On Continual Leakage of Discrete Log Representations

Shweta Agrawal* Yevgeniy Dodis† Vinod Vaikuntanathan‡ Daniel Wichs§

June 28, 2012

Abstract

Let $\mathbb{G}$ be a group of prime order q, and let $g_1, \ldots, g_n$ be random elements of $\mathbb{G}$. We say that a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ is a *discrete log representation* of some some element $y \in \mathbb{G}$ (with respect to $g_1, \ldots, g_n$) if $g_1^{x_1} \cdots g_n^{x_n} = y$. Any element y has many discrete log representations, forming an affine subspace of $\mathbb{Z}_q^n$. We show that these representations have a nice *continuous leakage-resilience* property as follows. Assume some attacker $\mathcal{A}(g_1, \ldots, g_n, y)$ can repeatedly learn L bits of information on arbitrarily many random representations of y. That is, $\mathcal{A}$ adaptively chooses polynomially many leakage functions $f_i : \mathbb{Z}_q^n \to \{0,1\}^L$, and learns the value $f_i(\mathbf{x}_i)$, where $\mathbf{x}_i$ is a *fresh and random* discrete log representation of y. $\mathcal{A}$ wins the game if it eventually outputs a valid discrete log representation $\mathbf{x}^*$ of y. We show that if the discrete log assumption holds in $\mathbb{G}$, then no polynomially bounded $\mathcal{A}$ can win this game with non-negligible probability, as long as the leakage on each representation is bounded by $L \approx (n-2)\log q = (1 - \frac{2}{n}) \cdot |\mathbf{x}|$.

As direct extensions of this property, we design very simple continuous leakage-resilient (CLR) one-way function (OWF) and public-key encryption (PKE) schemes in the so called "invisible key update" model introduced by Alwen et al. at CRYPTO'09. Our CLR-OWF is based on the standard Discrete Log assumption and our CLR-PKE is based on the standard Decisional Diffie-Hellman assumption. Prior to our work, such schemes could only be constructed in groups with a bilinear pairing.

As another surprising application, we show how to design the first leakage-resilient *traitor tracing* scheme, where no attacker, getting the secret keys of a small subset of decoders (called "traitors") *and* bounded leakage on the secret keys of all other decoders, can create a valid decryption key which will not be traced back to at least one of the traitors.

*UCLA. E-mail: `shweta@cs.ucla.edu`. Partially supported by DARPA/ONR PROCEED award, and NSF grants 1118096, 1065276, 0916574 and 0830803.

†NYU E-mail: `dodis@cs.nyu.edu`. Partially supported by NSF Grants CNS-1065288, CNS-1017471, CNS-0831299 and Google Faculty Award.

‡University of Toronto. E-mail: `vinodv@cs.toronto.edu`. Partially supported by an NSERC Discovery Grant, by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

§IBM Research, T.J. Watson. E-mail: `danwichs@us.ibm.com`

1 Introduction

Let $\mathbb{G}$ be a group of prime order q, and let $g_1, \ldots, g_n$ be random elements of $\mathbb{G}$. We say that a vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ is a *discrete log representation* of some some element $y \in \mathbb{G}$ with respect to $g_1, \ldots, g_n$ if $\prod_{i=1}^n g_i^{x_i} = y$. A basic and well-known property of discrete log representations says that, given one such discrete log representation, it is hard to find any other one, assuming the standard Discrete Log (DL) problem is hard. In various disguises, this simple property (and its elegant generalizations) has found a huge number of applications in building various cryptographic primitives, from collision-resistant hash functions and commitment schemes [Ped91], to actively secure identification schemes [Oka92], to chosen-ciphertext secure encryption [CS02], to key-insulated cryptography [DKXY02], to broadcast encryption [DF03], to traitor tracing schemes [BF99], just to name a few.

More recently, discrete log representations have found interesting applications in leakage-resilient cryptography [NS09, ADW09, KV09], where the secret key of some system is a discrete log representation $\mathbf{x}$ of some public y, and one argues that the system remains secure even if the attacker can learn some arbitrary (adversarially specified!) "leakage function" $z = f(\mathbf{x})$, as long as the output size L of f is just slightly shorter than the length of the secret $|\mathbf{x}| = n \log q$. Intuitively, these results utilize the fact that the actual secret key $\mathbf{x}$ still has some entropy even conditioned on the L-bit leakage z and the public key y, since the set of valid discrete log representations of y has more than L bits of entropy. On the other hand, the given scheme is designed in a way that in order to break it — with or without leakage — the attacker must "know" some valid discrete log representation $\mathbf{x}^*$ of y. Since the real key $\mathbf{x}$ still has some entropy even given z and y, this means that the attacker will likely know a different discrete log representation $\mathbf{x}^* \neq \mathbf{x}$, which immediately contradicts the discrete log assumption.[1]

Although very elegant, this simple argument only applies when the overall leakage given to the attacker is *a-priori upper bounded* by L bits, where L is somewhat less than the secret key length $n \log q$. Of course, this is inevitable without some change to the model, since we clearly cannot allow the attacker to learn the entire secret $\mathbf{x}$. Thus, when applied to leakage-resilient cryptography, so far we could only get *bounded-leakage-resilient* (BLR) schemes, where the bound L is fixed throughout the lifetime of the system. In contrast, in most applications we would like to withstand more powerful *continual leakage*, where one only assumes that the *rate* of leakage is somehow bounded, but the *overall leakage* is no longer bounded. To withstand continual leakage, the secret key must be continually *refreshed* in a way that: (a) the functionality of the cryptosystem is preserved even after refreshing the keys an arbitrary number of times, and yet, (b) one cannot combine the various leaked values obtained from different versions of the key to break the system. Such model of *invisible key updates* was formalized by Alwen et al. [ADW09]. In that model, one assumes the existence of a trusted, "leak-free" server, who uses some "master key" MSK to continually refresh the secret key in a way that it still satisfies the conflicting properties (a) and (b) above. We stress that the server is only present during the key updates, but not during the normal day-to-day operations (like signing or decrypting when the leakage actually happens). We will informally refer to this *continual-leakage-resilient* (CLR) model of "invisible key updates" as the *floppy model*, to concisely emphasize the fact that we assume an external leak-free storage (the

[1] This argument works for unpredictability applications, such as one-way functions. For indistinguishability applications, such as encryption, a similar, but slightly more subtle argument is needed. It uses the Decicional Diffie-Hellman (DDH) assumption in place of the DL assumption, as well as the fact that the inner product function is a good "randomness extractor" [CG88, NZ96].

"floppy" disk) which is only required for rare refreshing operations.[2]

We notice that all bounded leakage schemes based on discrete log representations naturally permit the following key refreshing procedure. The master key MSK consists of a vector of the discrete logarithms $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ of the generators $g_1, \ldots, g_n$ with respect to some fixed generator g. The refresh simply samples a random vector $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)$ orthogonal to $\boldsymbol{\alpha}$, so that $\prod g_i^{\beta_i} = g^{\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle} = 1$. The new DL representation $\mathbf{x}'$ of y is set to be $\mathbf{x}' := \mathbf{x} + \boldsymbol{\beta}$. It is easy to verify that $\mathbf{x}'$ is simply a fresh, random representation of y independent of the original DL representation $\mathbf{x}$. However, it is not obvious to see if this natural key refreshing procedure is *continual-leakage-resilient*. For the most basic question of key recovery,[3] this means that no efficient attacker $\mathcal{A}(g_1, \ldots, g_n, y)$ can compute a valid DL representation $\mathbf{x}^*$ of y despite (adaptively) repeating the following "L-bounded-leakage" step any polynomial number times. At period i, $\mathcal{A}$ chooses a leakage function $f_i : \mathbb{Z}_q^n \to \{0,1\}^L$, and learns the value $f_i(\mathbf{x}_i)$, where $\mathbf{x}_i$ is a *fresh and random* discrete log representation of y, as explained above.

Our Main Result. As our main conceptual result, we show that the above intuition is correct: *the elegant invisible key update procedure above for refreshing DL representations is indeed continual-leakage-resilient.* In other words, one can continually leak fresh discrete log representations of the public key, without affecting the security of the system. Moreover, the leakage bound L can be made very close to the length of our secret $\mathbf{x}$, as n grows: $L \approx (n-2)\log q = (1 - \frac{2}{n}) \cdot |\mathbf{x}|$.

Our proof crucially uses a variant of the *subspace-hiding with leakage* lemma from Brakerski et al. [BKKV10] (for which we also find an alternative and *much simpler* proof than that of [BKKV10]). In its basic form, this information-theoretic lemma states that, for a random (affine) subspace S of some fixed larger space U, it is hard to distinguish the output of a bounded-length leakage function $\mathsf{Leak}(s)$ applied to random sample $s \leftarrow S$, from the output of $\mathsf{Leak}(u)$ applied to random sample $u \leftarrow U$, even if the distinguisher can later learn the description of S *after* selecting the leakage function Leak. Given this Lemma, the overall high-level structure of our proof is as follows. Let U be the full $(n-1)$-dimensional affine space of valid discrete-log representations of y, and let S be a random $(n-2)$-dimensional affine subspace of U. Assume the attacker $\mathcal{A}$ leaks information on t different representations of y. In the original Game 0, all of the representations are sampled from the entire space U, as expected. In this case, the probability that $\mathcal{A}$ would output a representation $\mathbf{x}^* \in S$ is negligible since it gets no information about S during the course of the game and S takes up a negligible fraction U. We then switch to Game 1 where we give the attacker leakage on random representations from S rather than U. We do so in a series of hybrids where the *last* $i = 0, 1, \ldots, t$ representations are chosen from S and the first $t - i$ from U. We claim that, the probability of the attacker outputting a representation $\mathbf{x}^* \in S$ remains negligible between successive hybrids, which follows directly from the subspace-hiding with leakage lemma. Therefore, in Game 1, the attacker only sees (leakage on) representations in the small affine space S, but is likely to output a representation $\mathbf{x}^* \notin S$. This contradicts the standard DL assumption, as shown by an elegant lemma of Boneh and Franklin [BF99], which was proven in the context of traitor tracing schemes.

Applications. By extending and generalizing the basic CLR property of discrete log representations described above, we obtain the following applications.

First, we immediately get that the natural multi-exponentiation function $h_{g_1 \ldots g_n}(x_1 \ldots x_n) =$

[2]Another reason is to separate the floppy model from a more demanding CLR model of invisible updates subsequently introduced by [BKKV10, DHLW10], discussed in the Related Work paragraph below.

[3]For more powerful CLR goals (such as encryption and traitor tracing we discuss below), $\mathcal{A}$'s task could be more ambitious and/or $\mathcal{A}$ could get more information in addition to the public key and the leakage.

$g_1^{x_1} \dots g_n^{x_n}$ is a CLR one-way function (OWF) in the floppy model, under the standard DL assumption, with "leakage fraction" $L/|\mathbf{x}|$ roughly $1 - \frac{2}{n}$. This result elegantly extends the basic fact from [ADW09, KV09] that h is a bounded-leakage OWF with "leakage fraction" roughly to $1 - \frac{1}{n}$.

Second, we show that the Naor-Segev [NS09] bounded-leakage encryption scheme is also CLR-secure in the floppy model. The scheme is a very natural generalization of the ElGamal encryption scheme to multiple generators $g_1, \dots, g_n$. The secret key is $\mathbf{x}$, the public key is $y = g_1^{x_1} \dots g_n^{x_n}$, and the encryption of m is $(g_1^r, \dots, g_n^r, y^r \cdot m)$ (with the obvious decryption given $\mathbf{x}$). The scheme is known to be secure against bounded-leakage under the standard Decisional Diffie-Hellman (DDH) assumption. In this work, we examine the security of the scheme against continual leakage in the "floppy" model, with the same style of updates we described above for the one-way function. By carefully generalizing our one-wayness argument from DL to an indistinguishability argument from DDH, we show that this natural scheme is also CLR-secure in the floppy model.

As our final, and more surprising application, we apply our techniques to design the first leakage-resilient (public-key) *traitor tracing* (TT) scheme [CFN94, BF99]. Recall, in an N-user public-key traitor tracing scheme, the content owner publishes a public-key PK, generates N *individual* secret keys $\mathsf{SK}_1, \dots, \mathsf{SK}_N$, and keeps a special tracing key UK. The knowledge of PK allows anybody to encrypt the content, which can be decrypted by each user i using his secret key SK_i. As usual, the system is semantically secure given PK only. More interestingly, assume some T parties (so called "traitors") try to combine their (valid) secret keys in some malicious way to produce another secret key SK^* which can decrypt the content with noticeable probability. Then, given such a key SK^* and using the master tracing key UK, the content owner should be able to correctly identify at least one of the traitors contributing to the creation of SK^*. This non-trivial property is called *(non-black-box) traitor tracing*.

Boneh and Franklin [BF99] constructed a very elegant traitor tracing scheme which is semantically secure under the DDH assumption and traceable under the DL assumption. Using our new technique, we can considerably strengthen the tracing guarantee for a natural generalization of the Boneh-Franklin scheme. In our model, in addition to getting T keys of the traitors in full, we allow the attacker to obtain L bits of leakage on the keys of each of the $(N - T)$ remaining parties. Still, even with this knowledge, we argue the attacker cannot create a good secret key without the content owner tracing it to one of the traitors. We notice that, although our TT scheme is described in the bounded leakage model, where each user only gets one key and leaks L bits to the attacker, we can view the availability of N different looking keys as continual leakage "in space" rather than "time". Indeed, on a technical level we critically use our result regarding the continual leakage-resilience of DL representations, and our final analysis is considerably more involved than the analysis of our CLR-OWF in the floppy model.[4]

Related Work. As we mentioned, the floppy model was introduced by Alwen et al. [ADW09] as the extension of the BLR model considered by [AGV09, ADW09, NS09, KV09]. They observed that *bounded-leakage* signatures (and one-way *relations*) can be easily converted to the floppy model using any (standard) signature scheme. The idea is to have the floppy store the signing key sk for the signature scheme, and use it to authenticate the public key pk_i for the BLR signature scheme used in the i-th period. This certificate, along with the value of pk_i, is now sent with each BLR signature. Upon update, a completely fresh copy of the BLR scheme is chosen and certified. Unfortunately, this approach does not work for encryption schemes, since the encrypting party

[4] We believe that our TT scheme can also be extended to the floppy model; i.e., become continual both in "space" and "time". For simplicity of exposition, we do not explore this direction here.

needs to know which public key to use. In fact, it even does not work for maintaining a valid pre-image of a one-way *function* (as opposed to a one-way relation). In contrast, our work directly gives efficient and *direct* CLR one-way functions and encryption schemes.

Following [ADW09], Brakerski et al. [BKKV10] and Dodis et al. [DHLW10] considered an even more ambitious model for continual leakage resilience, where no leak-free device (e.g., "floppy") is available for updates, and the user has to be able to update his secret key "in place", using only fresh *local randomness*. Abstractly, this could be viewed as a "floppy" which does not store any long-term secrets, but only contributes fresh randomness to the system during the key update. In particular, [BKKV10, DHLW10] managed to construct signature and encryption schemes in this model. These works were further extended to the identity-based setting by [LRW11]. More recently, [LLW11, DLWW11] even constructed remarkable (but much less efficient) CLR encryption schemes where the attacker can even leak a constant fraction of the randomness used for each local key update. While the above encryption schemes do not require a "floppy", all of them *require a bi-linear group*, are based on the less standard/understood assumptions in bi-linear groups than the classical DL/DDH assumptions used here, and are generally quite less efficient than the simple schemes presented here. Thus, in settings where the existence of the "floppy" can be justified, our schemes would be much preferable to the theoretically more powerful schemes of [DHLW10, BKKV10, LRW11, LLW11, DLWW11].

More surprisingly, we point out that in some applications, such as traitor tracing considered in our work, the existence of local key updates is actually an *impediment* to the security (e.g., tracing) of the scheme. For example, the key updates used in prior bi-linear group CLR constructions had the (seemingly desirable) property that a locally updated key looks completely independent from the prior version of the same key. This held even if the prior version of this key is subsequently revealed, and irrespective of whatever trapdoor information the content owner might try to store a-priori. Thus, a single user can simply re-randomize his key without the fear of being traced later. In contrast, when a "floppy" is available, one may design schemes where it is infeasible for the user to locally update his secret key to a very "different" key, without the help of the "floppy". Indeed, our generalization of the Boneh-Franklin TT scheme has precisely this property, which enables efficient tracing, and which seems impossible to achieve in all the prior pairing-based schemes [DHLW10, BKKV10, LRW11, LLW11, DLWW11].

We also point out that the floppy model is similar in spirit to the key-insulated model of Dodis et al. [DKXY02], except in our model the "outside" does not know about the scheduling (or even the existence!) of key updates, so one cannot change the functionality (or the effective public key) of the system depending on which secret key is currently used.

Finally, although we mentioned much of the prior work with the most direct relation to our work, many other models for leakage-resilient cryptography have been considered in the last few years (see e.g., [ISW03, MR04, DP08] etc.). We refer the reader to [Wic11] and the references therein for a detailed discussion of such models.

2 Preliminaries

Below we present the definitions and lemmata that we will need. We begin with some standard notation.

2.1 Notation.

We will denote vectors by bold lower case letters (e.g., $\mathbf{u}$) and matrices by bold upper case letters (e.g., $\mathbf{X}$). For integers d, n, m with $1 \leq d \leq \min(n, m)$, we use the notation $\mathsf{Rk}_d(\mathbb{F}_q^{n\times m})$ to denote the set of all $n \times m$ matrices over $\mathbb{F}_q$ with rank d. If $\mathbf{A} \in \mathbb{F}_q^{n\times m}$ is a $n \times m$ matrix of scalars, we let $\mathsf{colspan}(\mathbf{A}), \mathsf{rowspan}(\mathbf{A})$ denote the subspaces spanned by the columns and rows of $\mathbf{A}$ respectively. If $\mathcal{V} \subseteq \mathbb{F}_q^n$ is a subspace, we let $\mathcal{V}^\perp$ denote the *orthogonal space* of $\mathcal{V}$, defined by $\mathcal{V}^\perp \stackrel{\text{def}}{=} \{\ \vec{w} \in \mathbb{F}_q^n \ \mid\ \langle \vec{w}, \vec{v}\rangle = 0\ \ \forall \vec{v} \in \mathcal{V}\ \}$. We write $(\vec{v}_1, \ldots, \vec{v}_m)^\perp$ as shorthand for $\mathsf{span}(\vec{v}_1, \ldots, \vec{v}_m)^\perp$. We let $\mathsf{ker}(\mathbf{A}) \stackrel{\text{def}}{=} \mathsf{colspan}(\mathbf{A})^\perp$. Similarly, we let $\mathsf{ker}(\boldsymbol{\alpha})$ denote the set of all vectors in $\mathbb{F}_q^n$ that are orthogonal to $\boldsymbol{\alpha}$.

If X is a probability distribution or a random variable then $x \leftarrow X$ denotes the process of sampling a value x at random according to X. If S is a set then $s \stackrel{\$}{\leftarrow} S$ denotes sampling s according to the *uniformly random* distribution over the set S. For a bit string $s \in \{0,1\}^*$, we let $|s|$ denote the bit length of s. We let $[d]$ denote the set $\{1, \ldots, d\}$ for any $d \in \mathbb{Z}^+$.

Throughout the paper, we let λ denote the *security parameter*. A function $\nu(\lambda)$ is called *negligible*, denoted $\nu(\lambda) = \mathsf{negl}(\lambda)$, if for every integer c there exists some integer N_c such that for all integers $\lambda \geq N_c$ we have $\nu(\lambda) \leq 1/\lambda^c$ (equivalently, $\nu(\lambda) = 1/\lambda^{\omega(1)}$).

Computational Indistinguishability. Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two ensembles of random variables. We say that X, Y are (t, ϵ)-indistinguishable if for every distinguisher D that runs in time $t(\lambda)$ we have

$$|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]| \leq \frac{1}{2} + \epsilon(\lambda).$$

We say that X, Y are *computationally indistinguishable*, denoted $X \stackrel{c}{\approx} Y$, if for every polynomial $t(\cdot)$ there exists a negligible $\epsilon(\cdot)$ such that X, Y are (t, ϵ)-indistinguishable.

Statistical Indistinguishability. The *statistical distance* between two random variables X, Y is defined by

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

We write $X \stackrel{s}{\approx}_\epsilon Y$ to denote $\mathbf{SD}(X, Y) \leq \epsilon$ and just plain $X \stackrel{s}{\approx} Y$ if the statistical distance is negligible in the security parameter. In the latter case, we say that X, Y are *statistically indistinguishable.*

Matrix-in-the-Exponent Notation: Let $\mathbb{G}$ be a group of prime order q generated by an element $g \in \mathbb{G}$. Let $\mathbf{A} \in \mathbb{F}_q^{n\times m}$ be a matrix. Then we use the notation $g^{\mathbf{A}} \in \mathbb{G}^{n\times m}$ to denote the matrix $\left(g^{\mathbf{A}}\right)_{i,j} \stackrel{\text{def}}{=} g^{(\mathbf{A})_{i,j}}$ of group elements. We will use a similar notational shorthand for vectors.

2.2 Computational Hardness Assumptions

We will rely on discrete-log type hardness assumptions in prime-order groups. We let such groups be defined via an abstract group generation algorithm $(\mathbb{G}, g, q) \stackrel{\$}{\leftarrow} \mathcal{G}(1^\lambda)$, where $\mathbb{G}$ is a (description of a) cyclic group of prime order q with generator g. We assume that the group operation, denoted by multiplication, can be computed efficiently.

Discrete Log Assumption. We say that the discrete log assumption holds for the group generation algorithm $\mathcal{G}$ if for every probabilistic polynomial time (PPT) adversary $\mathcal{A}$, there is a negligible function $\mu : \mathbb{N} \to [0, 1]$ such that the following holds:

$$\Pr[(\mathbb{G}, q, g) \xleftarrow{\$} \mathcal{G}(1^\lambda);\ \alpha \xleftarrow{\$} \mathbb{Z}_q;\ h \leftarrow g^\alpha;\ \alpha' \leftarrow A(g, h)\ :\ \alpha' = \alpha \pmod q] \le \mu(\lambda)$$

Decisional Diffie-Hellman (DDH). The DDH assumption on the group generation algorithm $\mathcal{G}$ states that:

$$\left\{(\mathbb{G}, q, g, g_0, g_1, g_0^r, g_1^r) :\ (\mathbb{G}, q, g) \xleftarrow{\$} \mathcal{G}(1^\lambda);\ x_0, x_1 \xleftarrow{\$} \mathbb{Z}_q;\ g_0 \leftarrow g^{x_0};\ g_1 \leftarrow g^{x_1};\ r \xleftarrow{\$} \mathbb{Z}_q\right\} \overset{c}{\approx}$$
$$\left\{(\mathbb{G}, q, g, g_0, g_1, g_0^{r_0}, g_1^{r_1}) :\ (\mathbb{G}, q, g) \xleftarrow{\$} \mathcal{G}(1^\lambda);\ x_0, x_1 \xleftarrow{\$} \mathbb{Z}_q;\ g_0 \leftarrow g^{x_0};\ g_1 \leftarrow g^{x_1};\ r_0, r_1 \xleftarrow{\$} \mathbb{Z}_q\right\}$$

Rank Hiding in the Exponent. The DDH assumption can be shown equivalent to the assumption that it is hard to distinguish between an n-by-m matrix $\mathbf{X}$ with rank $i \ge 1$ and one with rank $j > i$ in the exponent of a generator g of a prime order group $\mathbb{G}$ [NS09]. We call this the *rank hiding* assumption, defined formally below.

Let $\mathsf{Rk}_i(\mathbb{F}_q^{n\times m})$ denote the uniform distibution on all n-by-m matrices over $\mathbb{Z}_q$ of rank i.

Definition 1 (Rank Hiding Assumption)**.** The *rank hiding assumption* for a group generator $\mathcal{G}$ states that for any integers $i, j \in \mathbb{N}$ and $n, m \in \mathbb{N}$ satisfying $i, j \ge 1$, the following two distributions are computationally indistinguishable:

$$\left\{ \left(\mathbb{G}, q, g, g^{\mathbf{X}}\right) :\ (\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda), \mathbf{X} \xleftarrow{\$} \mathsf{Rk}_i(\mathbb{F}_q^{n\times m})\right\} \overset{c}{\approx}$$
$$\left\{ \left(\mathbb{G}, q, g, g^{\mathbf{X}}\right) :\ (\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda), \mathbf{X} \xleftarrow{\$} \mathsf{Rk}_j(\mathbb{F}_q^{n\times m})\right\}$$

We now postulate an apparent strengthening of the rank hiding assumption by requiring that a uniformly random rank-i matrix $\mathbf{X}_i \in \mathbb{Z}_q^{n\times m}$ and a uniformly random rank-j matrix, for $j > i$, $\mathbf{X}_j \in \mathbb{Z}_q^{n\times m}$ in the exponent are indistinguishable even given a number of vectors in the (left or right) kernel of $\mathbf{X}_i$, $\mathbf{X}_j$ in the clear (for $i, j \ge 1$). Clearly, the kernel of $\mathbf{X}_i$ (resp. $\mathbf{X}_j$) is a subspace of dimension $\min\{n, m\} - i$ (resp. $\min\{n, m\} - j$). Thus, if we are given $t > \min\{n, m\} - \max\{i, j\}$ (linearly independent) vectors $\mathbf{v}$ from the kernel, it is easy to distinguish whether the exponent contains a rank-i matrix or a rank-j matrix (note that it is easy to test whether a vector $\mathbf{v}$ given in the clear is in the kernel of the matrix $\mathbf{X}$ given in the exponent).

Surprisingly it turns out that for the case where $t \le \min\{n, m\} - \max\{i, j\}$, the additional vectors do not contribute any information about the rank of matrix in the exponent. This is captured formally in the *extended rank hiding assumption* which requires that one cannot distinguish between matrices of rank i and j in the exponent, even given $t \le \min\{n, m\} - \max\{i, j\}$ vectors in the kernel of the matrix. Although apparently rather strong, this assumption in fact turns out to be equivalent to DDH (and thus, also equivalent to the rank hiding assumption), as shown in [BKKV10].

Definition 2 (Extended Rank Hiding Assumption)**.** The *extended rank hiding assumption* for a group generator $\mathcal{G}$ states that for any integer constants $j > i \in \mathbb{N}$ and $n, m \in \mathbb{N}$ and $t \leq \min\{n, m\} - \max\{i, j\}$, the following two ensembles are computationally indistinguishable:

$$\left\{ \left(\mathbb{G}, q, g, g^{\mathbf{X}}, \mathbf{v}_1, \ldots, \mathbf{v}_t\right) : \ (\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda); \ \mathbf{X} \overset{\$}{\leftarrow} \mathsf{Rk}_i(\mathbb{F}_q^{n \times m}); \ \{\mathbf{v}_\ell\}_{\ell=1}^t \overset{\$}{\leftarrow} \mathsf{ker}(\mathbf{X}) \right\} \overset{c}{\approx}$$
$$\left\{ \left(\mathbb{G}, q, g, g^{\mathbf{X}}, \mathbf{v}_1, \ldots, \mathbf{v}_t\right) : \ (\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda); \ \mathbf{X} \overset{\$}{\leftarrow} \mathsf{Rk}_j(\mathbb{F}_q^{n \times m}); \ \{\mathbf{v}_\ell\}_{\ell=1}^t \overset{\$}{\leftarrow} \mathsf{ker}(\mathbf{X}) \right\}$$

Lemma 3. *The Extended Rank Hiding assumption is equivalent to the DDH assumption.*

Proof is implicit in [BKKV10].

Hardness of finding DL representation outside known span. We will also extensively use the following lemma of Boneh and Franklin, which states that given a number of discrete log representations of a group element h, an adversary cannot generate any other representation that is not in their span.

Lemma 4 ([BF99], Lemma 1)**.** *Let λ be the security parameter and let $(\mathbb{G}, q, g) \overset{\$}{\leftarrow} \mathcal{G}(1^\lambda)$. Under the discrete log assumption on the group generator $\mathbb{G}$, for every PPT adversary $\mathcal{A}$ and all integers $d = d(\lambda), n = n(\lambda)$ such that $d < n - 1$, there is a negligible function μ such that*

$$\begin{aligned}\Pr[&(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^\lambda); \ \boldsymbol{\alpha} \overset{\$}{\leftarrow} \mathbb{Z}_q^n; \ \beta \overset{\$}{\leftarrow} \mathbb{Z}_q; \ \mathbf{s}_1, \ldots, \mathbf{s}_d \overset{\$}{\leftarrow} \mathbb{Z}_q^n \text{ subject to } \langle \boldsymbol{\alpha}, \mathbf{s}_i \rangle = \beta; \\ &\mathbf{s}^* \leftarrow \mathcal{A}(\mathbb{G}, q, g, g^{\boldsymbol{\alpha}}, g^{\beta}, \mathbf{s}_1, \ldots, \mathbf{s}_d) : \mathbf{s}^* \notin \mathsf{span}(\mathbf{s}_1, \ldots, \mathbf{s}_d) \text{ and } \langle \boldsymbol{\alpha}, \mathbf{s}^* \rangle = \beta] \leq \mu(\lambda)\end{aligned}$$

where the probability is over the coins of $\mathcal{G}$ and the adversary $\mathcal{A}$ and all the random choices made in the experiment.

The above implies that any *valid* representation $\mathbf{s}^*$ that $\mathcal{A}(\mathbb{G}, q, g, g^{\boldsymbol{\alpha}}, g^{\beta}, \mathbf{s}_1, \ldots, \mathbf{s}_d)$ produces must lie in $\mathsf{span}(\mathbf{s}_1, \ldots, \mathbf{s}_d)$. In particualr, this means that $\mathbf{s}^*$ must be a *convex combination* of $\mathbf{s}_1, \ldots, \mathbf{s}_d$ (with coefficients summing up to 1) since only such combinations give valid representations.

2.3 Hiding Subspaces in the Presence of Leakage

In this section we prove various indistinguishability lemmas about (statistically) hiding subspaces given leakage on some of their vectors.

Hiding Subspaces. The following lemma says that, given some sufficiently small leakage on a random matrix $\mathbf{A}$, it is hard to distinguish random vectors from $\mathsf{colspan}(\mathbf{A})$ from uniformly random and independent vectors. A similar lemma was shown in [BKKV10, LLW11]. Here we give a significantly simpler proof using a variant of the *leftover-hash* lemma from [DORS08].

Lemma 5 (Subspace Hiding with Leakage)**.** *Let $n \geq d \geq u, s$ be integers, $\mathbf{S} \in \mathbb{Z}_q^{d \times s}$ be an arbitrary (fixed and public) matrix and $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^L$ be an arbitrary function with L-bit output. For randomly sampled $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times d}$, $\mathbf{V} \overset{\$}{\leftarrow} \mathbb{Z}_q^{d \times u}$, $\mathbf{U} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times u}$, we have:*

$$(\mathsf{Leak}(\mathbf{A}), \mathbf{AS}, \mathbf{V}, \mathbf{AV}) \overset{s}{\approx} (\ \mathsf{Leak}(\mathbf{A}), \mathbf{AS}, \mathbf{V}, \mathbf{U})$$

as long as $(d - s - u)\log(q) - L = \omega(\log(\lambda))$ and $n = \mathsf{poly}(\lambda)$.

Proof. The lemma follows by applying a variant of the leftover-hash lemma from [DORS08] to each row of $\mathbf{A}$ independently. In particular, take any row $\mathbf{a}_i$ of $\mathbf{A}$ and think of it as a random source (while all the other rows of $\mathbf{A}$ are arbitrarily fixed) whose conditional min-entropy is

$$\widetilde{\mathbf{H}}_\infty(\mathbf{a}_i \mid \mathbf{AS}, \mathsf{Leak}(\mathbf{A})) \geq d\log(q) - (s\log(q) + L).$$

Think of $\mathbf{V}$ as the seed of the universal hash function $h_{\mathbf{V}}(\mathbf{a}_i) = \mathbf{a}_i \cdot \mathbf{V}$ whose output size is $u\log(q)$ bits. The leftover-hash lemma tells us that the ith row of $\mathbf{AV}$ looks uniform. By using the hybrid argument over all n rows, the first part of the lemma follows.

□

We also show a dual version of Lemma 5, where a random matrix $\mathbf{A}$ is chosen and the attacker either leaks on random vectors in $\mathsf{colspan}(\mathbf{A})$ or uniformly random vectors. Even if the attacker is later given $\mathbf{A}$ in full, it cannot distinguish which case occurred. This version of "subspace hiding" was first formulated by [BKKV10], but here we present a significantly simplified proof and improved parameters.

Corollary 6 (Dual Subspace Hiding)**.** *Let $n \geq d \geq u$ be integers, and let $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^L$ be some arbitrary function. For randomly sampled $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times d}$, $\mathbf{V} \xleftarrow{\$} \mathbb{Z}_q^{d\times u}, \mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n\times u}$, we have:*

$$(\mathsf{Leak}(\mathbf{AV}), \mathbf{A}) \overset{s}{\approx} (\ \mathsf{Leak}(\mathbf{U}), \mathbf{A})$$

as long as $(d-u)\log(q) - L = \omega(\log(\lambda))$, $n = \mathsf{poly}(\lambda)$, and $q = \lambda^{\omega(1)}$.

Proof. We will actually prove the above assuming that $\mathbf{A}, \mathbf{V}, \mathbf{U}$ are random full-rank matrices, which is statistically close to the given statement since q is super-polynomial. We then "reduce" to Lemma 5.

Given $\mathbf{A}$ and $\mathbf{C}$ such that $\mathbf{C} = \mathbf{AV}$ or $\mathbf{C} = \mathbf{U}$, we can probabilistically choose a $n \times d'$ matrix A' depending only on $\mathbf{C}$ and a $n \times u'$ matrix $\mathbf{C}'$ depending only on $\mathbf{A}$ such that the following holds:

- If $\mathbf{C} = \mathbf{AV}$ for a random (full rank) $d \times u$ matrix $\mathbf{V}$, then $\mathbf{C}' = \mathbf{A}'\mathbf{V}'$ for a random (full rank) $d' \times u'$ matrix $\mathbf{V}'$.

- If $\mathbf{C} = \mathbf{U}$ is random (full rank) and independent of $\mathbf{A}$, then $\mathbf{C}' = \mathbf{U}'$ is random (full rank) and independent of $\mathbf{A}'$.

and where $d' = n - u$, $u' = n - d$. To do so, simply choose $\mathbf{A}'$ to be a random $n \times d'$ matrix whose columns form a basis of $\mathsf{colspan}(\mathbf{C})^\perp$ and choose $\mathbf{C}'$ to be a random $n \times u'$ matrix whose columns form a basis of $\mathsf{colspan}(\mathbf{A})^\perp$. If $\mathbf{C} = \mathbf{U}$ is independent of $\mathbf{A}$, then $\mathbf{C}' = \mathbf{U}'$ is a random full-rank matrix independent of $\mathbf{A}'$. On the other hand, if $\mathbf{C} = \mathbf{AV}$, then $\mathsf{colspan}(\mathbf{A})^\perp \subseteq \mathsf{colspan}(\mathbf{C})^\perp$ is a random subspace. Therefore $\mathbf{C}' = \mathbf{A}'\mathbf{V}'$ for some uniformly random $\mathbf{V}'$.

Now assume that our lemma does not hold and that there is some function Leak and an (unbounded) distinguisher D that has a non-negligible distinguishing advantage for our problem. Then we can define a function Leak' and a distinguished D' which breaks the problem of Lemma 5 (without even looking at AS, V). The function $\mathsf{Leak}'(A)$ samples C' as above and outputs $Leak = \mathsf{Leak}(C')$. The distinguisher D', given $(Leak, C)$ samples A' using C as above and outputs $D(Leak, A')$. The distinguisher D' has the same advantage as D. Therefore, by Lemma 5, indistinguishability holds as long as

$$(d' - u')\log(q) - L = \omega(\log(\lambda)) \Leftrightarrow (d-u)\log(q) - L = \omega(\log(\lambda))$$

□

It is also easy to extend the above corollary to the case where (the column space of) $\mathbf{A}$ is a subspace of some larger public space $\mathbf{W}$.

Corollary 7. *Let $n \geq m \geq d \geq u$. Let $\mathcal{W} \subseteq \mathbb{Z}_q^n$ be a fixed subspace of dimension m and let $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^L$ be some arbitrary function. For randomly sampled $\mathbf{A} \xleftarrow{\$} \mathcal{W}^d$ (interpreted as an $n \times d$ matrix), $\mathbf{V} \xleftarrow{\$} \mathbb{Z}_q^{d\times u}, \mathbf{U} \xleftarrow{\$} \mathcal{W}^u$ (interpreted as an $n \times u$ matrix), we have:*

$$(\mathsf{Leak}(\mathbf{AV}), \mathbf{A}) \overset{s}{\approx} (\ \mathsf{Leak}(\mathbf{U}), \mathbf{A})$$

as long as $(d-u)\log(q) - L = \omega(\log(\lambda))$, $n = \mathsf{poly}(\lambda)$, and $q = \lambda^{\omega(1)}$.

Proof. Let $\mathbf{W}$ be some $n \times m$ matrix whose columns span $\mathcal{W}$. Then we can uniquely write $\mathbf{A} = \mathbf{W}\mathbf{A}'$, where $\mathbf{A}' \in \mathbb{Z}_q^{m\times d}$ is uniformly random. Now we just apply Lemma 6 to $\mathbf{A}'$. □

A variant of the corollary holds also for affine subspaces. Namely:

Corollary 8. *Let $n \geq m \geq d \geq u$. Let $\mathcal{W} \subseteq \mathbb{Z}_q^n$ be a fixed subspace of dimension m and let $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^L$ be some arbitrary function and let $\mathbf{B} \in \mathbb{Z}_q^{n\times u}$ be an arbitrary matrix. For randomly sampled $\mathbf{A} \xleftarrow{\$} \mathcal{W}^d$ (interpreted as an $n \times d$ matrix), $\mathbf{V} \xleftarrow{\$} \mathbb{Z}_q^{d\times u}, \mathbf{U} \xleftarrow{\$} \mathcal{W}^u$ (interpreted as an $n \times u$ matrix), we have:*

$$(\mathsf{Leak}(\mathbf{AV} + \mathbf{B}), \mathbf{A}) \overset{s}{\approx} (\ \mathsf{Leak}(\mathbf{U}), \mathbf{A})$$

as long as $(d-u)\log(q) - L = \omega(\log(\lambda))$, $n = \mathsf{poly}(\lambda)$, and $q = \lambda^{\omega(1)}$.

3 One-Wayness of Discrete Log Representations under Continual Leakage

In this section, we show the one-wayness of discrete log representations under continual leakage. Namely, we show that for random $g_1, \ldots, g_n \xleftarrow{\$} \mathbb{G}$ and $h \xleftarrow{\$} \mathbb{G}$, obtaining leakage on many representations $\mathbf{x} = (x_1, ..., x_n)$ such that $\prod_{i=1}^n g_i^{x_i} = h$ does not help an efficient PPT adversary output any representation of h in terms of $g_1, \ldots, g_n$ in full (except with negligible probability) assuming that the discrete log assumption is true. Thus, in succinct terms, we show that discrete log representations are one-way under continual leakage, based on the (plain) discrete log assumption.

We first define the notion of a continual leakage resilient one-way function in the floppy model.

3.1 Defining One-Way Functions in Floppy Model

A continuous leakage resilient (CLR) one-way function in the Floppy Model (OWFF) consists of consists of the following PPT algorithms $(\mathsf{Gen}, \mathsf{Sample}, \mathsf{Eval}, \mathsf{Update})$:

1. $\mathsf{KeyGen}(1^\lambda)$ is a PPT algorithm that takes as input the security parameter λ and outputs the public parameters PP, the update key UK. The parameters PP are implicit inputs to all other algorithms and we will not write them explicitly for cleaner notation.

2. $\mathsf{Sample}(\mathsf{PP})$: Takes as input the public parameters PP and samples a random value $\mathbf{x}$.

3. $\mathsf{Eval}(\mathsf{PP}, \mathbf{x})$: This is a deterministic algorithm that takes as input $\mathbf{x}$ and outputs $\mathbf{y} \in \{0,1\}^*$.

4. $\mathsf{Update}(\mathsf{UK}, \mathbf{x})$ is a PPT algorithm that takes as input the update key UK and a string $\mathbf{x} \in \{0,1\}^*$ and outputs $\mathbf{x}' \in \{0,1\}^*$.

Correctness. We require that for any $(\mathsf{PP}, \mathsf{UK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and *any* $\mathbf{x} \in \{0,1\}^*$, we have

$$\mathsf{Eval}(\mathsf{Update}(\mathsf{UK}, \mathbf{x})) = \mathsf{Eval}(\mathbf{x}).$$

Security. Let $L = L(\lambda)$ be a function of the security parameter. We say that a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Update})$ is an L-CLR secure one-way function in the floppy model, if for any PPT attacker $\mathcal{A}$, there is a negligible function μ such that $\Pr[\mathcal{A} \text{ wins}] \leq \mu(\lambda)$ in the following game:

- The challenger chooses $(\mathsf{PP}, \mathsf{UK}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Next, it chooses a random element $\mathbf{x}_1 \leftarrow \mathsf{Sample}(\mathsf{PP})$ and sets $\mathbf{y} \leftarrow \mathsf{Eval}(\mathbf{x}_1)$. The challenger gives $\mathsf{PP}, \mathbf{y}$ to $\mathcal{A}$.
- $\mathcal{A}$ may adaptively ask for *leakage queries* on arbitrarily many pre-images. Each such query consists of a function (described by a circuit) $\mathsf{Leak} : \{0,1\}^* \rightarrow \{0,1\}^L$ with L bit output. On the ith such query Leak_i, the challenger gives the value $\mathsf{Leak}_i(\mathbf{x}_i)$ to $\mathcal{A}$ and computes the next pre-image $\mathbf{x}_{i+1} \leftarrow \mathsf{Update}(\mathsf{UK}, \mathbf{x}_i)$.
- $\mathcal{A}$ eventually outputs a vector $\mathbf{x}^*$ and *wins* if $\mathsf{Eval}(\mathbf{x}^*) = y$.

3.2 Constructing One-Way Function in the Floppy Model

We construct a one-way function $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{Sample}, \mathsf{Eval}, \mathsf{Update})$ as follows for some parameter $n = n(\lambda)$ which determined the amount of leakage that can be tolerated.

1. $\mathsf{KeyGen}(1^\lambda)$: Choose a group $\mathbb{G}$ of prime order q with generator g by running the group generation algorithm $\mathcal{G}(1^\lambda)$. Choose a vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \xleftarrow{\$} \mathbb{Z}_q^n$, and let $g_i = g^{\alpha_i}$ for $i \in [n]$. Output the parameters $\mathsf{PP} = (\mathbb{G}, g, g_1, \ldots, g_n)$ and the update key $\mathsf{UK} = \boldsymbol{\alpha}$.

2. $\mathsf{Sample}(\mathsf{PP})$: Sample a random vector $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_q^n$.

3. $\mathsf{Eval}(\mathsf{PP}, \mathbf{x})$: Parse $\mathbf{x} = (x_1, \ldots, x_n)$ and output $y := \prod_{i=1}^n g_i^{x_i}$.

4. $\mathsf{Update}(\mathsf{UK}, \mathbf{x})$: Choose a uniformly random vector $\boldsymbol{\beta} \xleftarrow{\$} \mathsf{ker}(\boldsymbol{\alpha})$, and output $\mathbf{x} + \boldsymbol{\beta}$.

Correctness follows from the fact that the inner product $\langle \mathbf{x} + \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle = \langle \mathbf{x}, \boldsymbol{\alpha} \rangle + \langle \boldsymbol{\beta}, \boldsymbol{\alpha} \rangle = \langle \mathbf{x}, \boldsymbol{\alpha} \rangle$, since $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are orthogonal (mod q).

Theorem 9. *Let $L = L(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter λ satisfying*

$$L < (n-2)\log(q) - \omega(\log(\lambda))$$

Then, $\mathcal{F}$ is an L-CLR secure one-way function in the floppy model (see definition 3.1) under the discrete log assumption for $\mathcal{G}$.

Proof. Suppose that the attacker has a non-negligible chance of winning the L-CLR-OWF game. Then, assuming that the DL assumption holds, we will arrive at a contradiction.

The proof proceeds by a sequence of games. Without loss of generality, assume that the attacker makes exactly T leakage queries.

Game 0: This is the security game in the definition of a CLR-one way function in the floppy model. Namely, the adversary is given the public parameters PP and $y = \mathsf{Eval}(\mathsf{PP}, \mathbf{x}_1)$, and asks a polynomial number of queries adaptively. Each query is a function $\mathsf{Leak}_i : \mathbb{Z}_q^n \to \{0,1\}^L$, in response to which the challenger returns $\mathsf{Leak}_i(\mathbf{x}_i)$ where, for $i > 1$, the ith preimage $\mathbf{x}_i$ is computed as $\mathbf{x}_i = \mathbf{x}_{i-1} + \boldsymbol{\beta}_i$ where $\boldsymbol{\beta}_i \xleftarrow{\$} \mathsf{ker}(\boldsymbol{\alpha})$.

By assumption, we have $\Pr[\mathcal{A} \text{ wins }] \geq \varepsilon(\lambda)$ for some non-negligible ε.

Game 1: Game 1 is defined as a sequence of $T+1$ sub-games denoted by Games $1.0, \ldots, 1.T$. For $i = 1, \ldots, T$, we have:

Game 1.i: In this game, the challenger chooses a random $(n-2)$-dimensional subspace $S \subseteq \mathsf{ker}(\boldsymbol{\alpha})$ in the beginning and answers the first $T - i$ queries differently from the last i queries as follows:

- For every $1 < j \leq T - i$, compute $\mathbf{x}_j = \mathbf{x} + \boldsymbol{\beta}_j$ where $\boldsymbol{\beta}_j \xleftarrow{\$} \mathsf{ker}(\boldsymbol{\alpha})$.
- For every $T - i < j \leq T$, compute $\mathbf{x}_j = \mathbf{x} + \mathbf{s}_j$ where $\mathbf{s}_j \xleftarrow{\$} S$.

In the above, we define $\mathbf{x} := \mathbf{x}_1$ to be the initial pre-image output by Sample.

Game 2: In Game 2, the challenger chooses all the vectors from the affine subspace $x + S$, i.e. it sets $\mathbf{x}_j = \mathbf{x} + \mathbf{s}_j$ where $\mathbf{s}_j \xleftarrow{\$} S,\ j \in [T]$.

Game 1.0 is identical to Game 0 since, in both games, all of the values $\mathbf{x}_i$ are just uniformly random over the affine space $\{\mathbf{x}_i \ :\ g^{\langle \mathbf{x}_i, \boldsymbol{\alpha} \rangle} = y\}$. By definition, Game $1.T$ is identical to Game 2.

In each of the games $1.i$, $i = 0, \ldots, T$, define the event E_i to be true if the adversary wins and returns a vector $\mathbf{x}^*$ such that $\mathbf{x}^* - \mathbf{x} \notin S$. Then, first we claim that in game 1.0, the probability of the event E_0 happening is negligibly close to ε.

Claim 10. There is a negligible function $\mu : \mathbb{N} \to [0,1]$ such that

$$\Pr[E_0] \geq \varepsilon(\lambda) - \mu(\lambda).$$

Proof. we have $\Pr[E_0] \leq \varepsilon(\lambda) - \Pr[\mathbf{x}^* - \mathbf{x} \in S] \leq 1/q$ over a random choice of S (since the adversary has no information about S in game 1.0). □

Next, we show that this probability does not change much across games:

Claim 11. There is a negligible function $\mu : \mathbb{N} \to [0,1]$ such that for every $1 \leq i \leq T$,

$$|\Pr[E_i] - \Pr[E_{i-1}]| \leq \mu(\lambda).$$

Proof. We have by Corollary 8, that as long as $L < (n-2)\log(q) - \omega(\log(\lambda))$ an attacker cannot distinguish leakage on $\boldsymbol{\beta}_i \xleftarrow{\$} \mathsf{ker}(\boldsymbol{\alpha})$ from leakage on $\mathbf{s}_i \xleftarrow{\$} S$, even if $\boldsymbol{\alpha}$ is public and known in the beginning and S becomes public *after* the leakage occurs. Therefore, knowing only $\boldsymbol{\alpha}$, we can simulate the first $i-1$ leakage queries for the attacker and then use leakage on the challenge vector ($\boldsymbol{\beta}_i$ or $\mathbf{s}_i$) to answer the ith query. We can then use knowledge of S (after the ith leakage query) to simulate the rest of the leakage queries and test if eventually the event (E_{i-1} or E_i) occurs. This proves the claim. □

Combining the above two claims and the observation that Game 2 is identical to Game 1.T, we have that there is a negligible function $\mu : \mathbb{N} \to [0, 1]$ such that in Game 2,

$$\Pr[\mathcal{A} \text{ wins and } \mathbf{x}^* - \mathbf{x} \notin S] \geq \varepsilon(\lambda) - \mu(\lambda)$$

.

Finally we show that the above contradicts the DL assumption.

Claim 12. If the Discrete Log assumption holds, then there is a negligible function $\mu : \mathbb{N} \to [0, 1]$ such that in Game 2,

$$\Pr[\mathcal{A} \text{ wins and } \mathbf{x}^* - \mathbf{x} \notin S] \leq \mu(\lambda)$$

Proof. Note that in Game 2, all the leakage queries of the adversary are answered using a randomly chosen $(n-2)$-dimensional subspace $S \subseteq \mathsf{ker}(\boldsymbol{\alpha})$, hence by Lemma 4 an adversary who outputs $\mathbf{x}^*$ such that $\mathbf{x}^* - \mathbf{x} \notin S$ can be transformed into one that solves the discrete log problem. □

Thus we arrive at a contradiction, which shows that under the Discrete Log assumption, the attacker could not have output $\mathbf{x}^*$ such that $f(\mathbf{x}^*) = \mathbf{y}$. Thus, $\mathcal{F}$ is an $L - CLR$ secure one way function for $L < (n-2)\log(q) - \omega(\log(\lambda))$. □

4 Public-key Encryption in the Continuous Leakage Model

In this section, we show the semantic security of the cryptosystems of Boneh et al. [BHHO08] and Naor and Segev [NS09] with continual leakage on the secret keys in the floppy model (i.e., with invisible updates) under the DDH assumption. We first define semantic security under continual leakage.

4.1 Defining Encryption in the Floppy Model

A CLR public key encryption scheme (CLR-PKE) in the Floppy Model consists of the following algorithms:

1. $\mathsf{KeyGen}(1^\lambda)$: Takes as input the security parameter λ and outputs the public key PK, the secret key SK and the update key UK.

2. $\mathsf{Update}(\mathsf{UK}, \mathsf{SK})$: Outputs an updated secret key SK'.

3. $\mathsf{Encrypt}(\mathsf{PK}, M)$: Outputs the ciphertext CT.

4. $\mathsf{Decrypt}(\mathsf{SK}, \mathsf{CT})$: Outputs the decrypted message M.

For convenience, we define the algorithm Update^i that performs $i \geq 0$ consecutive updates as:

$\mathsf{Update}^i(\mathsf{UK}, \mathsf{SK}) \to \mathsf{SK}'$: Let $\mathsf{SK}_0 = \mathsf{SK}, \mathsf{SK}_1 \leftarrow \mathsf{Update}(\mathsf{UK}, \mathsf{SK}_0), \ldots \mathsf{SK}_i \leftarrow \mathsf{Update}(\mathsf{UK}, \mathsf{SK}_{i-1})$. Output $\mathsf{SK}' = \mathsf{SK}_i$

Security. Let $L = L(\lambda)$ be a function of the security parameter. We say that a CLR PKE is L-CLR secure in the floppy model, if, for any PPT adversary $\mathcal{A}$, there is a negligible function μ such that $|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}| \leq \mu(\lambda)$ in the following game:

- Challenger chooses $(\mathsf{PK}, \mathsf{UK}, \mathsf{SK}_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- $\mathcal{A}$ may adaptively ask for *leakage queries* on arbitrarily many secret keys. Each such query consists of a function (described by a circuit) $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^L$ with L bit output. On the ith such query Leak_i, the challenger gives the value $\mathsf{Leak}_i(\mathsf{SK}_i)$ to $\mathcal{A}$ and computes the next updated key $\mathsf{SK}_{i+1} \leftarrow \mathsf{Update}(\mathsf{UK}, \mathsf{SK}_i)$.
- At some point $\mathcal{A}$ gives the challenger two messages M_0, M_1. The challenger chooses a bit $b \stackrel{\$}{\leftarrow} \{0,1\}$ and sets $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, M_b)$.
- The attacker $\mathcal{A}$ gets CT and outputs a bit $\tilde{b}$. We say $\mathcal{A}$ *wins* if $\tilde{b} = b$ with non-negligible probability.

4.2 Constructing Encryption in the Floppy Model

We define our scheme as follows for some parameter $n = n(\lambda)$ which determined the amount of leakage that can be tolerated.

1. $\mathsf{KeyGen}(1^\lambda)$: Let $(\mathbb{G}, q, g) \stackrel{\$}{\leftarrow} \mathcal{G}(1^\lambda)$. Choose vectors $\boldsymbol{\alpha} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$ and $\mathbf{x} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$, and let $f = g^{\langle \boldsymbol{\alpha}, \mathbf{x}_0 \rangle}$. The public parameters PK consists of $(g, f, g^{\boldsymbol{\alpha}})$.

   The update key $\mathsf{UK} = \boldsymbol{\alpha}$ and the secret key is set to $\mathsf{SK} = \mathbf{x} + \boldsymbol{\beta}$ where $\boldsymbol{\beta} \stackrel{\$}{\leftarrow} \mathsf{ker}(\boldsymbol{\alpha})$.

2. $\mathsf{Update}(\mathsf{UK}, \mathsf{SK})$: Choose $\boldsymbol{\beta} \stackrel{\$}{\leftarrow} \mathsf{ker}(\boldsymbol{\alpha})$, and output $\mathsf{SK} + \boldsymbol{\beta}$ as the updated secret key.

3. $\mathsf{Encrypt}(\mathsf{PK}, M)$: To encrypt $M \in \mathbb{G}$, pick a random scalar $r \stackrel{\$}{\leftarrow} \mathbb{Z}_q$. Output the ciphertext $\mathsf{CT} \leftarrow (g^{r\boldsymbol{\alpha}}, M \cdot f^r)$.

4. $\mathsf{Decrypt}(\mathsf{SK}, \mathsf{CT})$: Parse the ciphertext CT as $(g^{\mathbf{c}}, h)$ and output $h \cdot g^{-\langle \mathbf{c}, \mathsf{SK} \rangle}$ as the message.

A correctly formed ciphertext CT looks like $(g^{\mathbf{c}}, h) = (g^{r\boldsymbol{\alpha}}, M \cdot g^{r\langle \boldsymbol{\alpha}, \mathbf{x} \rangle})$. The secret key (after arbitrarily many updates) is $\mathsf{SK} = \mathbf{x} + \boldsymbol{\beta}$ where $\boldsymbol{\beta} \in \mathsf{ker}(\boldsymbol{\alpha})$. The decryption computes

$$h \cdot g^{-\langle \mathbf{c}, \mathbf{x} + \boldsymbol{\beta} \rangle} = M \cdot g^{r\langle \boldsymbol{\alpha}, \mathbf{x} \rangle} \cdot g^{-\langle r\boldsymbol{\alpha}, \mathbf{x} + \boldsymbol{\beta} \rangle} = M \cdot g^{r\langle \boldsymbol{\alpha}, \mathbf{x} \rangle} \cdot g^{-r\langle \boldsymbol{\alpha}, \mathbf{x} \rangle} = M$$

since $\langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle = 0 \pmod q$.

Theorem 13. *Let $L = L(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter λ satisfying*

$$L < (n-2)\log(q) - \omega(\log(\lambda))$$

Then, the public key encryption scheme (KeyGen, Update, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$) is L-CLR secure secure in the Floppy Model (see definition in Section 4.1) under the DDH assumption for $\mathcal{G}$.

Proof. The proof proceeds by a sequence of games. Without loss of generality, assume that the attacker makes exactly $T \in \mathsf{poly}(\lambda)$ leakage queries, and consider the following games:

Game 0: This is the semantic security game as in the definition in Section 4.1. In this game the secret keys $\{\mathsf{SK}_i\}_{i\in\{1,\ldots,T\}}$ are chosen uniformly and independently at random from an affine subspace $\mathbf{x} + \mathsf{ker}(\boldsymbol{\alpha})$ of dimension $n-1$. The challenge ciphertext $\mathsf{CT} := (g^{\mathbf{c}}, h)$ encrypting a message M is computed by choosing $r \xleftarrow{\$} \mathbb{Z}_q$ and setting

$$g^{\mathbf{c}} := g^{r\boldsymbol{\alpha}} \quad , \quad h := M \cdot f^r$$

The attacker is given $(\mathsf{PK}, \mathsf{CT}, \{\mathsf{Leak}_i(\mathsf{SK}_i)\}_{i=1}^{T})$ where the leakage functions Leak_i can be chosen adaptively by the adversary.

Game 1: This is the same as Game 0 except for the following changes: the challenger first chooses a random $(n-2)$-dimensional subspace $S \subset \mathsf{ker}(\boldsymbol{\alpha})$ and chooses all the secret keys $\{\mathsf{SK}_i\}$ from the affine subspace $\mathbf{x} + S$. Game 0 and Game 1 are statistically indistinguishable by Corollary 8 as long as $L < (n-2)\log(q) - \omega(\log(\lambda))$.

Game 2: In this game, the challenge ciphertext CT is computed using the secret key $\mathbf{x}$ by choosing $r \xleftarrow{\$} \mathbb{Z}_q$, setting

$$g^{\mathbf{c}} := g^{r\boldsymbol{\alpha}} \quad , \quad h := g^{\langle \mathbf{c}, \mathbf{x}\rangle} \cdot M$$

This retains the exact same distributions as in Game 0, since

$$f^r = g^{r\langle \boldsymbol{\alpha}, \mathbf{x}\rangle} = g^{\langle r\boldsymbol{\alpha}, \mathbf{x}\rangle} = g^{\langle \mathbf{c}, \mathbf{x}\rangle}$$

Therefore, this change is purely syntactical.

Game 3: In this game, during key generation, the challenger $\mathcal{C}$ chooses $\boldsymbol{\alpha}$ and $\mathbf{x}$ as before, but also a vector $\mathbf{c} \xleftarrow{\$} \mathbb{Z}_q^n$ and sets the space S to be the $(n-2)$-dimensional space $S = \mathsf{ker}(\boldsymbol{\alpha}, \mathbf{c})$ defined as

$$\mathsf{ker}(\boldsymbol{\alpha}, \mathbf{c}) = \{\boldsymbol{\beta} \in \mathbb{Z}_q^n : \ \langle \boldsymbol{\beta}, \boldsymbol{\alpha}\rangle = 0 \bmod q \text{ and } \langle \boldsymbol{\beta}, \mathbf{c}\rangle = 0 \bmod q\}$$

The secret keys $\{\mathsf{SK}_i\}$ are chosen as in the previous game from the space $\mathbf{x}+S$. The ciphertext is chosen as:

$$\mathsf{CT} = (g^{\mathbf{c}} \quad , \quad h := g^{\langle \mathbf{c}, \mathbf{x}\rangle} \cdot M).$$

using the same vector $\mathbf{c}$ that was chosen in the beginning and used to define S.

Note that the challenge ciphertext as produced in *Game 3* has the wrong distribution since it is unlikely that $\mathbf{c} \in \mathsf{span}(\boldsymbol{\alpha})$. However, it is still correctly decrypted to M by every version of the updated secret key, since the update vectors $\boldsymbol{\beta} \in S = \mathsf{ker}(\boldsymbol{\alpha}, \mathbf{c})$ are always orthogonal to $\mathbf{c}$ and hence do not affect decryption.

Games 2 and 3 are computationally indistinguishable by the extended rank-hiding assumption (which is equivalent to DDH by Lemma 3). To see this, think of the matrix $\mathbf{X} \in \mathbb{Z}_q^{2\times n}$ whose rows are $\boldsymbol{\alpha}$ and $\mathbf{c}$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_{n-2}$ be random vectors chosen via $\mathbf{v}_i \leftarrow \mathsf{ker}(\mathbf{X}) = \mathsf{ker}(\boldsymbol{\alpha}, \mathbf{c})$ and define $S = \mathsf{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{n-2})$. Now, given $g^{\mathbf{X}}$ and $\mathbf{v}_1, \ldots, \mathbf{v}_{n-2}$ we can simulate the public key and all of the secret keys that the attacker sees during the game, as well as the challenge ciphertext as defined in Game 3.

If $\mathbf{X} \xleftarrow{\$} \mathsf{Rk}_1(\mathbb{F}_q^{2\times n})$ then $\mathbf{c} = r\boldsymbol{\alpha}$ for a uniformly random $r \in \mathbb{F}_q$ and S is a random $n-2$ dimensional subspace of $\mathsf{ker}(\boldsymbol{\alpha})$ as in game *Game 2*. On the other hand, if $\mathbf{X} \xleftarrow{\$} \mathsf{Rk}_2(\mathbb{F}_q^{2\times n})$ then

$\boldsymbol{\alpha}$ and $\mathbf{c}$ are (statistically close to) random and independent and hence we get (statistically close to) the distribution of *Game 3*. Therefore, the two games are indistinguishable by the rank hiding assumption.

Game 4: In this game, we revert back to the original way that the secret keys were chosen. Namely, each of the secret keys SK_i that the attacker leaks on is chosen independently from the full $n-1$ dimensional space $x + \mathsf{ker}(\boldsymbol{\alpha})$.

Games 3 and 4 are statistically indistinguishable using *affine subspace hiding* (Corollary 8), using a proof technique similar to that in Theorem 9.

Game 5: In this game, the challenge ciphertext is completely independent of the message. Formally, $\mathcal{C}$ changes the second ciphertext component from $g^{\langle \mathbf{c},\mathbf{x}\rangle} \cdot M$ to g^v for some $v \xleftarrow{\$} \mathbb{Z}_q$. Thus the challenge ciphertext

$$\mathsf{CT}^* = (g^{\mathbf{c}}\ ,\ h := g^v).$$

This is statistically indistinguishable from the previous game since Game 4 does not reveal anything about $\mathbf{x}$ beyond the inner product $\langle \mathbf{x}, \boldsymbol{\alpha}\rangle$, and hence the inner product $\langle \mathbf{c}, \mathbf{x}\rangle$ is statistically close to uniform.

We showed that the game where the adversary is given the encryption of a message M_0 (resp. M_1) is computationally indistinguishable from one where the adversary is given a uniformly random sequence of group elements. This suffice to prove semantic security with leakage in the floppy model. □

5 Traitor Tracing in the Bounded Leakage Model

In this section, we generalize the constructions in Section 3 and Section 4 to obtain "leaky" traitor tracing in the bounded leakage model, which could be viewed as continual leakage-resilience in "space" rather than "time", but with strong traitor tracing properties. First, we define traitor tracing and associated security notions.

5.1 Definition of Traitor Tracing

The traitor tracing scheme is given by the following algorithms:

1. $\mathsf{KeyGen}(1^\lambda; 1^N, 1^T) \to \mathsf{PK}, \mathsf{SK}_1, \ldots, \mathsf{SK}_N$: Takes as input the security parameter λ, number of parties N, and number of traitors T. Outputs the public key PK, and secret keys $\{\mathsf{SK}_i\}_{i=1}^N$ for each party $i \in [N]$.

2. $\mathsf{Encrypt}(\mathsf{PK}, M) \to \mathsf{CT}$: Takes as input the public key PK, a message M and outputs the ciphertext CT.

3. $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{CT}, \mathsf{SK}) \to M$: Takes as input the public key PK, a ciphertext CT and a secret key SK and outputs a message M.

4. $\mathsf{Trace}(\mathsf{PK}, \mathsf{SK}^*) \to i$: Takes as input the public key PK, and some secret key SK^* and outputs an index $i \in [N]$ corresponding to an accused traitor.

Note that the tracing algorithm takes a valid secret key SK^* as input, and this is what makes the scheme *non black box*. This assumes that if the traitors collude and construct a "pirate decoder" that decrypts the encrypted content, then one can always extract the decryption key from this decoder. The stronger notion of *black box* traitor tracing only assumes that one can test whether the pirate decoder plays the encrypted content or not.

Correctness: For any integers $N, T, U, i \in [N]$ any $\mathsf{PK}, \mathsf{TK}, \mathsf{SK}_1, \ldots, \mathsf{SK}_N \leftarrow \mathsf{KeyGen}(1^\lambda; 1^N, 1^T)$, $\mathsf{CT} \leftarrow \mathsf{Encrypt}(M, \mathsf{PK})$ and $M' \leftarrow \mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}_i)$: we have $M' = M$.

We define security in terms of two properties: semantic security and tracing security.

Semantic Security: The standard notion of semantic security requires that, for any PPT $\mathcal{A}$, we have $|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}| \leq \mu(\lambda)$ in the following game:

- Attacker $\mathcal{A}$ chooses the values $1^N, 1^T$ to the challenger.
- Challenger chooses $(\mathsf{PK}, \mathsf{SK}_1, \ldots, \mathsf{SK}_N)$ and gives PK to $\mathcal{A}$.
- At some point $\mathcal{A}$ gives the challenger $\mathcal{C}$ two messages M_0, M_1.
- The challenger chooses a bit $b \leftarrow \{0,1\}$ at random and set $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, M_b)$.
- The attacker $\mathcal{A}$ gets CT and outputs a bit $\tilde{b}$. We say $\mathcal{A}$ *wins* if $\tilde{b} = b$.

Tracing Security: To define non-black-box tracing, we first define the predicate $\mathsf{GOOD}(\mathsf{PK}, \mathsf{SK})$ which holds iff there exists some message M in message-domain such that

$$\Pr[M' = M \ : \ \mathsf{CT} \leftarrow \mathsf{Encrypt}(M, \mathsf{PK}), M' \leftarrow \mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK})] \geq \frac{1}{2}.$$

In other words, a key SK is *good* if it succeeds in decryping at least some message M with probability at least a $\frac{1}{2}$. We say that *leakage-resilient traitor tracing* security holds if, for any PPT $\mathcal{A}$, we have $\Pr[\mathcal{A} \text{ wins }] \leq \mu(\lambda)$ in the following game:

- Attacker $\mathcal{A}$ chooses the values $1^N, 1^T$.
- Challenger $\mathcal{C}$ chooses $(\mathsf{PK}, \mathsf{SK}_1, \ldots, \mathsf{SK}_N)$ and gives PK to $\mathcal{A}$.
- $\mathcal{A}$ may adaptively ask $\mathcal{C}$ for the following type of queries:
  - **Leakage queries:** Attacker gives a user index $i \in [N]$ and a function (defined by a circuit) $\mathsf{Leak} \ : \ \{0,1\}^* \rightarrow \{0,1\}^L$ with L bit output. If no leakage query for user i was made before, then the challenger outputs $\mathsf{Leak}(\mathsf{SK}_i)$ and otherwise it ignores the query.
  - **Corrupt Queries:** Attacker asks for user index i and gets SK_i.
- At some point $\mathcal{A}$ outputs some SK^* and the challenger runs $i \leftarrow \mathsf{Trace}(\mathsf{PK}, \mathsf{SK}^*)$. We say that $\mathcal{A}$ wins if all of the following conditions hold: (1) $\mathcal{A}$ made at most T corrupt queries throughout the game, (2) the predicate $\mathsf{GOOD}(\mathsf{PK}, \mathsf{SK}^*)$ holds, (3) the traced index i was *not* part of any corrupt query.

Before presenting the encryption scheme, we review some necessary notions from the theory of error correcting codes.

Error Correcting Code. For traitor tracing with N parties and T traitors, we will rely on an $[N, K, 2T+1]_q$-linear-ECC over $\mathbb{F}_q$, where K is chosen as large as possible. For the Reed-Solomon code, we can set $K = N - 2T$, which we will assume from now on. Therefore, we will also assume that $N > 2T$ (this is without loss of generality as we can always increase N by introducing "dummy users" if necessary). Let $\mathbf{A}$ be a generation matrix and $\mathbf{B}$ be a parity check matrix so that $\mathbf{BA} = \mathbf{0}$. Note that $\mathbf{B}$ is a $2T \times N$ matrix. Lastly, we will assume *efficient syndrome-decoding* so that we can efficiently recover a vector $\mathbf{e} \in \mathbb{Z}_q^N$ from $\mathbf{B} \cdot \mathbf{e}$ as long as the hamming-weight of $\mathbf{e}$ is less than T. This holds for the Reed-Solomon code.

The Scheme. We now present our Traitor-Tracing scheme which is a natural generalization of the Boneh-Franklin scheme [BF99]. The scheme is defined as follows for some parameter $n = n(\lambda)$.

1. $\mathsf{KeyGen}(1^\lambda, 1^N, 1^T) \to \mathsf{PK}, \mathsf{SK}_1, \ldots, \mathsf{SK}_N$:
   Choose $(\mathbb{G}, q, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$. Choose $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{Z}_q^n$ and $\beta \xleftarrow{\$} \mathbb{Z}_q$.
   Let $\mathbf{B}$ be the parity-check matrix of an $[N, K, 2T+1]_q$-ECC as described above and let us label its columns by $\mathbf{b}_1, \ldots, \mathbf{b}_N$ where $\mathbf{b}_i \in \mathbb{Z}_q^{2T}$ for $i \in [N]$.
   For $i \in [N]$, choose $\mathsf{SK}_i = (\mathbf{b}_i || \mathbf{x}) \in \mathbb{Z}_q^n$ where $\mathbf{x} = (x_1, \ldots, x_{n-2T})$ and is constructed choosing $x_2, \ldots, x_{n-2T}$ uniformly random and uniquely fixing x_1 so that $\langle \boldsymbol{\alpha}, \mathsf{SK}_i \rangle = \beta$. Set $\mathsf{PK} := [\ g, g^{\boldsymbol{\alpha}} = (g_1, \ldots, g_n), f = g^\beta, \mathbf{B}]$.

2. $\mathsf{Encrypt}(\mathsf{PK}, M) \to \mathsf{CT}$: Choose a random $r \xleftarrow{\$} \mathbb{Z}_q$. Output $\mathsf{CT} \leftarrow (g^{r\boldsymbol{\alpha}}, f^r \cdot M)$

3. $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{CT}, \mathsf{SK}) \to M$: Let $\mathsf{CT} = (g^{\mathbf{c}}, h)$. Output $hg^{-\langle \mathbf{c}, \mathsf{SK} \rangle}$.

4. $\mathsf{Trace}(\mathsf{PK}, \mathsf{SK}^*) \to i$: Check that the input is a valid key SK^* satisfying $g^{\langle \boldsymbol{\alpha}, \mathsf{SK}^* \rangle} = f$. To trace, do the following: (1) Write $\mathsf{SK}^* = (\mathbf{b}^* || \mathbf{x}^*)$. (2) Use syndrome decoding on $\mathbf{b}^*$ to recover a low-weight "error vector" $(e_1, \ldots, e_N) \in \mathbb{Z}_q^N$. Output $\perp$ if this fails. (3) Output some index i such that $e_i \neq 0$.

Semantic security follows from [BF99] (under the DDH assumption). The reason is that given the public key values $(g^{\boldsymbol{\alpha}}, f = g^\beta)$ it is hard to distinguish the ciphertext values $g^{r\boldsymbol{\alpha}}, f^r$ for some $r \in \mathbb{Z}_q$ from a uniformly random and independent vector of $n+1$ group elements. Since this part does not involve leakage, we omit the formal proof and instead concentrate on the novel tracing part. The theorem below states the leakage resilient tracing security achieved by our scheme.

Theorem 14. *Assuming we choose $n \geq 3T + 2$ and $L \leq (n - 3T - 2)\log(q) - \omega(\log(\lambda))$ the above scheme satisfies L-leakage resilient tracing security under the DL assumption.*

Before giving the proof of Theorem 14, we define the "extended DL representation problem". Essentially, the attacker gets to fix some of the components of the representations. It also gets to see some representations *in full* and gets to leak on arbitrarily many others. The only representation that it can come up with should be in the convex span of the ones it saw in full. We formalize this as follows.

Definition 15 (Extended DL Representation Problem)**.** Let $\mathcal{G}$ be a prime order group generation algorithm. We define the (n, m, T, L)-*extended DL representation problem* for $n > m$ as a game between an attacker $\mathcal{A}$ and a challenger as follows:

- The challenger chooses $(\mathbb{G}, q, g) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ and random generators $g_1, \ldots, g_n, f \leftarrow \mathbb{G}$. It gives these values to the attacker $\mathcal{A}$.
- The attacker $\mathcal{A}$ may adaptively ask for two types of queries: **leakage** and **corrupt**. In both cases $\mathcal{A}$ specifies some *prefix* $x_1, \ldots, x_m \in \mathbb{Z}_q$ and the challenger chooses a uniformly random *suffix* $x_{m+1}, \ldots, x_n \in \mathbb{Z}_q$ subject to $\prod_{i=1}^n g_i^{x_i} = f$. Define $\mathbf{x} = (x_1, \ldots, x_m)$.
  - **Leakage Query:** $\mathcal{A}$ also specifies an L-bit leakage function (given as a circuit) $\mathsf{Leak} : \mathbb{Z}_q^n \to \{0,1\}^L$ and gets back $\mathsf{Leak}(\mathbf{x})$.
  - **Corrupt Query:** $\mathcal{A}$ gets the vector $\mathbf{x}$ in full.
- The attacker outputs a vector $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ and *wins* iff all of the following hold: (1) it made at most T corrupt queries throughout the game, (2) $\mathbf{x}^*$ is not in the span of the vectors given as responses to the corrupt queries, (3) $\mathbf{x}^*$ is a valid representation with $\prod_{i=1}^n g_i^{x_i^*} = f$.

We say that the (n, m, T, L)-*extended DL representation problem* is hard if for all PPT $\mathcal{A}$ there exists some negligible function $\mu(\cdot)$ such that $\Pr[\mathcal{A} \text{ wins }] \le \mu(\lambda)$ in the above game.

Lemma 16. *Under the* Discrete Log *assumption in* $\mathcal{G}$*, the* (n, m, T, L)*- extended DL representation problem is hard as long as* $n \ge T + m + 2$ *and* $L \le (n - m - T - 2)\log(q) - \omega(\log(\lambda))$.

Proof. The proof proceeds via a sequence of games. Let us assume w.l.o.g. that the attacker makes exactly T corruption queries.

Game 0. The original extended DL representation game. Let $\varepsilon(\lambda) = \Pr[\mathcal{A} \text{ wins in Game 0}]$.

Game 1. Game 1 proceeds exactly the same way as Game 0, but we redefine the winning condition. At the beginning of the game, the challenger also chooses random subspace $\mathcal{V} \subseteq \mathbb{Z}_q^n$ of dimension $n - T - 1$ which it keeps secret (and does not use anywhere in the game). We can think of this as choosing uniformly random vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1} \xleftarrow{\$} \mathbb{Z}_q^n$ at random ad defining $\mathcal{V} = \mathsf{span}(\mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1})$.[5] At the end of the game, let $\mathbf{x}_1, \ldots, \mathbf{x}_T$ be the T representation that the attacker sees in full as a result of the T corruption queries. Let $\mathbf{x}^*$ be the representation output by the attacker. In Game 1, the attacker only wins if, in addition to satisfying all of the required winning conditions from Game 0, the vector $\mathbf{x}^*$ satisfies

$$\mathbf{x}^* \notin \mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1}) \tag{1}$$

Since the attacker always loses in Games 0 and 1 if $\mathbf{x}^* \in \mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ we have:

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins in Game 1}] \;\ge\; & \Pr[\mathcal{A} \text{ wins in Game 0}] \\ & - \Pr[\mathbf{x}^* \in \mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1}) \setminus \mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_T)]\end{aligned}$$

Let us denote the event in the second line above by E. We claim that, over a random choice of $\mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1}$, the probability of E occuring is negligible. Wenever E occurs, there must be some coefficients $\mathbf{a} = (a_1, \ldots, a_T, a'_1, \ldots, a'_{n-T-1})$ such that $\mathbf{x}^* = \sum_{i=1}^T a_i \mathbf{x}_i + \sum_{j=1}^{n-T-1} a'_j \mathbf{v}_j$ and $a'_j \ne 0$ for some $j \in [n - T - 1]$. For any particular $\mathbf{a}$, the probability of this occurring is simply q^{-n} over the random choice of $\mathbf{v}_j$. Taking a union bound over all such coefficient vectors $\mathbf{a}$, of which there are $< q^{n-1}$, we get that $\Pr[E] \le 1/q$ is negligible. Therefore, there is some negligible μ such that $\Pr[\mathcal{A} \text{ wins in Game 1}] \ge \varepsilon(\lambda) - \mu(\lambda)$.

[5] The vectors are independent with overwhelming probability and so this is statistically close to choosing a random subspace of the given dimension.

Game 2. In this game, as in Game 1, the challenger also initially chooses the random subspace $\mathcal{V} \subseteq \mathbb{Z}_q^n$ of dimension $n-T-1$ and the winning condition is defined as in Game 1 with respect to the subspace $\mathcal{V}$. However, we now modify how the challenger answers *leakage queries*, by ensuring that all of the leaked-on representations $\mathbf{x}$ satisfy $\mathbf{x} \in \mathcal{V}$. It will be convenient to consider a (statically close) way of choosing the subspace $\mathcal{V}$ by selecting random vectors $\mathbf{w}_1, \ldots, \mathbf{w}_{T+1} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^n$ and setting $\mathcal{V} = \mathsf{ker}(\mathbf{w}_1, \ldots, \mathbf{w}_{(T+1)})$. Recall that in any leakage query the first m components $\mathbf{x}^{\mathsf{pre}} = (x_1, \ldots, x_m)$ of the representation are chosen by the attacker. In Game 2, the challenger chooses $\mathbf{x}^{\mathsf{suf}} = (x_{m+1}, \ldots, x_n)$ subject to $\mathbf{x} = (\mathbf{x}^{\mathsf{pre}}, \mathbf{x}^{\mathsf{suf}})$ satisfying:

$$\langle \mathbf{x}, \boldsymbol{\alpha} \rangle = \beta, \langle \mathbf{x}, \mathbf{w}_1 \rangle = 0, \ldots, \langle \mathbf{x}, \mathbf{w}_{T+1} \rangle = 0.$$

which is equivalent to choosing $\mathbf{x}^{\mathsf{suf}}$ uniformly at random from the affine space

$$\langle \mathbf{x}^{\mathsf{suf}}, \boldsymbol{\alpha}^{\mathsf{suf}} \rangle = \beta - \langle \mathbf{x}^{\mathsf{pre}}, \boldsymbol{\alpha}^{\mathsf{pre}} \rangle \quad , \quad \{ \ \langle \mathbf{x}^{\mathsf{suf}}, \mathbf{w}_j^{\mathsf{suf}} \rangle = -\langle \mathbf{x}^{\mathsf{pre}}, \mathbf{w}_j^{\mathsf{pre}} \rangle \ \}_{j=1}^{T+1}$$

where we use pre and suf denote the the first m and last $n-m$ components of the corresponding vector respectively.

In other words, in Game 2, each vector $\mathbf{x}^{\mathsf{suf}}$ is chosen from a random $(n-m-T-2)$-dimensional affine subspace of the $(n-m-1)$ dimensional affine space given by the left-most linear equation above. In Game 1, it is chosen from the full $n-m-1$ dimensional affine space. Therefore, by the subspace hiding lemma (Lemma 6), we can argue that the attacker cannot distinguish Game 1 from Game 2 even if it were given $\mathcal{V}$ at the end of the game (allowing it to evaluate the winning condition). Therefore, there is some negligible function μ such that

$$\Pr[\mathcal{A} \text{ wins in Game 2}] \geq \varepsilon(\lambda) - \mu(\lambda).$$

Contradiction. In Game 2, the attacker only wins if its representation $\mathbf{x}^*$ that it outputs at the end is a valid DL representation of f and satisfies $\mathbf{x}^* \notin \mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1})$ where the vectors $\mathbf{v}_j$ span $\mathcal{V}$. But all of the vectors that the attacker observes (through corrupt and leakage queries) are in the space $\mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{v}_1, \ldots, \mathbf{v}_{n-T-1})$ and therefore by the Boneh-Franklin representation lemma (Lemma 4) this can occur with at most negligible probability. Therefore, $\Pr[\mathcal{A}$ wins in Game 2$]$ is negligible and hence $\varepsilon(\lambda) = \Pr[\mathcal{A}$ wins in Game 0$]$ is also negligible.

□

Given this lemma, we can now easily prove that our traitor tracing scheme satisfies leakage resilient tracing security under the discrete log assumption.

Proof of Theorem 14. Recall that to win the game, the attacker $\mathcal{A}$ has to output a vector SK^* subject to the following conditions (1) It made at most T corrupt queries. (2) The predicate $\mathsf{GOOD}(\mathsf{PK}, \mathsf{SK}^*)$ holds. (3) The traced index i was *not* a corrupt query.

It's easy to see that the predicate $\mathsf{GOOD}(\mathsf{PK}, \mathsf{SK}^*)$ holds iff $\langle \boldsymbol{\alpha}, \mathsf{SK}^* \rangle = \beta$. Thus, the attacker $\mathcal{A}$ must output a DL representation of f, namely SK^*, where $\prod_{i \in [n]} g_i^{\mathsf{SK}_i^*} = f = g^\beta$ after seing T corruption queries and arbitrarily many leakage queries where the first $m = N - K = 2T$ components of each representation are fixed/known to the attacker (determined by the ECC). Given the above, we may apply Lemma 16 which shows that the key $\mathsf{SK}^* = (\mathbf{b}^* || \mathbf{x}^*)$ output by the attacker must be in the span of the T keys corresponding to the "traitors". Therefore we can write $\mathbf{b}^* = \mathbf{Be}$ where $\mathbf{e}$ has hamming weight at most T and is non-zero only for indices corresponding to the traitors. By the correctness and efficiency of syndrome decoding, the traitor tracing algorithm therefore correctly outputs the index of a traitor.

References

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [Hal09], pages 36–54.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *Sixth Theory of Cryptography Conference — TCC 2007*, volume 5444 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.

[BF99] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 338–353, London, UK, 1999. Springer-Verlag.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.

[BKKV10] Zvika Brakerski, Jonathan Katz, Yael Kalai, and Vinod Vaikuntanathan. Overcomeing the hole in the bucket: Public-key cryptography against resilient to continual memory leakage. In *FOCS* [IEE10], pages 501–510.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '94, pages 257–270, London, UK, 1994. Springer-Verlag.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[DF03] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption secure against adaptive chosen ciphertext attack. In *Proceedings of PKC*, 2003.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS* [IEE10], pages 511–520.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars Knudsen, editor, *Advances in Cryptology—EUROCRYPT 2002*, volume 2332 of *LNCS*. Springer-Verlag, 28 April–2 May 2002.

[DLWW11] Yevgeniy Dodis, Allison B. Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In *FOCS*, pages 688–697, 2011.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38:97–139, March 2008.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Symposium on Foundations of Computer Science*, pages 293–302, Philadelphia, PA, USA, October 25–28 2008. IEEE Computer Society.

[Hal09] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*. Springer-Verlag, 2009.

[IEE10] IEEE. *51th Symposium on Foundations of Computer Science*, Las Vegas, NV, USA, October 23–26 2010.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *LNCS*. Springer-Verlag, 2003.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology—ASIACRYPT 2009*, LNCS. Springer-Verlag, 2009. To Appear.

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, 2011.

[LRW11] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer-Verlag, February 19–21 2004.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Halevi [Hal09], pages 18–35.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.

[Oka92] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology—CRYPTO '92*, volume 740 of *LNCS*, pages 31–53. Springer-Verlag, 1993, 16–20 August 1992.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT 91*, volume 547 of *LNCS*, pages 522–526. Springer-Verlag, 8–11 April 1991.

[Wic11] Daniel Wichs. *Cryptographic Resilience to Continual Information Leakage*. PhD thesis, Department of Computer Science, NYU, 2011.