}^n$. Since X is distribution where $\mathrm{H}_\infty(X) = \omega(\log \lambda)$, $\vec{X}$ is a average-case well spread distribution (according to Definition 2.1). Since the function

$$f(x, b) = g^{(2x+b)^4+(2x+b)^3+(2x+b)^2+(2x+b)}$$

is one-to-one it is also true that $g^{h(\vec{X})}$ is average-case well spread. A one-to-one function can be applied before obfuscation without affecting privacy [KY18, Claim 3.1]. This proof directly carries over to the single bit digital locker setting. The strong vector DDH assumption then says that $\{r_i, r_i^{X_i}\}_{i=1}^n \approx \{r_i, u_i\}_{i=1}^n$ for uniform group elements u_i. This means the construction satisfies a weaker notion called distributional indistinguishability [BC10, Definition 5.3], which says no adversary can tell between obfuscations of related points and independent uniform points. Bitanski and Canetti [BC10] show that this definition implies composition for virtual-grey box obfuscation. (Their proof is for point obfuscators but can be modified for this setting.) Overall virtual black box security then follows using arguments from [CD08]. This argument also suffices to show that obfuscations are pseudorandom.

Nonmalleability We first recall that we use $h(x)$ to denote $x^4 + x^3 + x^2 + x$. In order to prove nonmalleability is preserved, we will show how, given an adversary that can maul our obfuscation given two distinct obfuscations with the same point x, we can create an algorithm that can break the Strong Power DDH assumption. We assume that key is a value known to both the reduction and the adversary. (That is, we do not rely on any uncertainty with respect to key.) That is, we assume that there exists some key and a PPT $\mathcal{A}$ such that for all negligible functions ϵ:

$$\Pr_{x \leftarrow X}\left[\begin{array}{c}(\mathsf{unlock}', f) \leftarrow \mathcal{A}(\mathsf{lock}(\mathsf{val}, \mathsf{key})) \\ \mathcal{V}(\mathsf{unlock}') = 1, f \in \mathcal{F}, \exists \mathsf{key}' \{0,1\}^n \wedge (I_{(f(\mathsf{val}), \mathsf{key}')} \equiv C)\end{array}\right] > \epsilon.$$

We show how to construct $\mathcal{A}'$ that breaks the $\tau = 4t$ entropic power DDH assumption (Assumption 4.2). Suppose we receive a sequence $\{g, g^{z_1}, g^{z_2}, ..., g^{z_\tau}\}$ where each z_i either equals x^i (sampled $x \leftarrow X$ where $X \in \{0,1\}^{\lambda - 1}$) or a random group element r_i. We first compute two values:

$$y_0 = g^{16z_4 + 8z_3 + 4z_2 + 2z_1} = g^{16 * z_4} \cdot g^{8 * z_3} \cdot g^{4z_2} \cdot g^{2z_1},$$
$$y_1 = g^{16z_4 + 40z_3 + 40z_2 + 20z_1 + 4}.$$

$\mathcal{A}'$ then computes a vector based on these values:

$$\{r_i \xleftarrow{\$} \mathbb{G}, r_i^{y_{\mathsf{key}_i}}\}_{i=1}^n.$$

Then $\mathcal{A}$ is initialized based on these values and outputs a function f and a $2n$ vector of group elements $\{r_{\mathcal{A},i}, w_{\mathcal{A},i}\}_{i=1}^n$. We assume f is specified by coefficients (if not, these coefficients can be interpolated using points from the distribution X, see [KY18]). We can then use the f provided by $\mathcal{A}$ to check if each point in the vector is a valid single bit digital locker of $f(x)$ and a bit 0 or 1. Details for this check are in Algorithm 2. We proceed to analyze the success of this algorithm in both the real case $z_i = x^i$ and the random case $z_i = r_i$.

The real case In the real case the adversary $\mathcal{A}$ sees pairs $\left(r_i, r_i^{g^{h(2x + \mathsf{key}_i)}}\right)$. This is exactly the distribution expected by $\mathcal{A}$. Furthermore, $\mathcal{A}'$ outputs 1 when the mauled obfuscation is a valid obfuscation of x on some key'. Thus, given the real distribution $\mathcal{A}'$ outputs 1 with probability at least ϵ.

The random case We now assume that each z_i is a uniform and randomly distributed s_i for $1 \leq i \leq \tau$. We assume that the adversary is computationally unbounded and is provided with two points $c_0 = 16s_4 + 8s_3 + 4s_2 + 2s_1$ and $c_1 = 16s_4 + 40s_3 + 40s_2 + 20s_1 + 4$. (We can provide the adversary also with the values r_i.) That is, we give the adversary direct access to the value in the exponent. Its clear if no adversary can win in this game, then no adversary can win in the original game.

input : $g, g^{z_1}, ..., g^{z_\tau}$
output: $\mathcal{P}(x)$

1. Sample $\mathsf{key} \leftarrow \{0,1\}^n$.
2. Compute $y_0 = g^{16z_4+8z_3+4z_2+2z_1}$ and $y_1 = g^{16z_4+40z_3+40z_2+20z_1+4}$.
3. Compute $\{r_i \xleftarrow{\$} \mathbb{G}, r_i^{y_{\mathsf{key}_i}}\}_{i=1}^n$..
4. Run $(f, \{r_{\mathcal{A},i}, w_{\mathcal{A},i}\}_{i=1}^n) \leftarrow \mathcal{A}(\{r_i, r_i^{y_{\mathsf{key}_i}}\}_{i=1}^n)$. If the output is not a function followed by $2n$ group elements output 0.
5. Compute coefficients α_i of $h(2f(x))$ and β_i of $h(2 * f(x) + 1)$.
6. For $i = 1$ to n
   (a) Check if $w_{\mathcal{A},i} \stackrel{?}{=} r_{\mathcal{A},i}^{(\sum_{i=0}^{\tau} \alpha_i z_i)}$ or $w_{\mathcal{A},i} \stackrel{?}{=} r_{\mathcal{A},i}^{(\sum_{i=0}^{\tau} \beta_i z_i)}$.
   (b) If neither check is true, output 0.
7. Output 1.

Algorithm 2: Construction of $\mathcal{A}'$ from $\mathcal{A}$

In order for $\mathcal{A}$ to succeed she needs to compute $c_\alpha = \sum_{i=0}^{\tau} \alpha_i s_i$ or $c_\beta = \sum_{i=0}^{\tau} \beta_i s_i$ (the vectors $\vec{\alpha}$ and $\vec{\beta}$ are defined in Algorithm 2). If the degree of the polynomial f is greater than 1 this requires computing a linear combination with some s_i where $i > 4$ that is independent of the adversary's view. In this case, both the distribution of both random variables C_α and C_β has entropy $\log |G_\lambda| = \lambda$ conditioned on the adversary's view [KY18, Claim 4.4]. By a union bound the probability of matching either C_α or C_β for some i is at most $\Pr[\text{success}] \leq \frac{2}{2^\lambda} = \frac{1}{2^{\lambda-1}}$.

We now move to the case where the polynomial f is of degree 1. That is, $f(x) = \mu x + \nu$. If the function f is a linear one, then we will show how an accurate obfuscation of $h(2f(x') + b)$ cannot be formed from the inputs. To do this, we will look at what information about the points that the adversary receives. The adversary receives a multiple linear combinations of s_1, s_2, s_3, s_4.

$$\begin{bmatrix} 16 & 8 & 4 & 2 & 0 \\ 16 & 40 & 40 & 20 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_4 \\ s_3 \\ s_2 \\ s_1 \\ 1 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ 1 \end{bmatrix}.$$

The first row of the above matrix corresponds to the linear combination used when $\mathsf{key}_i = 1$, the second row when $\mathsf{key}_i = 0$ and the last row is the constant group element. As stated, the function f can be rewritten as $f(x) = \mu * x' + \nu$. By substituting and simplifying, f can finally be rewritten as:

$$2f(x) + b' = 2 * (\mu x + \nu) + b' = 2\mu x + 2\nu + b' = 2ax + b$$

for some $b \in \{0,1\}$ and a is a field element. Note we consider this as an existential argument so $a = (\mu x + \nu)x^{-1} = \mu + \nu x^{-1}$ is a valid assignment for a. We can write the desired linear combination as follows:

$$\begin{bmatrix} 16a^4 \\ 32a^3b + 8a^3 \\ 20a^2b^2 + 16a^2b + 4a^2 \\ 6ab^3 + 6ab^2 + 6ab + 2a \\ b^4 + b^3 + b^2 + b \end{bmatrix}^\intercal \begin{bmatrix} s_4 \\ s_3 \\ s_2 \\ s_1 \\ 1 \end{bmatrix}.$$

We now show that even for an unbounded $\mathcal{A}$, this value is information theoretically hidden (given $c_0, c_1, 1$).

Lemma 4.1. *Let S_1, S_2, S_3, S_4 be uniformly distributed in $\mathbb{G}_{5\lambda}$ then define $C_0 = 16S_1 + 8S_2 + 4S_3 + 2S_4 \mod G_{5\lambda}$ and $C_1 = 16S_1 + 40S_2 + 40S_3 + 20S_4 + 4 \mod G_{5\lambda}$ Define for $a \in \mathbb{G}_{5\lambda}, b \in \{0,1\}$,*

$$\begin{aligned} C^*_{a,b} = 16a^4 S_4 + (32a^3b + 8a^3)S_3 + (20a^2b^2 + 16a^2b + 4a^2)S_2 \\ + (6ab^3 + 6ab^2 + 6ab + 2a)S_1 + (b^4 + b^3 + b^2 + b). \end{aligned}$$

Then the value $\mathrm{H}_\infty(C^*_{a,b} | C_0, C_1) \geq \lambda - 1$ if $a \neq 0, 1$ and $p \notin \{2, 3, 5, 11, 17\}$.

Proof of Lemma 4.1. We first show that when $a \neq 0, 1$, the value $c_{a,b}$ is linearly independent of the values $c_0, c_1, 1$. That is, we show the following system has no solutions $\alpha_0 c_0 + \alpha_1 c_1 + \alpha_2 c_2 = c_{a,b}$. Since $\mathcal{A}$ only has access to linear combinations of these variables in order for $\mathcal{A}$ to properly output some correct mauled obfuscation, they must find a solution to the following:

$$\begin{bmatrix} 16 & 8 & 4 & 2 & 0 \\ 16 & 40 & 40 & 20 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^\intercal \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 16a^4 \\ 32a^3b + 8a^3 \\ 20a^2b^2 + 16a^2b + 4a^2 \\ 6ab^3 + 6ab^2 + 6ab + 2a \\ b^4 + b^3 + b^2 + b \end{bmatrix}$$

where $\alpha_0, \alpha_1, \alpha_2$ are field elements. Because b is a single bit (i.e. either 0 or 1), we will examine the existence of solutions under these two possibilities. In both cases we assume that the adversary can exactly solve the last equation using α_2 as a free variable and consider the following reduced system:

$$\begin{bmatrix} 16 & 16 \\ 40 & 8 \\ 40 & 4 \\ 20 & 2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} 16a^4 \\ 32a^3b + 8a^3 \\ 20a^2b^2 + 16a^2b + 4a^2 \\ 6ab^3 + 6ab^2 + 6ab + 2a \end{bmatrix}$$

Case 1: $b = 0$ Considering the two by two system formed by the second and third equation yields that:

$$\begin{aligned} 40\alpha_0 &= 8a^2 - 8a^3, \\ 4\alpha_1 &= 8a^3 - 4a^2. \end{aligned}$$

Substituting these values into the fourth equation yields a quadratic for a:

$$4a^2 - 2a = 0$$

This has solutions of $a = 0, 2^{-1}$. Substituting the solutions for α_0, α_1 into the constraint from the first equation yields the quartic:

$$16a^4 - (32 + 16 * 5^{-1})a^3 + (16 - 16 * 5^{-1})a^2 = 0$$

This equation is consistent with the solutions where $a = 0$. However, when $a = 2^{-1}$, the first equation is only satisfied when $3 \equiv 0 \mod p$. Thus, it suffices for $p \neq 3$. Since the solution when $a = 0$ is considered trivial, nontrivial solutions exist in this case only when $p = 3$.

Case 2: $b = 1$ Again starting with the second and third linear constraints we have that:

$$\alpha_0 = 2a^2 - a^3,$$
$$\alpha_1 = 10a^3 - 10a^2.$$

Substituting these values into the fourth equation yields $180a^3 - 160a^2 - 20a = 0$. Which has the trivial solutions of $a = 0, 1$ and nontrivial solution $a = -1 * 9^{-1}$. However, when $a = -1 * 9^{-1}$, the first equation is only satisfied when $11968 \equiv 0 \mod p$. Thus, it suffices for $p \notin \{2, 11, 17\}$. Since the solution when $a = 0$ is considered trivial, nontrivial solutions exist only when $p \in \{2, 11, 17\}$.

Putting things together So, regardless of choice of b, if a is nontrivial, the value $c_{a,b}$ is linearly independent of c_0, c_1 and 1. Consider the following system of equations:

$$\begin{bmatrix} 16 & 16 & 0 & 16a^4 \\ 8 & 40 & 0 & 32a^3b + 8a^3 \\ 1 & 40 & 0 & 20a^2b^2 + 16a^2b + 4a^2 \\ 2 & 20 & 0 & 6ab^3 + 6ab^2 + 6ab + 2a \\ 0 & 4 & 1 & b^4 + b^3 + b^2 + b \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} s_4 \\ s_3 \\ s_2 \\ s_1 \\ 1 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ 1 \\ c_{a,b} \end{bmatrix}$$

This system of equations is rank 4 (in the case when a is nontrivial). This means for any particular c_0, c_1 there are at least 2^λ tuples of S_1, S_2, S_3, S_4 that produce any particular value of $c_{a,b}$. In particular, the probability that $\Pr[C_{a,b} = c | C_0, C_1] = \frac{1}{2^\lambda}$. Since the adversary has to match one of two values by union bound they succeed with probability at most $2^{-\lambda+1}$. This completes the proof of Lemma 4.1. □

Thus, in both random cases the probability of mauling is at most $2^{-(\lambda-1)}$. This allows us to state the distinguishing capability of $\mathcal{A}$:

$$\Pr[\mathcal{A}(\{g^{x^i}\}_{i=1}^{\tau}) = 1] - \Pr[\mathcal{A}(\{g^{r_i}\}_{i=1}^{\tau}) = 1] \geq \epsilon - \frac{1}{2^{\lambda-1}}.$$

This yields that $\mathcal{A}$ is a distinguisher between entropic powers and random exponents. This contradicts Assumption 4.2 and so completes Theorem 4.1. □

5 A stocky locker with multiple key bits

We also construct a stocky locker that outputs a logarithmic number of bits using the same assumptions as the nonmalleable point function of Komargodski and Yogev [KY18]. Encoding a logarithmic number of bits at a time has important implications for ensuring nonmalleability of key (described next). Many nonmalleable primitives consider non-binary symbols.

Suppose that we wish to encode τ bit symbols in each group element. The most natural idea is to extend our first construction by computing $h(2^\tau * \mathsf{val} + \mathsf{key}_i)$. As stated above, in the single bit case, nonmalleability requires showing that given $h(2\mathsf{val})$ and $h(2\mathsf{val}+1)$ it was difficult to compute $h(2\mathsf{val}'+b')$. Under the entropic power DDH assumption (Assumption 4.2) this requires one to show that $h(2\mathsf{val}'+b)$ is linearly independent of $h(2\mathsf{val})$ and $h(2\mathsf{val}+1)$. Roughly, this corresponds to an adversary being given $r_1+r_2+r_3+r_4$ and being asked to predict a sum with different weights. Since there are only 4 r_i with four (linearly independent) constraints, the adversary will be able to predict any value. To address this problem, we introduce a new hash function for symbols $y \in \{0,1\}^\tau$:

$$h(x,y) = \left(\sum_{i=\tau}^{4+\tau}(x+2)^i\right) + \left(\sum_{i=0}^{\tau-1} y_i \cdot (x+2)^i\right).$$

This construction ensures that all 2^τ different values of y span at most a τ dimensional subspace and cannot be used to predict the value of the hash function for any $x' \neq x$.

Construction 5.1. *Let all variables be as in Construction 4.1, let* $\mathsf{key} \in \{0,1\}^n$ *be some arbitrary value, and let* $\tau = O(\log\lambda)$. *Then define* $\mathsf{lock}(\mathsf{val},\mathsf{key})$ *as follows: for* $i = 1$ *to* n/τ *compute:*

1. *Sample* $r_i \leftarrow \mathbb{G}_{(6+\tau)\lambda}$.
2. *Set* $z = x+2$.
3. *Output* $(r_i, (r_i)^{g^{z^{4+\tau}+z^{3+\tau}+z^{2+\tau}+z^{1+\tau}+z^\tau+\sum_{i=0}^{\tau-1}\mathsf{key}_i z^i}})$.

Define $\mathsf{unlock}(\mathsf{val})$ *as follows: for* $i = 1$ *to* n/τ, *input* r_i, y_i *for each* $v_j \in [0, 2^\tau)$ *compute:*

$$P(x,v_j,i) = \left(r_i^{g^{(x+2)^{4+\tau}+(x+2)^{3+\tau}+(x+2)^{2+\tau}+(x+2)^{1+\tau}+(x+2)^\tau+\sum_{j=0}^{\tau-1} v_j(x+2)^j}} \stackrel{?}{=} y_i\right).$$

If $P(x,v_j,i)$ *outputs* 1 *then the user sets* $\mathsf{key}_i = v_j$. *Otherwise output* $\perp$.

There are a few notes about this new construction:

1. Opening checks each possible y value, running time is proportional to 2^τ making this construction efficient when $\tau = O(\log(\lambda))$.
2. Our single bit construction used a polynomial that included four nonzero powers. This construction has 5 powers that are not multiplied by part of y. If only 4 powers are not multiplied by a bit of y, there are three values of τ where nonmalleability is not guaranteed. This is because the choice of τ introduces a new degree of freedom to the linear system. The three values of τ are solutions to $\tau^3 - 15\tau^2 - 52\tau - 12 \equiv 0 \mod p$. It would be possible to avoid these τ by checking when τ is a solution for a particular p. Instead, our construction adds another power of x, which retains a single bad τ which is $\tau \equiv p-5 \mod p$. This value of τ never occurs for logarithmic τ.
3. To ensure the construction is correct and one-to-one, we restrict $x \in \{0,1\}^{\lambda-1}$ and compute powers of $x+2$. We can think of the construction as taking a sum of a subset of powers of x, we need to manually exclude $x \in \{0,1\}$ to ensure the function is one-to-one.

4. The group operations need to be in a group of size $(6+\tau)\lambda$ to ensure that operations do not overflow. In the first construction this size is only 5λ.

5. The construction allows a weaker vector DDH assumption at the cost of a stronger power DDH assumption. It thus allows a tradeoff between these two parameters.

6. In the theorem below we only consider nonmalleability over the locked val and assume no distribution over key.

Theorem 5.1. *Suppose that*

1. *The $\frac{n}{\tau}$-Strong average vector DDH assumption holds,*

2. *The $(5+\tau)t$-Strong entropic power DDH assumption holds,*

3. *The selected prime $p \notin \{2, 31, 41, 73\}$. (As $\mathbb{G}_\lambda$ increases these primes will never be selected. We include this condition as the proof does not apply if $p \in \{2, 31, 41, 73\}$.)*

4. *X is a distribution over $\{0,1\}^{\lambda-1}$ such that $\mathrm{H}_\infty(X) = \omega(\log \lambda)$.*

Then the $(\mathsf{lock}, \mathsf{unlock})$ in Construction 5.1 is n/τ-composable point obfuscation that is nonmalleable for $\mathcal{F} = \{f | \deg(f) \leq t\}$ (excluding constant polynomials and the identity polynomial).

Proof. As before we separately consider completeness, soundness, and nonmalleability.

Completeness For completeness first note that

$$(x+1)^{4+\tau} + (x+1)^{3+\tau} + (x+1)^{2+\tau} + (x+1)^{1+\tau} + (x+1)^{\tau} > \sum_{i=0}^{4+\tau} x^i$$

In particular, the binomial expansion of $(x+1)^{4+\tau}$ has nonzero coefficients on every power $i \leq 4+\tau$. This shows that for any $\mathsf{key}, \mathsf{key}' \in \{0,1\}^\tau$ and any $x > x'$ it is true that

$$g^{(x+2)^{4+\tau}+(x+2)^{3+\tau}+(x+2)^{2+\tau}+(x+2)^{1+\tau}+(x+2)^{\tau}+\sum_{i=0}^{\tau-1} \mathsf{key}_i (x+2)^i} \neq$$
$$g^{(x'+2)^{4+\tau}+(x'+2)^{3+\tau}+(x'+2)^{2+\tau}+(x'+2)^{1+\tau}+(x'+2)^{\tau}+\sum_{i=0}^{\tau-1} \mathsf{key}'_i (x'+2)^i}.$$

That is, the value of key cannot cause the obfuscation to unlock on a different point. Furthermore, the function is one-to-one as long as $x \notin \{0,1\}$. These cases are avoiding by computing $x+2$ before computing the polynomial. Note that since $x \in \{0,1\}^{\lambda-1}$ it always holds that $(x+2) \in \{0,1\}^\lambda$.

Soundness The privacy argument for this construction is exactly the same as in Theorem 4.1 since the function

$$f(x, \mathsf{key}_i) = g^{(x+2)^{4+\tau}+x^{3+\tau}+(x+2)^{2+\tau}+(x+2)^{1+\tau}+(x+2)^{\tau}+\sum_{j=0}^{\tau-1} \mathsf{key}_{i,j}(x+2)x^j}$$

is a one-to-one function.

Nonmalleability The analysis for the random case when the mauling function has degree greater than 1 are exactly the same as in Theorem 4.1. We focus on showing that the adversary cannot find a function linear f using linear combinations of the known values. We can think of the adversary being given values of following type:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & \mathsf{key}_{0,0} & \dots & \mathsf{key}_{0,\tau-1} \\ 1 & 1 & 1 & 1 & 1 & \mathsf{key}_{0,0} & \dots & \mathsf{key}_{0,\tau-1} \\ \dots & & & & & & & \\ 1 & 1 & 1 & 1 & 1 & \mathsf{key}_{n/\tau,0} & \dots & \mathsf{key}_{n/\tau,\tau-1} \end{bmatrix} \begin{bmatrix} r_{4+\tau} \\ r_{3+\tau} \\ r_{2+\tau} \\ r_{1+\tau} \\ r_\tau \\ \dots \\ r_1 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ \dots \\ c_{n/tau} \end{bmatrix}$$

Without loss of generality, we assume that an adversary can perfectly set/predict the powers x^j where $j < \tau$. However, to change the obfuscated point they will also need to change the higher order powers. We can think of the adversary having to find α, β, γ such that

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

We can write the desired linear combination as follows:

$$\begin{bmatrix} \alpha^{4+\tau} \\ \alpha^{3+\tau}\left(\binom{\tau+4}{1}\beta + \binom{\tau+3}{0}\right) \\ \alpha^{2+\tau}\left(\binom{\tau+4}{2}\beta^2 + \binom{\tau+3}{1}\beta + \binom{\tau+2}{0}\right) \\ \alpha^{1+\tau}\left(\sum_{i=0}^{3}\left(\binom{\tau+4-i}{3-i}\beta^{3-i}\right)\right) \\ \alpha^{\tau}\left(\sum_{i=0}^{4}\left(\binom{\tau+4-i}{4-i}\beta^{4-i}\right)\right) \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} r_{4+\tau} & 0 & 0 & 0 & 0 \\ 0 & r_{3+\tau} & 0 & 0 & 0 \\ 0 & 0 & r_{2+\tau} & 0 & 0 \\ 0 & 0 & 0 & r_{1+\tau} & 0 \\ 0 & 0 & 0 & 0 & r_\tau \end{bmatrix} = \gamma \begin{bmatrix} r_{4+\tau} \\ r_{3+\tau} \\ r_{2+\tau} \\ r_{1+\tau} \\ r_\tau \end{bmatrix}$$

Substituting, one has that

1. If $\beta = 0$ then this implies $\alpha^{\tau+4} = \alpha^{\tau+3} = \alpha^{\tau+2} = \alpha^{\tau+1} = \alpha^\tau$ which only has solutions if $\alpha = 0$ or $\alpha = 1$. These are both considered trivial solutions.
2. $\gamma = \alpha^{\tau+4}$ (using first equation),
3. $(\tau+4)\beta + 1 = \alpha$ (using second equation),
4. $(\tau+4)\beta = 2$ (using third equation, and relying on $\beta \neq 0$).
5. $\alpha = 3$ (substitution of third constraint into second equation)
6. $\gamma = 81$ (substitution of α in first equation)
7. $\tau = -5$ or $\tau = -114 * 31^{-1}$ (solving fourth equation using prior constraints).

Thus, using the first four equations we are able rule out all solutions unless $\tau = -5$ or $\tau = -114 * 31^{-1}$. Using the last equation we can almost always rule out this second possibility for τ. Namely, the fifth equation (recalling that $\beta = 2/(\tau+4)$)

$$\binom{\tau+4}{4}\left(\frac{2}{\tau+4}\right)^4 + \binom{\tau+3}{3}\left(\frac{2}{\tau+4}\right)^3 + \binom{\tau+2}{2}\left(\frac{2}{\tau+4}\right)^2 + (\tau+1)\frac{2}{\tau+4} + 1 = 81$$

This solution is only satisfied for $\tau = -114 * 31^{-1}$ if $191552 \equiv 0 \mod p$. Thus, it suffices to choose $p \notin \{2, 31, 41, 73\}$. This allows us to conclude that the adversary's value in the random case is linearly independent, which again leads to this value having entropy λ. Since the adversary only has to match a single value for each index by union bound their overall probability may be as high as $\tau/2^{\lambda}$. Thus, in both random cases the probability of mauling is at most $\chi/2^{(\lambda)}$. We note that since $\tau = O(\log \lambda)$ for large enough λ one can be sure that $\tau \not\equiv -5 \mod \mathbb{G}_{\lambda}$. This allows us to state the distinguishing capability of $\mathcal{A}$:

$$\Pr[\mathcal{A}(\{g^{x^i}\}_{i=1}^{4t}) = 1] - \Pr[\mathcal{A}(\{g^{r_i}\}_{i=1}^{4t}) = 1] \geq \epsilon - \frac{\tau}{2^{\lambda}}.$$

As in Theorem 4.1, this breaks the $(5+\tau)t$-Strong entropic power DDH assumption. This is a contradiction and so completes the proof of Theorem 5.1. □

6 Nonmalleable digital lockers from stocky lockers

We now use stocky lockers to construct a nonmalleable digital locker. 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 6.1. *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 6.2. *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 that the output only has to be statistical close to a distribution with minentropy rather than considering a uniform output. Importantly, it is possible to construct condensers where the adversary is allowed to output the chosen distribution after seeing the seed.

Definition 6.3. *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 $k' = \omega(\log \lambda)$. Furthermore, for key-nonmalleability we require that $\tilde{H}_{\infty}(X|\mathtt{cond}(X; seed)) \geq \omega(\log \lambda)$.

With all of this in mind, we now present the construction.

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

1. Compute $y = \mathtt{cond}(\mathsf{val}, \mathsf{seed})$.
2. Compute $z = \mathsf{Enc}(\mathsf{key}||y)$.
3. Initialize $\mathtt{Out} = \perp$.
4. For $i = 1$ to n compute:
   $r_i = \mathsf{stockLock}(\mathsf{val}, z_i)$,
   append $\mathtt{Out} = \mathtt{Out}||r_i$.
5. Output $\mathtt{Out}$.

$\mathsf{unlock}(\mathsf{val})$, input in $\{0,1\}^{\lambda}$:

1. Compute $y = \mathtt{cond}(\mathsf{val}, \mathsf{seed})$.
2. For $i = 1$ to n, input r_i:
   compute: $z_i = \mathsf{stockUnlock}(r_i, \mathsf{val})$
3. Run decode $\mathsf{key}' = \mathsf{Dec}(z)$.
4. If $\mathsf{key}'_{k\ldots k+n} \neq y$ output $\perp$.
   Else output $\mathsf{key}'_{0\ldots k-1}$.

Figure 1: nonmalleable digital locker preventing tampering over both val and key. A seed of a seed-dependent condenser must be public and global.

Construction 6.1. *Let* $(\mathsf{stockLock}, \mathsf{stockUnlock})$ *be a stocky locker. Let* $(\mathsf{Enc}, \mathsf{Dec})$ *be a coding scheme where* $\mathsf{Enc} : \{0,1\}^{k+\beta} \to \{0,1\}^n$. *Let* $\mathtt{cond} : \{0,1\}^{\lambda} \times \{0,1\}^d \to \{0,1\}^{\alpha}$ *be a seed-dependent condenser. Define the algorithms* $(\mathsf{lock}, \mathsf{unlock})$ *as in Figure 1.*

Theorem 6.1. *Let* $\lambda \in \mathbb{N}$ *be a security parameter and let* $\{0,1\}^{\lambda}$ *be the domain. Let* $(\mathsf{stockLock}, \mathsf{stockUnlock})$ *be a* $(\mathcal{F}_{single}, \mathcal{X}, \rho, \tau)$*-nonmalleable stocky locker that is nonmalleable for* val.

1. *Suppose for any* $s = \mathtt{poly}(\lambda)$ *there exists* $\beta = \omega(\log \lambda)$ *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* $\tilde{\mathrm{H}}_{\infty}(X|\mathsf{seed}, \mathtt{cond}(\mathsf{seed}, X)) \geq \mu = \beta$.
4. *Let a description of* $\mathbb{G}_{5\lambda}$*, a generator* g *for* $\mathbb{G}_{5\lambda}$ *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 6.1 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.*

Proof. Completeness, soundness, and point nonmalleability follow from the underlying properties of the stocky locker. We focus on showing nonmalleability of key. The core of our proof is a theorem (which may be of independent interest) that allows us to use nonmalleable codes in a nontraditional way where the adversary is provided with pseudorandom information that is correlated to the encoded codeword before choosing which function $f \in \mathcal{F}$ to tamper with. We first define an adaptive tampering experiment as follows for arbitrary distributions X, Y, Z and binary predicate $\mathtt{Test}$:

Experiment $\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F}_{nmc}, X, Y, Z, \mathcal{A}, \mathtt{Test}}$:
Sample $(x, y, z) \leftarrow (X, Y, Z)$
Sample $f \leftarrow \mathcal{A}(x)$.
If $f \notin \mathcal{F}_{nmc}$ output 0.
If $f(y) = y$ output 0.
If $\mathtt{Test}(f(y), z)$ output 1.
Else output 0.

Theorem 6.2. *Let $Z \in \{0,1\}^\alpha$ be a distribution such that $\mathrm{H}_\infty(Z) \geq \beta$. Let $(\mathsf{Enc}, \mathsf{Dec})$ be a $(\epsilon_{nmc}, s_{nmc}, \mathcal{F})$ nonmalleable code and $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n$ where $k = \alpha + \gamma$. Let $\mathsf{key} \in \{0,1\}^\gamma$, define the distribution Y_{key} by sampling $z \leftarrow Z$ and computing $\mathsf{Enc}(\mathsf{key}, z)$. For inputs y and z, define $\mathtt{Test}(y,z) = 1$ if and only if $\mathsf{Dec}(y)_{\gamma \ldots (\gamma+\alpha-1)} = z$. 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. *X be an arbitrary distribution over $\mathcal{M}$ such that*

$$\delta^{\mathcal{D}_{s_{pr}}}((X, Y_{\mathsf{key}}, Z), (U_{\mathcal{M}}, Y_{\mathsf{key}}, Z)) \leq \epsilon_{pr}.$$

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

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

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

The above theorem says that providing an adversary with some information X that may be correlated to the encoded codeword Y is not harmful as long as X is pseudorandom in the presence of Y. Crucially, it must be possible to test if the adversary tampers to an independent codeword. This necessitates the use of the auxiliary distribution Z that is part of the value encoded in Y. (In our construction Z is the output of a seed-dependent condenser applied to val.)

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

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

Lemma 6.1. *Suppose that $\mathrm{H}_\infty(Z) \geq \beta$, for any f, $\Pr[\mathbf{Exp}^{\mathrm{sim}}_{f,Z,D_f}(k) = 1] \leq 2^{-\beta}$.*

Proof of Lemma 6.1. We note that whenever $\tilde{s} = \mathtt{same}$ the output of the experiment is 0. Thus, we can restrict our attention to cases when $\tilde{s} \neq \mathtt{same}$. Then $\Pr[Z = \tilde{s}_{k-\alpha \ldots k}] \leq 2^{-\mathrm{H}_\infty(Z)} = 2^{-\beta}$. This completes the proof of Lemma 6.1. □

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, $\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}}$ and $\mathbf{Exp}^2_{\mathcal{F},X,Y,Z,\mathcal{A}}$. In moving from $\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F},X,Y,Z,\mathcal{A}}$ to $\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}}$ we will replace the distribution X with a random distribution that is uncorrelated to Y, Z. In $\mathbf{Exp}^2_{f,Y,Z}$ we will eliminate the uniform distribution as input and move from the adversary picking a function to defining the experiment for a particular function f. Finally in moving to $\mathbf{Exp}^{\mathrm{sim}}_{f,Z,D_f}(k)$ we will rely on the hardness of nonmalleable codes. The two experiments are described formally below.

Experiment $\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}}$:	**Experiment** $\mathbf{Exp}^2_{f,Y,Z}$:
Sample $(y,z) \leftarrow (Y,Z)$	Sample $(y,z) \leftarrow (Y,Z)$
Sample $u \leftarrow \mathcal{M}$.	If $f(y) = y$ output 0.
Sample $f \leftarrow \mathcal{A}(u)$.	Output $\mathtt{Test}(f(y), z)$.
If $f \notin \mathcal{F}$ output 0.	
If $f(y) = y$ output 0.	
Output $\mathtt{Test}(f(y), z)$.	

We now show each of these games are computationally close.

Lemma 6.2. *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 $s_{\mathcal{F},\mathtt{check}}$.*
3. *That $\delta^{\mathcal{D}_{s_{pr}}}((X, Y_{\mathsf{key}}, Z), (U_{\mathcal{M}}, Y_{\mathsf{key}}, Z)) \leq \epsilon_{pr}$.*

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

$$\left|\Pr\left[\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F},X,Y,Z,\mathcal{A}}(k) = 1\right] - \Pr[\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}}(k) = 1]\right| \leq \epsilon_{pr}.$$

Proof of Lemma 6.2. Suppose not. That is, suppose that there exists an $\mathcal{A}$ of size at most $s_{pr} - |\mathtt{Test}| - s_{\mathcal{F},\mathtt{eval}} - s_{\mathcal{F},\mathtt{check}}$ such that

$$|\Pr[\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F},X,Y,Z,\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}} = 1]| > \epsilon_{pr}.$$

Then the following program D (of size at most s_{pr}) is a distinguisher for $((X,Y,Z)$ and $(U_{\mathcal{M}},Y,Z))$:

1. On input x, y, z.
2. Run $f \leftarrow \mathcal{A}(x)$.
3. If $f \notin \mathcal{F}$ or $f(y) = y$ output 0.
4. Else output $\mathtt{Test}(f(y), z)$.

That is,

$$\begin{aligned}&|\Pr[D(X,Y,Z) = 1] - \Pr[D(U_{\mathcal{M}},Y,Z) = 1]| \\ &= \left|\Pr[\mathbf{Exp}^{\mathrm{ad-nmc}}_{\mathcal{F},X,Y,Z,\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}} = 1]\right| > \epsilon_{pr}.\end{aligned}$$

This contradicts the pseudorandomness of $X|(Y,Z)$ and completes the proof of Lemma 6.2. □

Lemma 6.3. *There exists some $f \in \mathcal{F}$ such that for any $\mathcal{A}$ (here $\mathcal{A}$ need not be computationally bounded):*

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

Proof of Lemma 6.3. 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

$$\Pr[\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}} = 1] = \sum_{f \in \mathcal{F}} \Pr\left[D_{\mathcal{A}} = f\right] \Pr_{(y,z) \leftarrow (Y,Z)}\left[f(y) \neq y \wedge \mathtt{Test}(y,z) = 1\right].$$

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

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

Then one has

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

This is a contradiction and completes the proof of Lemma 6.3. □

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 6.4. *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*

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

Proof of Lemma 6.4. 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 $\mathcal{D}_{s_{nmc}}$,

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

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

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

Proof of Lemma 6.5. Suppose not, that is suppose that there exists some $f \in \mathcal{F}$ such that

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

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 6.5. □

Combining Lemmas 6.1, 6.2, 6.3 and 6.5 completes the proof of Theorem 6.2. □

Application of Theorem 6.2 yields Theorem 6.1. □

Acknowledgements

The work of Peter Fenteany was funded by a grant from Comcast Inc. The work of Ben Fuller was funded in part by NSF Grant CNS 1849904. 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 discussion 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.

[BCM11] Mihir Bellare, David Cash, and Rachel Miller. Cryptography secure against related-key attacks and tampering. In *Advances in Cryptology – ASIACRYPT*, pages 486–503. Springer, 2011.

[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.

[BMZ18] James Bartusek, Fermi Ma, and Mark Zhandry. The distinction between fixed and random generators in group-based assumptions. 2018.

[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.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[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.

[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.

[KY18] 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.

[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.

[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.