# Non-malleable Digital Lockers
# for Efficiently Sampleable Distributions

Peter Fenteany*  Benjamin Fuller†

February 14, 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 is private if it provides an adversary with no information with high probability. An ideal digital locker would also detect if an adversary that mauls an obfuscation on one password and key into a new program that obfuscates a related password or key. Such a primitive is achievable in the random oracle model.

Komargodski and Yogev (Eurocrypt, 2018) constructed a simpler primitive - a *non-malleable point function* - which is a digital locker with no key.

This work describes the first non-malleable digital locker. This construction is built in two main steps:

1. Constructing non-malleable digital lockers for short keys. We present one construction for a single bit key and a second for a logarithmic length keys. These constructions can be safely composed with the same input password. This composed construction is non-malleable with respect to the password. Security relies on variants of the strong and power DDH assumptions.
2. An extension to polynomial length keys that additionally provides nonmalleability over the stored key. This extension combines the digital locker for short keys, non-malleable codes, and seed-dependent condensers. The password distribution can depend on the seed of the condenser as long as it is efficiently sampleable. The seed condenser must be public and random but programmability is not required.

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 non-malleable code.

**Keywords:** Digital Lockers; Point obfuscation; Virtual black-box obfuscation; Non-malleable codes; Seed-dependent condensers

## 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]. We do not consider indistinguishability obfuscation in this work [GGH+13, GGH+16, SW14, PST14, GLSW15, AJ15] [BR17]. Barak et

*Email: `peter.fenteany@uconn.edu`. University of Connecticut.
†Email: `benjamin.fuller@uconn.edu`. University of Connecticut.

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].

Our focus is on *digital lockers*. A digital locker obfuscator inputs a value, $\mathsf{val}$, and key, $\mathsf{key}$. The output is a program $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}(\cdot)$ which outputs $\mathsf{key}$ if and only if the input is $\mathsf{val}$. Privacy says $\mathsf{unlock}_{\mathsf{val},\mathsf{key}}$ should reveal nothing about $\mathsf{val}$ or $\mathsf{key}$ if the adversary cannot guess $\mathsf{val}$. 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 $\mathsf{val}$, and from then on indicates when $\mathsf{val}$ exactly is inputted to it. An obfuscated point function only needs to hide $\mathsf{val}$ [Can97]. It is possible to compose point functions to build a digital locker (if the point function retains security when composed) [CD08]. The construction for this is straightforward: for each bit of the $\mathsf{key}$, either a random point (corresponding to a 0 in $\mathsf{key}$) or $\mathsf{val}$ (corresponding to a 1) is obfuscated producing $\mathsf{unlockPoint}_{y_i}$. When running the program, the user runs each $\mathsf{unlockPoint}_{y_i}$, the program will fail when a random point was obfuscated and indicate if the true point was obfuscated. We call this digital locker the *real-or-random* construction.

**Nonmalleability** A desirable property of an obfuscated program is non-malleability. A *non-malleable* 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 non-malleable digital lockers and point functions is easy: for a random oracle $\mathsf{RO}$ one outputs the program $\mathsf{RO}(\mathsf{val}) \oplus (\mathsf{key}||\mathsf{RO}(\mathsf{key}))$, where $\mathsf{RO}(\mathsf{key})$ is truncated.

Recently, Komargodski and Yogev showed how to build a non-malleable point obfuscators [KY18]. Their construction follows. Let $g$ be a fixed group generator. To obfuscate the point $\mathsf{val}$, the obfuscator computes a random $r$ and outputs $\mathcal{O}(x) = (r, r^{g^{h(\mathsf{val})}})$, where $h(x) = x^4 + x^3 + x^2 + x$. 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.

Security of the construction relies on two variants of the Decisional Diffie-Hellman (DDH) assumption [DH76] - the strong DDH assumption (see [BC10]) and the power DDH assumption (introduced in [GJM02]). For an obfuscated point $\mathsf{val}$, the construction detects if the adversary creates an obfuscation of $f(\mathsf{val})$ from an obfuscation of $\mathsf{val}$ for polynomials $f$ of degree related to the assumed hardness in the power DDH assumption. While this primitive only confirms or denies equality of an input to a stored point, it is easy to see a connection between this goal and the desired functionality of digital lockers. This raises the question:

Do non-malleable digital lockers exist?

To this end, it seems natural to try and compose Komargodski and Yogev's construction for each bit of the key using the *real-or-random* construction. However, that approach does not ensure nonmalleability over the bits of $\mathsf{key}$. It is easy to permute bits of $\mathsf{key}$ by reordering group elements and to set individual bits of $\mathsf{key}$ to 0 by replacing a group element by a random group element. Consider a case when a user encrypts their files with $\mathsf{key}$. 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]. More is needed to achieve a non-malleable digital locker than what previous constructions provide.

## 1.1 Our Contribution

We present the first construction of a non-malleable digital locker. As a first technical step, we make a small but important change to the cryptographic primitive that is being composed. Instead of composing point functions, we compose digital lockers that natively store a small number of key bits. We present two constructions of digital lockers that support short keys and are secure when composed with the same $\mathsf{val}$: one that supports a single bit and the second for a logarithmic number of bits. This syntactic change allows the construction to actually read out symbols in $\mathsf{key}$ exactly rather than interpreting whether an obfuscation "opens."

**The single-bit digital locker** Point obfuscators output 1 on the correct input and 0 everywhere else. Our *single-bit digital locker* has three possible outputs: 1, 0, and $\perp$ (to indicate an incorrect point). To specify the output, the obfuscator now takes an additional bit $b$ as input. Our construction of a single-bit digital locker is:

$$\mathcal{O}(\mathsf{val}, b) = (r, r^{g^{h(2*\mathsf{val}+b)}}) = (r, r^{g^{h(\mathsf{val}||b)}}).$$

Here $h(x) = x^4 + x^3 + x^2 + x$ as in Komargodski and Yogev's point function obfuscator. With an input value $\mathsf{val}'$ the user attempts to "open" with both values of $b$. Since the underlying function $h$ is one-to-one, the construction remains correct. The intuitive argument for nonmallability of a single obfuscation is simple: if an adversary could change either $\mathsf{val}$ or $b$ that would correspond to a break of the original system.

However, creating a digital locker requires composition of several obfuscations. This small modification makes it more difficult to show nonmalleability (even just for functions on $\mathsf{val}$). This is because the adversary now gets access to obfuscations of distinct points $2\mathsf{val}$ and $2\mathsf{val} + 1$. Now the adversary may be able to compute a function $f'$ that takes $h(2\mathsf{val}), h(2\mathsf{val} + 1)$ and 1, yielding $h(2\mathsf{val}' + b')$. Despite this additional information provided to the adversary, we show it is still difficult to maul using a bounded degree polynomial.

In Theorem 3.2, we have to directly reduce to the strong power and strong vector Diffie-Hellman assumptions (defined in Section 2.1). As previously detailed, previous constructions of multi-bit point obfuscators [CD08, BC10] could be constructed by composing generic point obfuscators. However, we need to directly reduce to the number theoretic assumptions.

**Extending to nonbinary alphabet** Our second construction builds a digital locker that encodes a logarithmic number of bits in a pair of group elements. While a logarithmic length $\mathsf{key}$ can be "emulated" using the single bit construction, encoding a logarithmic number of bits at a time has important implications for ensuring nonmalleability of $\mathsf{key}$ (described next). Many non-malleable primitives consider non-binary symbols.

Suppose that we wish to encode $\tau$ 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 power DDH assumption (see Section 2.1) this requires one to show that $h(2\mathsf{val}' + b)$ is linearly independent of $h(2\mathsf{val})$ and $h(2\mathsf{val}+1)$. Since $h(x) = x^4+x^3+x^2+x$, the space spanned by different $h(x)$ is only dimension 4. As the adversary is given more linearly independent values $h(2^\tau * \mathsf{val} + \mathsf{key}_i)$ these span a larger dimensional space, and fresh values may be linearly dependent. To address this problem, we introduce a new hash function for symbols $y \in \{0,1\}^\tau$:

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

This construction ensures that all $2^\tau$ different values of $y$ span at most a $\tau$ dimensional subspace and cannot be used to predict the value of the hash function for any $x' \neq x$. There are a few notes about this new construction:

1. Opening checks each possible $y$ value, running time is proportional to $2^\tau$ 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 $\tau$ where nonmalleability is not guaranteed. This is because the choice of $\tau$ introduces a new degree of freedom to the linear system. The three values of $\tau$ are solutions to $\tau^3 - 15\tau^2 - 52\tau - 12 \equiv 0 \mod p$. It would be possible to avoid these $\tau$ by checking when $\tau$ is a solution for a particular $p$. Instead, our construction adds another power of $x$, which retains a single bad $\tau$ which is $\tau \equiv p - 5 \mod p$. This value of $\tau$ never occurs for logarithmic $\tau$.

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 $5\lambda$.

We note that neither of these constructions make any attempt to detect tampering of $\mathsf{key}$.

**Authenticating** $\mathsf{key}$ As mentioned, the *real-or-random* digital locker does not detect mauling of $\mathsf{key}$. One could prove (using a non-interactive zero-knowledge proof of knowledge) knowledge of $\mathsf{key}$ to keep the adversary from modifying the program. Boldyreva et al. considered this idea to build a non-malleable hash function [BCFW09]. A non-malleable 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]. However, in the digital locker application, no distribution is assumed over $\mathsf{key}$, the adversary may in fact know the locked value. As an alternative, one could apply a non-malleable hash to both $h(\mathsf{key}||\mathsf{val})$. 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 $\mathsf{key}$ and $\mathsf{val}$.

2. Any information leaked by $h(\mathsf{key}||\mathsf{val})$ makes it more difficult to show security of the digital locker. Digital lockers have to be simultaneously private for all distributions with super-logarithmic entropy.

Our authenticator combines seed-dependent condensers [DRV12] and non-malleable codes [DPW10]. Our strategy has two components: 1) use a non-malleable code to ensure an adversary can only tamper to independent values and 2) use a seed-dependent condenser to ensure an independent value is unlikely to authenticate. Our construction proceeds as follows:

1. First, we compute $d = \mathsf{cond}(\mathsf{val}, \mathsf{seed})$ where $\mathsf{cond}$ is a seed-dependent condenser. A seed dependent condenser has a high entropy output even if the distribution over $\mathsf{val}$ is adversarially dependent on the randomness $\mathsf{seed}$. Seed-dependent condensers for efficiently sampleable distributions are instantiable using collision-resistant hash functions [DRV12, Theorem 4.1]. We note that $\mathsf{val}$ can be correlated to $\mathsf{seed}$ as long as it is efficiently sampleable. Importantly, a random $\mathsf{seed}$ does need to be sampled for this system and be publicly known. Non-malleable extractors [DW09, CRS14] and non-malleable hashes [BCFW09] [BFS11, CQZ$^+$16] also require choosing a function randomly from some family and do not allow the adversary to change the function.

   We then compute $c = \mathsf{key}||\mathsf{cond}(\mathsf{val};\mathsf{seed})$. An adversary outputting any fixed $\tilde{c}$ 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 non-malleable code. Let $\mathcal{F}$ be some function class. A non-malleable code is a pair $\mathsf{Enc}$ and $\mathsf{Dec}$ where for functions $f \in \mathcal{F}$ the value $\tilde{s} = \mathsf{Dec}(f(\mathsf{Enc}(s)))$ is independent of $s$. We set

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

   In a non-malleable 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 a technical result that non-malleable codes can be used a nonstandard way where the tampering function is chosen after seeing the obfuscated values. Importantly, in Theorem 4.2, we can only show key non-malleability when the distribution over $\mathsf{val}$ has entropy conditioned on $\mathsf{cond}(\mathsf{val};\mathsf{seed})$. In contrast, privacy and point non-malleability hold for all distributions where $\mathsf{val}$ has unconditional entropy.

Non-malleable codes provide guarantees when the adversary tampers "obliviously." Our technical result allows us to argue their success probability when given the obfuscations does not deviate from oblivious tampering. In our approach, the only public randomness required is for $\mathsf{seed}$ of the condenser. Constructions based on non-malleable extractors [DW09, CRS14] and one-way hashes [BCFW09, BFS11, CQZ$^+$16] seem possible. However, these primitives would require a second public value (the extractor seed or hash function description). Lastly, security definitions for these primitives do not consider the case when the distribution of $\mathsf{val}$ depends on the description of the function.

**Choosing a non-malleable code** There are non-malleable codes that detect if the adversary permutes the bit vector, sets individual bits, or arbitrary functions that are applied separately to different parts of the encoded value. This class of of adversary is called a "split-state" adversary. More recently, Chattopadhyay and Li described a construction that detects tampering in the class $\mathtt{AC0}$ [CL17]. We recommend using a non-malleable code that detects at least setting and flipping of individual bits and permutations [AGM$^+$15a] [AGM$^+$15b]. One concern about non-malleable 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."

**Non-malleable codes with manipulation detection** Recently, Kiayias et al. [KLT18] introduced *non-malleable 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 non-malleable code with manipulation detection but their construction requires each symbol of the code to come from a polynomial size alphabet or equivalently for each symbol to have logarithmic length. Note this can be handled by our second construction. 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 non-malleable codes with manipulation detection are discovered, our construction can readily incorporate them.

**Open Questions** We present two main open questions resulting from this work. The first is whether our construction can be modified to use other non-malleable primitives such as extractors or one-way hashes. The second open question is for our multi-bit digital locker: would a different polynomial allow for a more compact group and thus more efficient operations? It seems necessary for the group size to scale with $\tau$ if the linear independence argument is used. It may be possible to compress group size by allowing imperfect correctness.

**Organization** The rest of this work is organized as follows Section 2 reviews definitions and computational assumptions, Section 3 introduces the single bit digital locker and shows security under composition for $\mathsf{val}$, Section 4 adds nonmalleability for $\mathsf{key}$, lastly Section 5 shows how to support multiple bits in each pair of group elements, enabling support for logarithmic size symbols.

## 2 Definitions and Background

For a 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. Usually, we use capitalized letters for random variables and corresponding lowercase letters for their samples.

**Single Bit Digital Locker** A point function is a function $I_{\mathsf{val}}$: $\{0, 1\}^n \mapsto \{0, 1\}$ outputs 1 on input $x$ and 0 elsewhere. An obfuscator preserves functionality while hiding the point $\mathsf{val}$ if $\mathsf{val}$ is not provided as input to the program. We will build a function that outputs a single bit when $\mathsf{val}$ is provided: $I_{\mathsf{val},b}$: $\{0, 1\}^n \mapsto \{\perp, 0, 1\}$. Here $I_{\mathsf{val},b}(\mathsf{val}) = b$ and $I_{\mathsf{val},b}(\mathsf{val}') = \perp$ for all other points $\mathsf{val}' \neq \mathsf{val}$. We call this primitive a *single-bit digital locker*. All definitions include a requirement of *polynomial slowdown*, which is omitted for space considerations. Running time of our constructions can be easily verified. We define a single-bit digital locker by adapting Komargodski and Yogev's definition [KY18]:

**Definition 2.1.** *For security parameter $\lambda \in \mathbb{N}$ an $n$-*composable same point ***single bit digital locker*** $\mathsf{lock}$ *is a probabilistic polynomial-time algorithm that inputs a point* $\mathsf{val} \in \{0,1\}^{\lambda-1}$ *and any bit string* $b \in \{0,1\}^n$*, and outputs a circuit* $\mathsf{unlock}$ *such that the following two conditions are met:*

1. ***Completeness****: For all $\lambda \in \mathbb{N}$, all $x \in \{0,1\}^{\lambda-1}$, and either $b \in \{0,1\}$, it holds that* $\Pr[\mathsf{unlock}(\cdot) \equiv I_{x,b}(\cdot) | \mathsf{unlock} \leftarrow \mathsf{lock}(x, b)] = 1$*, where the probability is over the randomness of* $\mathsf{lock}$.
2. ***Soundness****: For every probabilistic polynomial-time algorithm $\mathcal{A}$ 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* $\mathsf{val} \in \{0,1\}^{\lambda-1}$*, any bit string $b$, and for any $\mathcal{P} : \{0,1\}^\lambda \mapsto \{0,1\}$,*
$$\left|\Pr[\mathcal{A}(\{\mathsf{lock}(\mathsf{val}, b)\}_{i=1}^m) = \mathcal{P}(\mathsf{val}, b)] - \Pr[\mathcal{S}^{\{I_{\mathsf{val},b}\}_{i=1}^n}(1^\lambda) = \mathcal{P}(\mathsf{val}, b)]\right| \leq \frac{1}{p(\lambda)},$$
*where $S$ is allowed $q(\lambda)$ oracle queries to any of the oracles $\{I_{\mathsf{val},b_i}\}_{i=1}^n$ and the probabilities are over the internal randomness of $\mathcal{A}$ and* $\mathsf{lock}$*, and of $\mathcal{S}$, respectively. Here $I_{x,b_i}$ is an oracle that returns $b_i$ when provided input $x$ otherwise $I_{x,b_i}$ returns $\perp$.*

We note the above definition is *virtual grey-box obfuscation* as the simulator is allowed unbounded time but a limited number of queries. The definition of a point function is analagous with the removal of $b$ and a suitable change to the oracles $I$. Bitanski and Canetti [BC10] showed for digital lockers virtual grey box obfuscation and virtual black box obfuscation are equivalent.

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].

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

**Definition 2.2.** *A PPT algorithm $\mathcal{V}$ for a digital locker,* $\mathsf{lock}$*, for* $\mathsf{val} \in \{0,1\}^{\lambda-1}$*, $b \in \{0,1\}$ is called a* verifier *if for all $\lambda \in \mathbb{N}$ and all* $\mathsf{val} \in \{0,1\}^{\lambda-1}, b \in \{0,1\}$*, it holds that* $\Pr[\mathcal{V}(\mathsf{lock}(x, b)) = 1] = 1$*, (prob. over the randomness of $\mathcal{V}$ and* $\mathsf{lock}$*).*

Our constructions consist of pairs of group elements. Thus, in all applications our verification algorithm consists of checking if the string corresponds to two group elements. With this definition, we can define nonmalleability. Our definition includes the bit $b$, does not assume a distribution over $b$, and does not assume hardness of the adversary tampering with $b$.

**Definition 2.3.** *Let* $\mathsf{lock}$ *be a single bit digital locker for* $\mathsf{val} \in \{0,1\}^{\lambda-1}$ *with an associated verifier $\mathcal{V}$. Let $\mathcal{F} : \{0,1\}^{\lambda-1} \to \{0,1\}^{\lambda-1}$ be a family of functions and let $\mathcal{X}$ be a family of distributions over $\{0,1\}^{\lambda-1}$. A single bit digital locker* $\mathsf{lock}$ *is* non-malleable *for $\mathcal{F}$ and $\mathcal{X}$ if for any polynomial-time adversary $\mathcal{A}$, there exists a negligible function $\epsilon$, such that for all $b \in \{0,1\}$ it holds that:*

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

**Digital Locker** Our full construction is a non-malleable digital locker. In addition to defining a digital locker, we provide two definitions of non-malleability. The first ensures non-malleability over just the encoded point $\mathsf{val}$. The second provides non-malleability over both the encoded point $\mathsf{val}$ as well as $\mathsf{key}$. These definitions are used in Section 3 and Section 4 respectively.

**Definition 2.4.** *The algorithm* $\mathsf{lock}$ *with security parameter* $\lambda$ *is a* secure digital locker *if the following hold:*

- ***Correctness*** *For every* $\mathsf{key}$ *and* $\mathsf{val}$*,*

$$\Pr[\mathsf{unlock}(\cdot) \equiv I_{\mathsf{val},\mathsf{key}}(\cdot) | \mathsf{unlock} \leftarrow \mathsf{lock}(\mathsf{val}, \mathsf{key})] = 1.$$

- ***Security*** *For every PPT adversary* $\mathcal{A}$ *and every positive polynomial* $p$*, there exists a simulator* $S$ *and a polynomial* $q(\lambda)$ *such that for any sufficiently large* $\lambda$*, any polynomially-long sequence of values* $(\mathsf{val}_i, \mathsf{key}_i)$ *for* $i = 1, \ldots, \ell$*, and for any predicate* $\mathcal{P}$*,*

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

*where* $\mathcal{S}$ *is allowed* $q(\lambda)$ *oracle queries to the oracle* $I_{\mathsf{val},\mathsf{key}}$*.*

**Definition 2.5.** *Let* $\mathcal{F} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ *be a family of functions and let* $\mathcal{X}$ *be a family of distributions over* $\{0,1\}^\lambda$*. A digital locker,* $\mathsf{lock}$*, with security parameter* $\lambda$ *is* point non-malleable *for* $\mathcal{F}$ *if for any PPT* $\mathcal{A}$ *there exists a negligible function* $\epsilon$ *such that for all* $\mathsf{key} \in \{0,1\}^k$ *it holds that:*

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

**Definition 2.6.** *Let* $\mathcal{F} : \{0,1\}^\lambda \to \{0,1\}^\lambda, \mathcal{G} : \{0,1\}^n \to \{0,1\}^n$ *be families of functions and let* $\mathcal{X}$ *be a family of distributions over* $\{0,1\}^\lambda$*. A digital locker,* $(\mathsf{lock}, \mathsf{unlock})$*, with security parameter* $\lambda$ *is* point non-malleable *for* $\mathcal{F}$ *and* key *non-malleable for* $\mathcal{G}$ *if for any PPT* $\mathcal{A}$ *there exists a negligible function* $\epsilon$ *such that for all* $\mathsf{key} \in \{0,1\}^k$ *it holds that:*

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

## 2.1 Hardness Assumptions

Our constructions will rely on multiple decisional assumptions in suitable groups. The most well known of these is the decisional Diffie-Hellman (DDH) assumption, which says that for a prime $p$ for a generator $g$ of $\mathbb{Z}_p^*$ the tuple $(g, g^x, g^y, g^{xy})$ is computationally indistinguishable from $(g, g^x, g^y, g^u)$ where $x$, $y$, $u$ are uniform elements in $\mathbb{Z}_p^*$. Here, we specifically consider the Diffie-Hellman structure (as opposed to using, for example, elliptic curves) as we must use the properties of this specific group in order to prove security and non-malleability. We consider an ensemble of groups with efficient operations where $\mathbb{G}_\lambda$ is a group of prime order $p \in (2^\lambda, 2^{\lambda+1})$.

From this, the first assumption we will use is the strong $\ell$-vector assumption introduced by Bitansky 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 $\mathcal{X}_1, \ldots, \mathcal{X}_\ell$. 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 min-entropy.

**Definition 2.7.** *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* $1^\lambda$ *as input whose output is identically distributed as* $(X_\lambda, Y_\lambda)$*.*

2. *For all large enough* $\lambda \in \mathbb{N}$*, it has super-logarithmic conditional min-entropy. Namely,* $\tilde{\mathrm{H}}_\infty(X_\lambda | Y_\lambda) = \omega(\log \lambda)$*.*

**Assumption 2.1** ($t$-Strong Average Vector DDH)**.** *Let* $\ell = \mathtt{poly}(\lambda)$ *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* $\mathcal{X}, \mathcal{Y}$ *where* $\mathcal{X}$ *is a vector in* $(\mathbb{Z}_p^*)^t$ *that is average case well-spread 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* $((x_1, y_1), ..., (x_t, y_t)) \leftarrow (\mathcal{X}, \mathcal{Y})$ *and* $u_i$ *is sampled uniformly from* $\mathbb{Z}_p^*$*.*

We use this variant as it is conceptually cleaner but random variables with super logarithmic average min-entropy have worst-case super-logarithmic min-entropy with overwhelming probability.

The second assumption we will consider is a variant of the power DDH assumption. The $t$-power DDH assumption (introduced in [GJM02]) says that increasing powers of a single element $x$ are indistinguishable from uniformly random. That is, $(g, g^x, g^{x^2}, ..., g^{x^t})$ is pseudorandom for a uniformly random $x$. One can naturally extend the power DDH assumption assumption to the strong setting:

**Assumption 2.2** ($t$-Strong Average 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* $\mathcal{X}, \mathcal{Y}$*, the following holds for any* $s_{sec} = \mathtt{poly}(\lambda)$ *there exists* $\epsilon = \mathtt{ngl}(\lambda)$ *such that*

$$\delta^{s_{sec}}((g, g^{X_\lambda}, g^{X_\lambda^2}, ..., g^{X_\lambda^t}, Y_\lambda), (g, g^{r_1}, g^{r_2}, ..., g^{r_t}, Y_\lambda)) \leq \epsilon.$$

*where the distribution* $(X_\lambda, Y_\lambda)$ *is jointly sampled. Here the elements* $r_1, ..., r_t$ *are sampled uniformly at random from* $\mathbb{Z}_p^*$*.*

# 3 Non-malleability for the locked value

**Construction 3.1.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter and let* $\{0,1\}^\lambda$ *be the domain. Let* $\mathcal{F}_{t,\mathtt{poly}} \stackrel{def}{=} \{f : \{0,1\}^\lambda \to \{0,1\}^\lambda\}_{\lambda \in \mathbb{N}}$ *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* $x \in \{0,1\}^{\lambda-1}, b \in \{0,1\}$ *and randomness* $r \in \mathbb{G}_{5\lambda}$*. and outputs:*

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

*Given a program* $\mathsf{unlock}$ *consisting of two group elements* $r', y'$ *for test password* $x$ *and obfuscated program* $\mathsf{unlock}$*, the user runs both* $\mathsf{unlock}(x,0)$ *and* $\mathsf{unlock}(x,1)$*. That is, the user computes:*

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

*If* $\mathsf{unlock}(x,b)$ *outputs* $1$ *then the user outputs* $b$*. Otherwise,* $\perp$ *is the output.*

We will first show this construction is secure when the adversary receives a single instance and then discuss nonmalleability when composed. Throughout our discussion we use $h(z) = z^4 + z^3 + z^2 + s$ as shorthand for the polynomial being computed in the exponent. In order to prove security of this construction for a single obfuscation we reduce to the construction of Komgrodski and Yogev which computes $\mathsf{lockPoint}(x; r) = (r, r^{g^{h(x)}})$ for input $x \in \{0,1\}^\lambda$. The proof is deferred to Appendix A.

**Theorem 3.1.** *Suppose that* $\mathsf{lockPoint}(x; r) = (r, r^{g^{h(x)}})$ *is a non-malleable point function obfuscator for points* $x \in \{0,1\}^\lambda$ *for all distributions* $X = \mathcal{X}_\lambda$ *such that* $H_\infty(X) = \omega(\log \lambda)$*. Then* $\mathsf{lock}(x,b) = \mathsf{lockPoint}(2x+b)$ *is a single bit digital locker for inputs* $x \in \{0,1\}^{\lambda-1}, b \in \{0,1\}$ *for all distributions* $X$ *such that* $H_\infty(X) = \omega(\log \lambda)$*.*

### 3.1 Composing the Single Bit Construction

We now show this construction can be composed to built digital locker. This requires showing that soundness, completeness, and nonmalleability are preserved when the adversary is provided with single bit digital lockers that are correlated. There are two things that could go wrong when an adversary receives single bit digital lockers on correlated points: 1) the inclusion of correlated points 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 $g^{h(2x)}$ and $g^{h(2x+1)}$. That is, we don't rely on the generators $r_i$ in providing any protection against tampering. To show that privacy is preserved we rely on the $t$-strong vector DDH assumption. Our construction is a simple concatenation of the single bit digital locker but the proof is substantially more involved.

**Construction 3.2.** *Let all variables be as in Construction 3.1 and let* $\mathsf{key} \in \{0,1\}^n$ *be some arbitrary value. Then define* $\mathsf{lock}(\mathsf{val}, \mathsf{key})$ *as follows (initialize* $\mathtt{Out} = \perp$*): for* $i = 1$ *to* $n$ *compute:*

1. *Sample* $r_i \leftarrow \mathbb{G}_{5\lambda}$.
2. *Append* $\mathtt{Out} = \mathtt{Out}||(r_i, (r_i)^{g^{(2\mathsf{val}+\mathsf{key}_i)^4+(2\mathsf{val}+\mathsf{key}_i)^3+(2\mathsf{val}+\mathsf{key}_i)^2+(2\mathsf{val}+\mathsf{key}_i)}})$.

*Define* $\mathsf{unlock}(\mathsf{val})$ *as follows for input* $\{r_i, y_i\}_{i=1}^n$*:*
*For* $i = 1$ *to* $n$ *compute:*

$$\gamma_{i,0} = (2\mathsf{val})^4 + (2\mathsf{val})^3 + (2\mathsf{val})^2 + (2\mathsf{val})$$
$$\gamma_{i,1} = (2\mathsf{val}+1)^4 + (2\mathsf{val}+1)^3 + (2\mathsf{val}+1)^2 + (2\mathsf{val}+1)$$
$$P(x,0,i) = \left(r_i^{g^{\gamma_{i,0}}} \stackrel{?}{=} y_i.\right)$$
$$P(x,1,i) = \left(r_i^{g^{\gamma_{i,1}}} \stackrel{?}{=} y_i.\right)$$

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

**Theorem 3.2.** *Suppose that*

1. *The $n$-strong vector DDH assumption holds,*
2. *The $4t$-strong 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 3.2 is $n$-composable single bit digital locker that is point non-malleable for $\mathcal{F} = \{f | \deg(f) \leq t\}$ (excluding constant polynomials and the identity polynomial).*

*Proof.* We separately consider correctness, soundness, and nonmalleability. Correctness is a direct extension of correctness for the single bit digital locker whose correctness is shown in Theorem 3.1. (The crucial fact is that $g^{h(2x+b)}$ is a one-to-one function.)

**Privacy** 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 2.7). 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 effecting 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].

**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 $\mathsf{key}$ is a value known to both the reduction and the adversary. (That is, we do not rely on any uncertainty with respect to $\mathsf{key}$.) That is, we assume that there exists some $\mathsf{key}$ and a PPT $\mathcal{A}$ such that for all negligible functions $\epsilon$:

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

We show how to construct $\mathcal{A}'$ that breaks the $\tau = 4t$ strong power DDH assumption (Assumption 2.2). 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:

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

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

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

**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 $\alpha_i$ of $h(2f(x))$ and $\beta_i$ 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 1:** Construction of $\mathcal{A}'$ from $\mathcal{A}$

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 1. 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 $\mathsf{key}'$. Thus, given the real distribution $\mathcal{A}'$ outputs 1 with probability at least $\epsilon$.

**The random case** We now assume that each $z_i$ is a uniform and randomly distributed $s_i$ for $1 \le i \le \tau$. 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.

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 1). 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 $C_\alpha$ and $C_\beta$ 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 $C_\alpha$ or $C_\beta$ for some $i$ is at most $\Pr[\text{success}] \le \frac{2}{2^\lambda} = \frac{1}{2^{\lambda-1}}$.

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:

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

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

**Lemma 3.1.** *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 $a \in \mathbb{G}_{5\lambda}, b \in \{0, 1\}$,*

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

*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 3.1.* 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:

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

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 $\alpha_2$ 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 $\alpha_0, \alpha_1$ 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:

$$\begin{aligned} \alpha_0 &= 2a^2 - a^3, \\ \alpha_1 &= 10a^3 - 10a^2. \end{aligned}$$

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:

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

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 $2^\lambda$ 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 3.1. $\square$

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 is a contradiction and completes the proof of Theorem 3.2. □

# 4 Non-malleability for the key

In this section, we extend our construction to prevent tampering over both $\mathsf{val}$ and $\mathsf{key}$. We need several new tools including non-malleable codes and seed-dependent condensers. We introduce these in turn. We first need to introduce the notion of non-malleable codes introduced by Dziembowski, Pietrzak, and Wichs [DPW10].

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

**Definition 4.2.** *A coding scheme,* $(\mathsf{Enc}, \mathsf{Dec})$ *is called* $(\epsilon_{nmc}, s_{nmc}, \mathcal{F})$-non-malleable *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:*

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

**Seed Dependent Condensers** Seed dependent condensers were introduced by Dodis, Ristenpart, and Vadhan [DRV12]. The goal of a condenser is similar to a traditional randomness extractor except that rather than considering a uniform output, the output only has to be statistical close to a distribution with min-entropy. Importantly, it is possible to construct condensers where the adversary is allowed to output the chosen distribution after seeing the seed (called *seed-dependent*).

**Definition 4.3.** *Let* $\mathtt{cond} : \{0,1\}^\lambda \times \{0,1\}^d \to \{0,1\}^\alpha$ *be a* $(k, k', s, \epsilon)$ 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* $X_{\mathsf{seed}}$ *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 $\tilde{H}_\infty(X|\mathtt{cond}(X; \mathit{seed})) \geq \omega(\log \lambda)$.

## 4.1 The Construction

Intuitively, we can combine non-malleable codes and seed-dependent condensers to check if the adversary tampers over the $\mathsf{key}$ value. We use the locked point $\mathsf{val}$ as input to a seed dependent condenser as part of the value encoded in the non-malleable code. If the adversary tampers to an *independent value* there are unlikely to match the output of the condenser on the real $\mathsf{val}$.

| | |
|---|---|
| $\mathsf{lock}(\mathsf{val}, \mathsf{key})$, input in $\{0,1\}^{\lambda+k}$:<br>1. Compute $y = \mathtt{cond}(\mathsf{val}, \mathsf{seed})$.<br>2. Compute $z = \mathsf{Enc}(\mathsf{key}\|\|y)$.<br>3. Initialize $\mathtt{Out} = \perp$.<br>4. For $i = 1$ to $n$ compute:<br>(a) Sample random generator $r_i \leftarrow \mathbb{G}_{5\lambda}$.<br>(b) Compute $\gamma_i = (2\mathsf{val} + z_i)^4 + (2\mathsf{val} + z_i)^3 + (2\mathsf{val} + z_i)^2 + (2\mathsf{val} + z_i).$<br>(c) Append $\mathtt{Out} = \mathtt{Out}\|\| \ (r_i, (r_i)^{g^{\gamma_i}})$.<br>5. Output $\mathtt{Out}$. | $\mathsf{unlock}(\mathsf{val})$, input in $\{0,1\}^{\lambda}$:<br>1. Compute $y = \mathtt{cond}(\mathsf{val}, \mathsf{seed})$.<br>2. For $i = 1$ to $n$, input $r_i, y_i$ compute: $\gamma_{i,0} = \sum_{j=1}^{4} (2\mathsf{val})^j$, $\gamma_{i,1} = \sum_{j=1}^{4} (2\mathsf{val}+1)^j$, $P(x,0,i) = \left(r_i^{g^{\gamma_{i,0}}} \stackrel{?}{=} y_i.\right)$, $P(x,1,i) = \left(r_i^{g^{\gamma_{i,1}}} \stackrel{?}{=} y_i.\right)$<br>(a) If $P(x,b,i)$ outputs 1, set $z_i = b$. Otherwise output $\perp$.<br>3. Run decode $\mathsf{key}' = \mathsf{Dec}(z)$.<br>4. If $\mathsf{key}'_{k\ldots k+n} \neq y$ output $\perp$. Else output $\mathsf{key}'_{0\ldots k-1}$. |

**Figure 1:** Non-malleable digital locker preventing tampering over both val and key. A group generator $g$ and a seed of a seed-dependent condenser are global system parameters.

**Construction 4.1.** *Let $\lambda \in \mathbb{N}$ be a security parameter and let $\{0,1\}^\lambda$ be the domain.*

1. *Let $\mathcal{F}_{t,\mathtt{poly}}$ be the family of polynomial functions of degree at most $t$.*
2. *Let $(\mathsf{Enc}, \mathsf{Dec})$ be a coding scheme where $\mathsf{Enc} : \{0,1\}^{k+\beta} \to \{0,1\}^n$.*
3. *Let $\mathcal{G} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a group ensemble with efficient representation and operations where each $\mathbb{G}_{5\lambda}$ is a group of prime order $q \in (2^{5\lambda}, 2^{5\lambda+1})$.*
4. *$X$ is a distribution such that $\mathrm{H}_\infty(X) \geq \mu = \omega(\log \lambda)$. There is no requirement that $X$ is independent of system parameters (in particular seed) as long as it is efficiently samplable.*
5. *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.*
6. *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.*

*Define the algorithms $(\mathsf{lock}, \mathsf{unlock})$ as in Figure 1.*

**Theorem 4.1.** *Suppose that*

1. *The $n$-strong average vector DDH assumption holds,*
2. *The $4t$-strong average power DDH assumption holds,*
3. *The selected prime $p \notin \{2, 3, 5, 11, 17\}$.*
4. *Suppose that $\mu - \beta = \omega(\log \lambda)$.*
5. *The code $(\mathsf{Enc}, \mathsf{Dec})$ is an $(\epsilon_{nmc}, s_{nmc}, \mathcal{F}_{nmc})$ non-malleable code.*

*Then $(\mathsf{lock}, \mathsf{unlock})$ in Construction 4.1 is point non-malleable for $\mathcal{F}_{t,\mathtt{poly}}$ and key nonmallable for $\mathcal{F}_{nmc}$.*

*Proof.* Correctness, privacy, and point non-malleability follow using the same arguments as Theorem 3.2.

For non-malleability the condition on the distribution $X$ is changed. We now require that $X|(\mathtt{cond}(X; S), S)$ has min-entropy (recalling that $X$ can be chosen dependent on $S$). Note all other properties are ensured only if $X$ has min-entropy. The core of our proof is a theorem (which may be of independent interest) that allows us to use non-malleable 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** $\mathbf{Exp}^{\text{ad}-\text{nmc}}_{\mathcal{F}_{nmc},X,Y,Z,\mathcal{A},\mathtt{Test}}$:
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 4.2.** *Let $Z \in \{0,1\}^\alpha$ be a distribution such that $\mathrm{H}_\infty(Z) \geq \beta$. Let $(\mathsf{Enc}, \mathsf{Dec})$ be a $(\epsilon_{nmc}, s_{nmc}, \mathcal{F})$ non-malleable 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 $Y_{\mathsf{key}}$ 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*

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

*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 $\mathsf{val}$.)

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

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

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

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

We will now argue that the adversary in the adaptive adversary does not perform substantially better than in the simulated experiment. We use a hybrid argument with two intermediate games, $\mathbf{Exp}^1_{\mathcal{F},Y,Z,\mathcal{A}}$ and $\mathbf{Exp}^2_{\mathcal{F},X,Y,Z,\mathcal{A}}$. In moving from $\mathbf{Exp}^{\text{ad-nmc}}_{\mathcal{F},X,Y,Z,\mathcal{A}}$ 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 $\mathbf{Exp}^{\text{sim}}_{f,Z,D_f}(k)$ we will rely on the hardness of non-malleable 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 $\texttt{Test}(f(y), z)$. |
| If $f \notin \mathcal{F}$ output 0. | |
| If $f(y) = y$ output 0. | |
| Output $\texttt{Test}(f(y), z)$. | |

We now show each of these games are computationally close.

**Lemma 4.2.** *Suppose that*

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

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

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

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

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

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 $\texttt{Test}(f(y), z)$.

That is,

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

This contradicts the pseudorandomness of $X|(Y,Z)$ and completes the proof of Lemma 4.2. □

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

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

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

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

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

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

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

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

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

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

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

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

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

This contradicts security of the non-malleable code. □

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

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

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

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

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

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

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

Application of Theorem 4.2 yields Theorem 4.1. □

# 5 Encoding multiple key bits in each digital locker

In this section, we transform Construction 3.2 to support encoding multiple bits in each group element. The reason for this extension is to support non-malleable codes where $\mathcal{F}_{nmc}$ allows tampering over nonbinary symbols. We show how to support $\tau$ bits in each obfuscation at the cost of running time proportional to $2^\tau$. This increase in running time is due to exhaustively checking each possible value of the symbol. In addition, it allows a weaker vector DDH assumption at the cost of a stronger power DDH assumption. It thus allows a tradeoff between these two parameters.

**Construction 5.1.** *Let all variables be as in Construction 3.1, 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* $n/\tau$ *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* $n/\tau$*, 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* $\perp$*.*

**Nonmalleability over keys** This construction can be augmented using a seed-dependent condenser and a non-malleable code in the same method as in Construction 4.1. In theorem below we only consider nonmalleability over the locked $\mathsf{val}$ and assume no distribution over $\mathsf{key}$.

**Theorem 5.1.** *Suppose that*

1. *The* $\frac{n}{\tau}$*-strong average vector DDH assumption holds,*
2. *The* $(5 + \tau)t$*-strong average 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* $n/\tau$*-composable point obfuscation that is non-malleable for* $\mathcal{F} = \{f \mid \deg(f) \leq t\}$ *(excluding constant polynomials and the identity polynomial).*

*Proof of Theorem 5.1.* As before we separately consider correctness, soundness, and nonmalleability.

**Correctness** For correctness 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 $\mathsf{key}$ 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 $(x + 2) \in \{0,1\}^\lambda$.

**Privacy** The privacy argument for this construction is exactly the same as in Theorem 3.2 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 3.2. 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 $\alpha, \beta, \gamma$ 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:

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

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 $\alpha$ 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 $\tau$. 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 $\lambda$. 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 $\lambda$ one can be sure that $\tau \not\equiv -5 \mod \mathbb{G}_\lambda$. This allows us to state the distinguishing capability of $\mathcal{A}$:

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

In particular, this breaks the $(5+\tau)t$-strong average power DDH assumption. This is a contradiction and completes the proof of Theorem 5.1. □


## Acknowledgements

The work of Peter Fenteany was funded by a grant from Comcast Inc. The authors thank Luke Demarest, Pratyay Mukherjee, Alex Russell, and Mayank Varia for their helpful feedback.


## 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.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability. In *Annual Cryptology Conference*, pages 1–20. Springer, 2010.

[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.

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *Annual International Cryptology Conference*, pages 108–125. Springer, 2008.

[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.

[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.

[CL17] Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1171–1184. ACM, 2017.

[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.

[GJM02] Philippe Golle, Stanislaw Jarecki, and Ilya Mironov. Cryptographic primitives enforcing communication and storage complexity. In *International Conference on Financial Cryptography*, pages 120–135. Springer, 2002.

[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.

# A Analysis of One Time Security

*Proof of Theorem 3.1.* We separately argue correctness, soundness, and nonmalleability.

**Correctness** For correctness, it suffices to show that the pair $(r, r^{g^{h(2x+b)}})$ is unique for each choice of $x$ and $b$ is unique for each pair $x, b$. First note that $2x + b$ is injective in both variables. Then $h(x) = x^4 + x^3 + x^2 + x$ is injective, thus the value $h(2x + b)$ will also be unique for each $x, b$. Let $p$ be the prime corresponding to the chosen group $\mathbb{G}_{5\lambda}$. Then, it holds that $h(2x + b) \leq 2^{5\lambda}$ and thus, the value of $g^{h(2x+b)}$ is one-to-one. This ensures the pair $(r, r^{g^{h(2x+b)}})$ is unique for unique $x, b$. Correctness 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 $\mathsf{lock}$ is intended to secure it is contained in the set of distributions that $\mathsf{lockPoint}'$ is intended to secure. Similarly, let $Z \in \mathcal{Z}_\lambda$ and define the distribution $X$ using the probability density function $\Pr[X = x] = \Pr[Z = (2x+0)]+\Pr[Z = (2x+1)]$. Then $\mathrm{H}_\infty(X) \geq \omega(\log \lambda)$, thus $X \in \mathcal{X}_{\lambda-1}$.

We show soundness by contradiction. First assume that the construction described $\mathsf{lockPoint}(x) = (r, r^{g^{h(x)}})$ 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 our construction 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 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 $\mathsf{lock}$ 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 $\mathsf{lock}(x'||b)$ for $x = x'||b$. Any

**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 2:** Construction of $\mathcal{S}_{dl}$ from $\mathcal{S}_p$

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, and in particular they succeed with the same probability because we have exactly produced the probability distribution that $\mathcal{A}^*_{dl}$ is expecting. 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 $\lambda>\lambda_{\max,c}$,

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

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 2.

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:

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

This is a contradiction and completes the argument of soundness.

**Nonmalleability** We proceed by contradiction assuming that there exists some ensemble $X'_\lambda$ 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 $X_\lambda$ with the bit $b$ appended. Note that this is a valid ensemble for the obfuscator $\mathsf{lockPoint}$ In showing non-malleability, we are able to inherit the non-malleability guarantees of the underlying obfuscation, with a slight adjustment for bounds. This follows both for polynomial functions of $x$ as well as for bit flipping on $b$. We design $\mathcal{A}_p$ as follows:

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

There are two pieces to how $\mathcal{A}_p$ is using $\mathcal{A}_{dl}$. First, $\mathcal{A}_p$ is trying to produce a tampering function on $x \in \{0,1\}^\lambda$, while $\mathcal{A}^*$ is tampering on $x'$ contained in $x'||b$. Consider the example when $f(x) = 3x+1$ is output by the adversary. This is akin to the adversary producing a valid obfuscation of $6x+2+1$ or $6x+2$. The two functions $f_0$ and $f_2$ are for this difference. The second main difference is that the $\mathcal{A}_{dl}$ does not know if the last bit of $x$ will be 0 or 1 so they output 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. $\square$