On the Insecurity of Parallel Repetition for Leakage Resilience

Allison Lewko*
University of Texas at Austin
alewko@cs.utexas.edu

Brent Waters‡
University of Texas at Austin
bwaters@cs.utexas.edu

***Abstract*—A fundamental question in leakage-resilient cryptography is: can leakage resilience always be amplified by parallel repetition? It is natural to expect that if we have a leakage-resilient primitive tolerating ℓ bits of leakage, we can take n copies of it to form a system tolerating $n\ell$ bits of leakage. In this paper, we show that this is not always true. We construct a public key encryption system which is secure when at most ℓ bits are leaked, but if we take n copies of the system and encrypt a share of the message under each using an n-out-of-n secret-sharing scheme, leaking $n\ell$ bits renders the system insecure. Our results hold either in composite order bilinear groups under a variant of the subgroup decision assumption *or* in prime order bilinear groups under the decisional linear assumption. We note that the n copies of our public key systems share a common reference parameter.**

I. Introduction

Traditional security definitions for cryptographic schemes address an adversary who only has black box access to the scheme, assuming that the secret key and other internal state remains completely hidden. In practice, the adversary might gain partial knowledge of the secret key or other internal state through various side-channel and memory attacks [29], [8], [5], [7], [34], [6], [30], [39], [22], [25]. Such attacks might leverage physical phenomena like computation time, power use, etc. to deduce partial information about the secret key or state. The cold-boot attack of [25] also demonstrates that an adversary can learn noisy information about the memory contents of a machine after the machine is powered down.

Devising specific countermeasures for each known kind of attack is an unsatisfying approach, since it may require frequent updates to cryptographic systems and always leaves them potentially vulnerable to new attacks which have not yet been anticipated. A relatively new alternative approach is to develop new cryptographic security definitions that model a wide class of attacks by allowing the adversary to specify a leakage function f and learn the output of f applied to the secret key or other portions of the internal state. Clearly, there must be limits placed on the leakage function, or the adversary could learn the entire secret key and the system would be insecure. Typically, we assume that the leakage function must be efficiently computable and that the size of its output is bounded by ℓ bits, where ℓ is a function of the security parameter and is less than the bit-length of the secret key.

This approach has yielded leakage-resilient constructions of many cryptographic primitives, including stream ciphers, signatures, symmetric key encryption, and public key encryption [33], [28], [19], [38], [15], [2], [3], [20], [17], [14], [11], [18], [4]. Given a construction that can tolerate ℓ bits of leakage, it is natural to ask: what if we expect even greater leakage? Recently, Alwen, Dodis, and Wichs [4] and Alwen, Dodis, Naor, Segev, Walfish, and Wichs [3] successfully employed parallel repetition to amplify leakage resilience for particular schemes and raised the fundamental question of whether leakage resilience can *always* be amplified by parallel repetition. More concretely, suppose we are given a public key encryption scheme which remains secure when ℓ bits are leaked (i.e. against an adversary who obtains ℓ bits of information about the secret key before seeing a challenge ciphertext). We can take n independent copies of the system corresponding to n public key, private key pairs. To encrypt, we now split the message into n shares, and encrypt the i^{th} share under the i^{th} public key. One may expect that this new system will remain secure up to $n\ell$ bits of leakage. Alwen et. al. [3] successfully apply this technique for specific schemes. As explained by [4], [3], it would seem quite difficult to prove this works in general, since a general reduction would need to simulate $n\ell$ bits of leakage for the parallel scheme using only ℓ bits of leakage from the original scheme.

We note that parallel repetition does hold generically if we weaken the definition of leakage resilience by restricting the leakage to a be subset of the bits representing the secret key, instead of allowing more complicated functions. This model was previously considered in [10], [16], [27]. In this setting, parallel repetition can be proven via the pigeonhole principle, since if $\leq n\ell$ bits are leaked from n keys, then there is some key for which at most ℓ bits are leaked, and security can then be proven via a reduction. (In fact, if $< n(\ell+1)$ bits are leaked from n keys, then there is some key for which $\leq \ell$ bits are leaked.)

Though posed in the context of public key encryption, parallel repetition naturally extends to other primitives, and would be a powerful general tool for amplifying leakage resilience while preserving reasonable levels of efficiency.

*Supported by a National Defense Science and Engineering Graduate Fellowship

‡Supported by NSF CNS-0716199, CNS-0915361, and CNS-0952692, Air Force Office of Scientific Research (AFO SR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO), Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), and the Alfred P. Sloan Foundation

We note that a more basic approach to improving resilience might be to artificially increase the security parameter, λ. The success of this approach will depend on how ℓ grows as a function of λ, and it also leads to an unacceptable loss in efficiency, since many common operations require time $O(\lambda^3)$ to compute.

Our Contribution: We show that there exist public key encryption schemes which are ℓ-leakage-resilient, but for which parallel repetition fails to yield an $n\ell$-leakage-resilient system for any $n > 1$. In fact, the parameters of our schemes can be chosen to rule out $\Omega(n\ell)$-leakage-resilience of the parallel schemes. Our results hold *either* under a variant of the subgroup decision assumption in composite order bilinear groups *or* under the decisional linear assumption in prime order bilinear groups. In both cases, our n parallel copies of the system share common setup parameters (i.e. are instantiated over the same group). Assuming a common group is natural in many settings, e.g. when using curves recommended by NIST [36].

Often, leakage resilience is established by employing mostly information-theoretic techniques, e.g. leveraging the fact that a function f with bounded output length cannot leak enough useful information about a key with sufficient min-entropy *even* if f is computationally unbounded. This approach is employed by [28], [33], [4], [3], for example. In the arguments of [3], [33], a computational assumption is used to argue that a valid ciphertext can be replaced by an invalid ciphertext. However, since the adversary does not receive the ciphertext until after the leakage, it is not clear that even a computationally unbounded leakage function would allow the adversary to distinguish the two cases.[1] For these kinds of arguments, it seems plausible that if ℓ bits of leakage is not enough to compromise the security of one key, then 2ℓ bits of leakage should not be enough to compromise the security of 2 keys. (We consider the case $n = 2$ here for concreteness and will later generalize.) However, security against computationally unbounded functions f is not strictly necessary. It is possible instead to have keys with less than ℓ bits of entropy, but where it is computationally hard to compress all of the information needed for decryption into only ℓ bits.

The main idea of our approach is to design a system where it is computationally hard to represent the needed information about a single key in ℓ bits, but where two keys can be efficiently compressed into 2ℓ bits. As a first attempt at creating keys with less than ℓ bits of entropy which are computationally hard to compress, one might try using pseudorandom generators. However, it is not clear how one might find suitable structure to allow compression of two keys using this approach. Instead, we use the structure of bilinear groups. We describe our approach in terms of composite order groups for ease of exposition. We suppose we have a bilinear group G of order $N = p_1 p_2 q$, which is a product of 3 distinct primes. This group has subgroups G_{p_1}, G_{p_2}, and G_q of orders p_1, p_2, and q respectively, and whenever elements of these different subgroups are paired together under the bilinear map, the result is the identity. In this sense, the subgroups are orthogonal to each other. In our system, keys and ciphertexts will take on one of two types: type 1 keys and ciphertexts will involve only elements of G_{p_1}, while type 2 keys and ciphertexts will involve only elements of G_{p_2}. Ciphertext elements are paired with key elements in order to decrypt. A key of type 1 and a key of type 2 can be efficiently compressed into a single key by multiplying them together in the group. This new key will now decrypt ciphertexts for *both* of the private keys, since the multiplied G_{p_2} elements will not affect the result of the pairing with the type 1 ciphertext, and the multiplied G_{p_1} elements will not affect the result of the pairing with the type 2 ciphertext. Assuming for simplicity that group elements are represented by approximately $\log(N)$ bits, we can set $\ell = \frac{1}{2}\log(N)$ so that 2ℓ bits is enough to leak a group element, but ℓ bits is not.

We now have a system that is attackable when parallelized, but it is not clear that it is leakage-resilient in the first place. To prove that a single key cannot be compromised by the leakage of ℓ bits, we cannot simply make an information-theoretic argument, since either $\log(p_1)$ or $\log(p_2)$ will be less than ℓ (hence there is min-entropy $< \ell$ in at least one type of secret key). To overcome this difficulty, we introduce an expansion technique which leverages the computational bound on the leakage function. More specifically, we use the G_q subgroup as what we call an "expansion space" to argue that the secret keys have sufficiently high pseudo-entropy (i.e. their distribution is computationally indistinguishable from a distribution with high min-entropy). Relying on a close variant of the subgroup decision assumption, we expand the keys into the G_q space, and argue that an attacker cannot distinguish between elements of G_{p_1} and $G_{p_1 q}$, where $G_{p_1 q}$ denotes the subgroup of order $p_1 q$ in G (and similarly cannot distinguish between elements of G_{p_2} and $G_{p_2 q}$). We note that the expansion space G_q is shared by both key types. In this computational step of the proof, it is crucial that the leakage function f is computationally bounded (since a computationally unbounded function could distinguish the subgroups). We next expand the ciphertexts into the G_q subgroup as well, and we are then able to finish our proof with an information-theoretic argument.

Prime Order Groups: We also provide a system following this framework in prime order bilinear groups, under the decisional linear assumption. Again, we expand keys into an expansion space to obtain sufficient entropy. As in our composite order system, we accomplish this expansion through a computational step.

[1] In fact, we conjecture that their schemes could be proven secure against a computationally unbounded leakage function under the stronger assumption that the computational problem remains hard against an adversary who is allowed unlimited preprocessing, given only the public parameters.

Extension to Signatures and Other Primitives: While we state and prove our result here in the context of public key encryption, our methodology is broader and applies to parallel repetition in other contexts. A natural application is to parallel repetition for signature systems, where one realizes parallel repetition by signing the same message under n different signing keys. In the full version [31], we sketch how our technique can be extended to provide a counterexample to parallel repetition for signature schemes, using the framework developed by Katz and Vaikuntanathan [28]. An alternate counterexample for signatures (due to Wichs [41]) is also discussed in Section VIII.

Related Work: Alwen et. al. [3] provide a counterexample to the security of parallel repetition in a more restricted setting where the public key setup is done by a single trusted authority holding a master secret key who additionally employs an n-out-of-n secret sharing scheme. In this system, leakage resilience cannot be amplified beyond the size of the master secret key. Such a setup occurs, for example, when an IBE scheme is employed. In contrast, our counterexample requires only that the n copies of the PKE scheme share the same underlying group.

Once our keys occupy the expansion space, the rest of our proof strategy is very similar to the machinery of hash proof systems (HPS), a primitive introduced by Cramer and Shoup [13] and used by Naor and Segev [33] to obtain leakage resilient PKE schemes.

More generally, various forms of leakage resilience have been studied in many previous works [40], [37], [28], [4], [11], [14], [18], [16], [27], [17], [33], [2], [3], [10], [15], [19], [26], [32], [38], [20]. Several models of leakage resilience have been proposed, differing primarily in the restrictions placed on the leakage functions and the internal state they are applied to. We discuss the key features and distinctions of these approaches below, organizing references according to their models.

Exposure-resilient cryptography [10], [16], [27] considered an adversary who could only learn a limited subset of the secret key bits, while [26] considered an adversary who could only learn the values on certain wires of the circuit implementing a computation. For models allowing arbitrary efficiently computable leakage functions f, one can bound the amount of leakage totally (bounding the total leakage over the lifetime of the system) or locally (bounding the amount of leakage per usage, e.g. per signature generated by a leakage-resilient signature scheme). A local bound is only reasonable if the internal state is continually updated, and the amount of leakage between updates is bounded. (If the secret key is unchanging, and one can leak ℓ bits of it many times, an attacker will eventually learn the entire secret key.) A total bound is employed e.g. by [2], [28], [33], [4], [3], while a local bound is employed e.g. by [20], [19], [38].

There is also a distinction between models which allow the leakage function to depend only on the secret key and models where the leakage function can depend on additional internal state. For schemes where the secret key is the only internal state, the secret key is a natural choice for the input to the leakage functions. For signatures, for example, the signer may maintain additional state. Micali and Reyzin [32] introduce the assumption that "only computation leaks information". Under this assumption, one may define the input to the leakage function to be the portion of the internal state which is accessed on that particular invocation. This approach is employed by [20] for stateful signatures with a local leakage bound, for example.

A general approach to tolerating leakage that is less than the length of the secret key is to guarantee that the secret key will have sufficient min-entropy conditioned on the leakage. The works [28], [2], [4], [3], [19], [38], for example, fall into this framework. Another possibility is considered by [15], who present schemes that can tolerate leakage of arbitrary length if the secret key remains sufficiently difficult to compute from the leakage (in this case, it is possible the secret key is information-theoretically determined by the leakage). One difficulty with this approach is that it may be hard to decide if a particular collection of possible leakage functions satisfy this criterion.

II. Organization

In Section III, we give the necessary background. In Section IV, we give our PKE system in composite order bilinear groups. In Section V, we prove it is leakage-resilient up to ℓ bits of leakage. In Section VI, we present an attack on the parallel version of our system with $n\ell$ bits of leakage. In Section VII, we give our PKE system in prime order bilinear groups. In Section VIII, we discuss variations on our system and attack and an alternative counterexample for signatures. In Section IX, we discuss possible extensions of our work.

III. Background

A. Leakage-resilient Public Key Encryption

We define IND-CPA leakage-resilient public key encryption schemes in terms of the following game between a challenger and an attacker. We let λ denote the security parameter, ℓ denote the leakage parameter, and (KeyGen, Encrypt, Decrypt) denote the algorithms of the PKE scheme. (Typically, ℓ is a function of λ.)

Key Generation: The challenger computes $(\mathrm{PK}, \mathrm{SK}) \leftarrow \mathrm{KeyGen}(\lambda, \ell)$ and gives PK to the attacker.

Leakage: The attacker chooses a function $f : \{0,1\}^* \rightarrow \{0,1\}^\ell$ that can be computed in polynomial time and receives $f(\mathrm{SK})$ from the challenger.

Challenge: The attacker chooses two messages, M_0 and M_1, and gives these to the challenger. The challenger chooses a uniformly random bit $\beta \in \{0,1\}$, and gives the attacker $\mathrm{CT} \rightarrow \mathrm{Encrypt}(M_\beta, PK)$.

Guess: The attacker outputs a bit $\beta' \in \{0,1\}$.

The attacker succeeds if $\beta = \beta'$. We define the advantage of an attacker $\mathcal{A}$ in this game to be $Adv_{\mathcal{A}}(\lambda, \ell) := |Pr[\beta = \beta'] - \frac{1}{2}|$.

Definition III.1. *A public key encryption system (KeyGen, Encrypt, Decrypt) is ℓ-leakage-resilient if all polynomial time attackers $\mathcal{A}$ have a negligible advantage in the above game.*

B. Parallel Repetition

We now formally state the parallel repetition question introduced by [4], [3]. We let (KeyGen, Encrypt, Decrypt) denote the algorithms of a PKE scheme. For each positive integer n, we define a new scheme, (KeyGen_n, Encrypt_n, Decrypt_n) as follows. KeyGen_n calls KeyGen n times to produce n pairs of public and secret keys: $(\text{PK}_1, \text{SK}_1), \ldots, (\text{PK}_n, \text{SK}_n)$. The public key is $\overline{\text{PK}} = (\text{PK}_1, \ldots, \text{PK}_n)$ and the secret key is $\overline{SK} = (\text{SK}_1, \ldots, \text{SK}_n)$. $\text{Encrypt}_n(M, \overline{PK})$ first splits the message M into n shares $M_1, \ldots, M_n$, using an n-out-of-n secret-sharing scheme. It then produces the ciphertext as: $\overline{\text{CT}} = (\text{Encrypt}(M_1, PK_1), \ldots, \text{Encrypt}(M_n, PK_n))$. $\text{Decrypt}_n(\overline{\text{CT}}, \overline{\text{SK}})$ calls Decrypt(Encrypt(M_i, PK_i), SK_i) for each i to produce M_i, and then reconstructs the secret M from its shares.

Question III.2. *If (KeyGen, Encrypt, Decrypt) is ℓ-leakage-resilient, then is (KeyGen_n, Encrypt_n, Decrypt_n) necessarily $n\ell$-leakage-resilient for each positive integer n?*

Below, we answer this question in the negative, even for $n\ell$ replaced by $\Omega(n\ell)$.

C. Bilinear Groups

We define bilinear groups by using a group generator $\mathcal{G}$, an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group G. For prime order bilinear groups, $\mathcal{G}$ outputs (p, G, G_T, e) where p is prime, G and G_T are cyclic groups of order p, and $e : G^2 \to G_T$ is a map such that:

1) (Bilinear) $\forall g, h \in G, a, b \in \mathbb{Z}_p, e(g^a, h^b) = e(g, h)^{ab}$
2) (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order p in G_T.

We will also use composite order bilinear groups (first introduced in [9]), where $\mathcal{G}$ outputs $(N = p_1 p_2 q, G, G_T, e)$ such that p_1, p_2, q are distinct primes, G and G_T are cyclic groups of order N, and $e : G^2 \to G_T$ is a bilinear map.

We further require that the group operations in G and G_T as well as the bilinear map e are computable in polynomial time with respect to λ. Also, we assume the group descriptions of G and G_T include generators of the respective cyclic groups. For composite order groups, we let G_{p_1}, G_{p_2}, and G_q denote the subgroups of order p_1, p_2 and q in G respectively. We note that when two elements coming from different prime order subgroups are paired together under e, the result is the identity element in G_T. In this sense, the subgroups G_{p_1}, G_{p_2}, and G_q are orthogonal to each other.

D. Complexity Assumptions

We will first present a version of our system in composite order bilinear groups and prove its security under the following assumption, which is a variant of the subgroup decision problem from [9].

In the assumption below, we let $G_{p_1 q}$ denote the subgroup of order $p_1 q$ in G.

Subgroup Decision Assumption: Given a group generator $\mathcal{G}$ for composite order bilinear groups, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 q, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_{p_1}, Y_1 \xleftarrow{R} G_{p_1},\ g_{p_2} \xleftarrow{R} G_{p_2}, Y_q \xleftarrow{R} G_q$$

$$P = (\mathbb{G}, g_{p_1}, g_{p_2}, Y_1 Y_q),$$

$$T_1 \xleftarrow{R} G_{p_1 q},\ T_2 \xleftarrow{R} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking the subgroup decision assumption to be:

$$AdvSD_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(P, T_1) = 1] - Pr[\mathcal{A}(P, T_2) = 1]\big|.$$

Definition III.3. *We say that $\mathcal{G}$ satisfies the subgroup decision assumption if $AdvSG_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

We also provide a translation of our system into prime order groups, where security is proven from the decisional linear assumption.

Decisional Linear Assumption: Given a group generator $\mathcal{G}$ for prime order bilinear groups, we define the following distribution:

$$\mathbb{G} = (p, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_0, g_1, g_2 \xleftarrow{R} G,\ r_1, r_2 \xleftarrow{R} \mathbb{Z}_p,$$

$$D = (\mathbb{G}, g_0, g_1, g_2, g_1^{r_1}, g_2^{r_2}),$$

$$T_1 = g_0^{r_1 + r_2},\ T_2 \xleftarrow{R} G.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking the decisional linear assumption to be:

$$AdvDLin_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition III.4. *We say that $\mathcal{G}$ satisfies the decisional linear assumption if $AdvDLin_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

E. Min-Entropy and Extractors

We will use strong extractors, which are established in [35]. We will also need the following standard lemma about min-entropy (e.g. [28]).

Lemma III.5. *Let X be a random variable with min-entropy h and let f be an arbitrary function with range $\{0,1\}^\ell$. For any $\tau \in [0, h-\ell]$, we define the set*

$$V_\tau := \{v \in \{0,1\}^\ell \mid H_\infty(X \mid v = f(X)) \leq h - \ell - \tau\}.$$

Then:

$$Pr[f(X) \in V_\tau] \leq 2^{-\tau}.$$

IV. Construction for Composite Order Groups

Our PKE system for composite order groups consists of four algorithms, (Setup, KeyGen, Encrypt, Decrypt). We assume the messages to be encrypted are elements of $\{0,1\}^m$. We build our system in a bilinear group whose order is a product of three primes, $N = p_1 p_2 q$, and prove its security for leakage parameter ℓ (sufficiently smaller than $\log(q)$) based on the subgroup decision assumption. We will state the precise conditions on ℓ that are needed in the following section. The main idea is that secret keys in the system will have *less than* ℓ bits of entropy, but since the leakage function is constrained to be computationally efficient, it cannot be used to break the system with only ℓ bits of leakage. More specifically, the secret key and ciphertext will be in subgroup G_{p_1} (or G_{p_2}) in the real system, but the subgroup decision assumption allows us to expand them to be in $G_{p_1 q}$ (or $G_{p_2 q}$), and now we have min-entropy greater than ℓ. However, if we make two copies of our system with a common setup phase, leakage of 2ℓ bits will allow us to leak a complete element of the group G, and we can obtain a "compressed" secret key that can decrypt for both copies of the system at once. The details of this will be given Section VI.

Setup$(\lambda) \rightarrow \mathrm{PP}$*:* The setup algorithm takes in the security parameter λ, and chooses distinct, sufficiently large primes p_1, p_2, q (it will choose q to be much larger p_1, p_2 - we will discuss this more precisely later). It chooses a bilinear group G of order $N = p_1 p_2 q$, along with generators g_{p_1}, g_{p_2} of the subgroups G_{p_1} and G_{p_2} respectively. (We let G_{p_1} denote the subgroup of G of order p_1, and G_{p_2} denote the subgroup of G of order p_2.) We let $S(G)$ denote the number of bits used to represent an element of G, and $S(G_T)$ denote the number of bits used to represent an element of G_T. It also chooses a uniformly random seed $D \in \{0,1\}^d$ for a (k,ϵ)-extractor $E : \{0,1\}^{S(G_T)} \times \{0,1\}^d \rightarrow \{0,1\}^m$, where m is the bit length of the messages. We assume the parameters k, ϵ, m are chosen so that ϵ is negligible with respect to the security parameter λ, and k is significantly smaller than $\log(q)$. It outputs the public parameters:

$$\mathrm{PP} := \{N, G, g_{p_1}, g_{p_2}, E, D\}.$$

KeyGen$(\mathrm{PP}) \rightarrow (\mathrm{PK}, \mathrm{SK})$*:* The key generation algorithm produces two types of keys: type 1 and type 2. It chooses a type $t \in \{1,2\}$ randomly, and then chooses a random $u \in G_{p_t}$. It sets $A = e(u, g_{p_t})$. The public key is $\mathrm{PK} = (t, A)$, and the secret key is $\mathrm{SK} = u$.

Encrypt$(\mathrm{PP}, \mathrm{PK}, M) \rightarrow \mathrm{CT}$*:* The encryption algorithm takes in the public parameters PP, a public key PK, and a message $M \in \{0,1\}^m$. It chooses a random $s \in \mathbb{Z}_N$ and computes $C_1 = g_{p_t}^s$ (where t is the type of the public key PK) and $C_2 = E(A^s, D) \oplus M$. It outputs the ciphertext $\mathrm{CT} = (C_1, C_2)$.

Decrypt$(\mathrm{CT}, \mathrm{SK}) \rightarrow M$*:* The decryption algorithm computes:

$$E(e(C_1, \mathrm{SK}), D) \oplus C_2 = M.$$

V. Leakage Resilience of our PKE system in Composite Order Groups

We will prove our system is ℓ-leakage-resilient for $\ell \leq \log(q) - k - \tau$ (where λ is our security parameter, k is the min-entropy required by our extractor E, and τ is a parameter chosen so that $2^{-\tau}$ is negligible in λ). We note that an attacker $\mathcal{A}$ who has a non-negligible advantage against the scheme must have a non-negligible advantage against type 1 or type 2 keys. Since we treat p_1 and p_2 symmetrically, we can assume without loss of generality that such an adversary has a non-negligible advantage against type 1 keys. Therefore, it suffices to prove security considering only type 1 keys (the proof for type 2 keys is exactly the same, with the roles of p_1 and p_2 interchanged). Our proof will proceed by a hybrid argument over the following sequence of games:

- Game_0: The real security game. Here the private key SK is a random group element in $\mathbb{G}_{p_1}$.
- Game_1: The private key SK is chosen as a random element in $\mathbb{G}_{p_1 q}$.
- Game_2: The challenge CT is generated by choosing a random $C_1 \in \mathbb{G}_{p_1}$ and setting $C_2 = E(e(\mathrm{SK}, C_1), D) \oplus M_\beta$. (SK is generated as in Game 1.)
- Game_3: The challenge CT is generated by choosing a random $C_1 \in \mathbb{G}_{p_1 q}$ and setting $C_2 = E(e(\mathrm{SK}, C_1), D) \oplus M_\beta$. (SK is generated as in Game 1.)
- Game_4: The challenge CT is generated by choosing a random $C_1 \in \mathbb{G}_{p_1 q}$ and setting C_2 as a uniformly random string.

To transition from Game_0 to Game_1, we employ our expansion technique: private keys are expanded into the G_q space, and an adversary cannot notice this without violating the subgroup decision assumption. The transition to Game_2 is made easily, since it is identically distributed to Game_1. To transition from Game_2 to Game_3, we again employ our expansion technique, this time expanding the ciphertext into the G_q space. We note that $e(\mathrm{SK}, C_1)$ now involves a term

from the G_q subgroup, and this term provides sufficient min-entropy to transition to Game_4 via an information-theoretic argument. The attacker has advantage 0 in Game_4, since the ciphertext is independent of the bit β. We prove these games are indistinguishable in the following lemmas.

Lemma V.1. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_0Adv_{\mathcal{A}} - Game_1Adv_{\mathcal{A}} = \delta$. Then we can build a polynomial time algorithm $\mathcal{B}$ with advantage δ in breaking the subgroup decision assumption.*

Proof: $\mathcal{B}$ receives $N, G, g_{p_1}, g_{p_2}, Y_1Y_q, T$. It chooses a uniformly random seed $D \in \{0,1\}^d$ and a (k,ϵ)-extractor $E : \{0,1\}^{S(G_T)} \times \{0,1\}^d \rightarrow \{0,1\}^m$. It sets the public parameters as

$$\text{PP} := \{N, G, g_{p_1}, g_{p_2}, E, D\}$$

and gives these to $\mathcal{A}$. Next, it sets $u = T$, i.e. $\text{SK} = T$, and $\text{PK} = (1, A = e(u, g_{p_1}))$. It gives PK to $\mathcal{A}$. When $\mathcal{A}$ chooses the leakage function, $\mathcal{B}$ computes $f(T)$ and sends this to $\mathcal{A}$. $\mathcal{A}$ then sends $\mathcal{B}$ two messages, M_0 and M_1. $\mathcal{B}$ chooses a random bit β and a random $s \in \mathbb{Z}_N$. It sets $C_1 = g_{p_1}^s$ and $C_2 = E(A^s, D) \oplus M_\beta$. It gives $\mathcal{A}$ the ciphertext $\text{CT} = (C_1, C_2)$.

If $T \in G_{p_1}$, then $\mathcal{B}$ has properly simulated Game_0. If $T \in G_{p_1q}$, then $\mathcal{B}$ has properly simulated Game_1. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve advantage δ in breaking the subgroup decision assumption. ∎

Lemma V.2. *For any algorithm $\mathcal{A}$, $Game_1Adv_{\mathcal{A}} = Game_2Adv_{\mathcal{A}}$.*

Proof: This simply follows from the fact that Game_1 and Game_2 are identically distributed (note that the additional component of G_q now included in SK will not effect the value of $e(\text{SK}, C_1)$, since $C_1 \in G_{p_1}$, and G_{p_1} and G_q are orthogonal under the pairing e). ∎

Lemma V.3. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_2Adv_{\mathcal{A}} - Game_3Adv_{\mathcal{A}} = \delta$. Then we can build a polynomial time algorithm $\mathcal{B}$ with advantage δ in breaking the subgroup decision assumption.*

Proof: $\mathcal{B}$ receives $N, G, g_{p_1}, g_{p_2}, Y_1Y_q, T$. It chooses a uniformly random seed $D \in \{0,1\}^d$ and a (k,ϵ)-extractor $E : \{0,1\}^{S(G_T)} \times \{0,1\}^d \rightarrow \{0,1\}^m$. It sets the public parameters as

$$\text{PP} := \{N, G, g_{p_1}, g_{p_2}, E, D\}$$

and gives these to $\mathcal{A}$. It sets $u = Y_1Y_q$, i.e. $\text{SK} = Y_1Y_q$, and $\text{PK} = (1, e(u, g_{p_1}))$, and gives PK to $\mathcal{A}$. When $\mathcal{A}$ chooses the leakage function, $\mathcal{B}$ computes $f(u)$ and sends this to $\mathcal{A}$. $\mathcal{A}$ then sends $\mathcal{B}$ two messages, M_0 and M_1. $\mathcal{B}$ chooses a random bit β and a random $s' \in \mathbb{Z}_N$. It sets $C_1 = T^{s'}$ and $C_2 = E(e(\text{SK}, C_1), D) \oplus M_\beta$. It gives $\text{CT} = (C_1, C_2)$ to $\mathcal{A}$.

If $T \in G_{p_1}$, then $\mathcal{B}$ has properly simulated Game_2. If $T \in G_{p_1q}$, then $\mathcal{B}$ has properly simulated Game_3. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve advantage δ in breaking the subgroup decision assumption. ∎

Lemma V.4. *For any polynomial time algorithm $\mathcal{A}$, $Game_3Adv_{\mathcal{A}} - Game_4Adv_{\mathcal{A}}$ is negligible.*

Proof: We prove that with all but negligible probability, the distributions of Game_3 and Game_4 are negligibly close in $\mathcal{A}$'s view. In Game_3, the value u can be written as $u = u_1u_q$, where $u_1 \in G_{p_1}$ and $u_q \in G_q$. By the orthogonality of G_{p_1} and G_q, the public key element $e(u, g_{p_1})$ only contains information about u_1, and reveals no information about u_q, which is distributed as a random element of G_q. Thus, even conditioned on PK and C_1, the value $e(\text{SK}, C_1)$ has min-entropy $\log(q)$ from the attacker's perspective.

Since the attacker also receives $f(u)$ for the leakage function f with range $\{0,1\}^\ell$, we invoke Lemma III.5 to assert that with probability $\geq 1 - 2^{-\tau}$, $e(\text{SK}, C_1)$ will still have min-entropy $\geq k$ as long as $\log(q) - \ell - \tau \geq k$, i.e. $\ell \leq \log(q) - k - \tau$. Therefore, when $\ell \leq \log(q) - k - \tau$, $E(e(\text{SK}, C_1), D)$ has statistical distance at most ϵ from a uniformly random string of length m with all but negligible probability. Since ϵ and $2^{-\tau}$ are chosen to be negligible in the security parameter λ, we have that $Game_3Adv_{\mathcal{A}} - Game_4Adv_{\mathcal{A}}$ is negligible as well. (We note here that it is crucial for the adversary to choose the leakage function *before* seeing the ciphertext. Otherwise, the leakage function f could change when we replace $E(e(\text{SK}, C_1), D)$ with a random string of length m, and we would not be able to argue the indistinguishability of the games.) ∎

This completes our proof of the following theorem:

Theorem V.5. *For $\ell \leq \log(q) - k - \tau$, our PKE system is ℓ-leakage-resilient.*

VI. Attack on the Parallel System for $n > 1$ with Leakage $n\ell$

We first describe our attack for $n = 2$ when 2ℓ bits of leakage is sufficient to return a whole group element. We then generalize the attack to higher values of n. We recall that $S(G)$ denotes the number of bits representing an element of G, and we discuss the value of $S(G)$ for some particular groups in the full version [31].

A. Attack for $n = 2$ when $2\ell \geq S(G)$

We define the system (Setup, KeyGen_2, Encrypt_2, Decrypt_2) as before, except we additionally assume that the two copies of the system are generated by a common setup phase, so the public parameters are shared. In other words, Setup is called only once, and outputs a single set of public parameters PP. KeyGen_2 then calls KeyGen twice on the *same* public parameters PP to generate two keys, $(\text{PK}_1, \text{SK}_1)$ and $(\text{PK}_2, \text{SK}_2)$. Encrypt_2 splits the message M into two shares M_1 and M_2. It encrypts the first share

by calling Encrypt(PK_1, M_1) and encrypts the second share by calling Encrypt(PK_2, M_2). Decrypt$_2$ calls Decrypt on the first encrypted share with SK_1 to obtain the share M_1, and calls Decrypt on the second encrypted share with SK_2 to obtain M_2. It then reconstructs M from its shares.

The attacker receives the public parameters, PP, and two public keys, PK_1 and PK_2. If the two public keys are of the same type, the attacker aborts (this occurs with probability $\frac{1}{2}$) and guesses β' randomly. Otherwise, we assume PK_1 is of type 1 and PK_2 is of type 2. The attacker chooses the leakage function

$$f(\mathrm{SK}_1, \mathrm{SK}_2) = \mathrm{SK}_1 \cdot \mathrm{SK}_2,$$

which is permitted as long as $2\ell \geq S(G)$ (recall that $S(G)$ denotes the number of bits used to represent an element of the group G). Assuming this holds, the attacker receives the value $\mathrm{SK}_1 \cdot \mathrm{SK}_2$, and can use this to decrypt *both* ciphertexts. For example, to decrypt the ciphertext (C_1, C_2) for PK_1, the attacker computes:

$$E(e(C_1, \mathrm{SK}_1 \cdot \mathrm{SK}_2), D) \oplus C_2 = E(e(C_1, \mathrm{SK}_1), D) \oplus C_2,$$

since $C_1 \in G_{p_1}$ and $\mathrm{SK}_2 \in G_{p_2}$. This yields the first message share, and the second message share is obtained similarly, since the first ciphertext value for PK_2 will be orthogonal to SK_1. Hence, the attacker can reconstruct the encrypted message and succeed with probability 1 when the keys are of different types. This gives the attacker an overall advantage of $\frac{1}{4}$.

When $S(G) \leq 2(\log(q) - k - \tau)$, we can choose a single value of ℓ such that $\ell \leq \log(q) - k - \tau$ and $2\ell \geq S(G)$. This will yield a PKE system (Setup, KeyGen, Encrypt, Decrypt) that is ℓ-leakage-resilient, but (Setup, KeyGen$_2$, Encrypt$_2$, Decrypt$_2$) is *not* 2ℓ-leakage-resilient. We note that we will choose q to be significantly larger than p_1 and p_2 in order to satisfy $S(G) \leq 2(\log(q) - k - \tau)$ for groups G of order $N = p_1 p_2 q$.

B. General Values of n

We now consider attacking the more general case, where we have n public and secret key pairs, with a common setup phase. On average, the attacker can expect that close to half of the n keys will be of type 1 and the others will be of type 2. More specifically, a Chernoff bound implies that for any positive constant c, there will be at least $\frac{n}{2} - c\sqrt{n}$ keys of each type with probability $\geq 1 - 2e^{-2c^2}$.

When this distribution of keys occurs, the attacker can organize the keys into at least $\frac{n}{2} - c\sqrt{n}$ pairs of keys and at most $2c\sqrt{n}$ individual keys, where each pair of keys contains one key of type 1 and one key of type 2. The attacker then defines the leakage function so that it reveals the product of each pair of secret keys and all of the remaining unpaired secret keys. This requires $(\frac{n}{2} - c\sqrt{n})S(G) + 2c\sqrt{n}S(G) = S(G)(\frac{n}{2} + c\sqrt{n})$ bits of leakage. We note that the attacker can now decrypt messages encrypted under *any* of the n public keys.

As long as

$$n\ell \geq S(G)\left(\frac{n}{2} + c\sqrt{n}\right) \Leftrightarrow \ell \geq S(G)\left(\frac{1}{2} + \frac{c}{\sqrt{n}}\right)$$

holds, this is a valid attack with $n\ell$ bits of leakage that succeeds with probability $\geq 1 - 2e^{-2c^2}$.

We recall that $\ell \leq \log(q) - k - \tau$ is required for our proof that a single copy of our system is ℓ-leakage-resilient. For this condition and the attack condition $\ell \geq S(G)(\frac{1}{2} + \frac{c}{\sqrt{n}})$ to hold simultaneously for some value of ℓ, it suffices to have a group G such that:

$$S(G) \leq \frac{2}{1 + \frac{2c}{\sqrt{n}}}(\log(q) - k - \tau).$$

Improving the Compression Factor: We have thus far described a system with two key types and a corresponding attack where two keys of different types can essentially be "compressed" into one key of the same size. This will work well for leakage $n\ell$ when $S(G) < 2\log(N)$ and q is chosen to be sufficiently large with respect to p_1 and p_2. To adapt our attack to work with leakage $\gamma\ell n$ for any fixed constant $\gamma > 0$ or for groups where $S(G)$ may be larger than $2\log(N)$, we can improve the compression factor by allowing $T > 2$ key types, where T keys of distinct types can be compressed into one key (assuming $n \geq T$). We discuss this more in Section VIII.

VII. Realizing our Construction in Prime Order Groups

We now describe a realization of our system in prime order groups, and prove it is ℓ-leakage-resilient from the decisional linear assumption. The main idea of our construction is unchanged: we begin with keys that have less than ℓ bits of entropy, and we use the decisional linear assumption to expand them into a larger space, where they will have entropy much greater than ℓ.

We start with a bilinear group G of prime order p, generated by g. Previously, we made use of the orthogonal subgroups G_{p_1}, G_{p_2}, and G_q in our bilinear group of order N. To make suitable analogs of these subgroups using the prime order group G, we will construct "orthogonal" subgroups of G^j for some (relatively small) positive integer j. For a vector $\vec{x} = (x_1, \ldots, x_j) \in \mathbb{Z}_p^j$, we write $g^{\vec{x}}$ to denote the j-tuple of elements $(g^{x_1}, \ldots, g^{x_j})$ in G^j. We let

$$< g^{\vec{x}}, g^{\vec{y}} > := \{(g^{ax_1 + by_1}, \ldots, g^{ax_j + by_j}) \mid a, b \in \mathbb{Z}_j\}$$

denote the subgroup of G^j generated by $g^{\vec{x}}$ and $g^{\vec{y}}$. When we write an expression of the form $g^{\vec{x}} g^{\vec{y}}$, we mean componentwise multiplication, i.e. $g^{\vec{x}} g^{\vec{y}} = g^{\vec{x}+\vec{y}}$.

We define the map $e_j : G^j \times G^j \to G_T$ by:

$$e_j(g^{\vec{x}}, g^{\vec{y}}) = \prod_{i=1}^{j} e(g^{x_i}, g^{y_i}) = e(g, g)^{\vec{x} \cdot \vec{y}},$$

where e is the bilinear map from $G \times G$ into G_T. We note that $e_j(g^{\vec{x}}, g^{\vec{y}})$ is the identity element in G_T when $\vec{x}$ and $\vec{y}$ are orthogonal as vectors over $\mathbb{Z}_p$. The orthogonal subgroups G_{p_1} and G_{p_2} in composite order groups can now be replaced with subgroups $< g^{\vec{x}}, g^{\vec{y}} >$ and $< g^{\vec{v}}, g^{\vec{z}} >$ in G^j where $\vec{v}, \vec{z}$ are each orthogonal to both $\vec{x}$ and $\vec{y}$. G_q will be replaced by the subgroup comprised of elements $g^{\vec{w}} \in G^j$ where $\vec{w}$ is orthogonal to all of $\vec{x}, \vec{y}, \vec{v}, \vec{z}$. The technique of using orthogonal vectors over $\mathbb{Z}_p$ to simulate orthogonal subgroups in a prime order group was also employed by Freeman [21], though his results do not encompass our construction.

A. Construction

Our system in a prime order group G can now be described as follows:

Setup(λ*)* $\rightarrow$ PP*:* The setup algorithm takes in the security parameter, and chooses a sufficiently large prime p. It chooses a bilinear group G of order p, along with a generator g and a suitable integer $j > 5$. We let $S(G)$ denote the number of bits used to represent an element of G, and $S(G_T)$ denote the number of bits used to represent an element of G_T. It also chooses a uniformly random seed $D \in \{0,1\}^d$ for a (k, ϵ)-extractor $E : \{0,1\}^{S(G_T)} \times \{0,1\}^d \rightarrow \{0,1\}^m$, where m is the bit length of the messages. We assume the parameters k, ϵ, m are chosen so that ϵ is negligible with respect to the security parameter λ, and k is $\leq \frac{1}{4} \log p$. It next chooses uniformly random vectors $\vec{x}, \vec{y} \in \mathbb{Z}_p^j$. Vectors $\vec{v}, \vec{z}$ are then chosen uniformly at random from the space of vectors which are orthogonal to both $\vec{x}$ and $\vec{y}$.

It outputs the public parameters:

$$\text{PP} := \{G, p, g, j, g^{\vec{x}}, g^{\vec{y}}, g^{\vec{v}}, g^{\vec{z}}, E, D\}.$$

*KeyGen(*PP*)* $\rightarrow$ (PK, SK)*:* The key generation algorithm produces two types of keys: type 1 and type 2. It chooses a type $t \in \{1, 2\}$ randomly, and then chooses random values $u_1, u_2 \in \mathbb{Z}_p$. If the type $t = 1$, it sets

$$A_1 = e_j(g^{u_1\vec{x}+u_2\vec{y}}, g^{\vec{x}}),\ A_2 = e_j(g^{u_1\vec{x}+u_2\vec{y}}, g^{\vec{y}}).$$

The public key is $\text{PK} = (1, A_1, A_2)$, and the secret key is $\text{SK} = g^{(u_1\vec{x}+u_2\vec{y})}$. If the type $t = 2$, it sets

$$A_1 = e_j(g^{u_1\vec{v}+u_2\vec{z}}, g^{\vec{v}}),\ A_2 = e_j(g^{u_1\vec{v}+u_2\vec{z}}, g^{\vec{z}}).$$

The public key is $\text{PK} = (2, A_1, A_2)$, and the secret key is $\text{SK} = g^{(u_1\vec{v}+u_2\vec{z})}$.

*Encrypt(*PP, PK, M*)* $\rightarrow$ CT*:* The encryption algorithm takes in the public parameters PP, a public key PK, and a message $M \in \{0,1\}^m$. It chooses random values $s_1, s_2 \in \mathbb{Z}_p$. If the type $t = 1$, it computes

$$C_1 = g^{s_1\vec{x}+s_2\vec{y}},\ C_2 = E(A_1^{s_1} \cdot A_2^{s_2}, D) \oplus M.$$

If the type $t = 2$, it computes

$$C_1 = g^{s_1\vec{v}+s_2\vec{z}},\ C_2 = E(A_1^{s_1} \cdot A_2^{s_2}, D) \oplus M.$$

*Decrypt(*CT, SK*)* $\rightarrow$ M*:* The decryption algorithm computes:

$$E(e_j(C_1, \text{SK}), D) \oplus C_2 = M.$$

This system is leakage-resilient up to $\ell = (j-5)\log p$ bits of leakage:

Theorem VII.1. *For $\ell \leq (j-5)\log(p)$, our PKE system in prime order groups is ℓ-leakage-resilient under the decisional linear assumption.*

The proof of this theorem is contained in the full version of this paper [31], and employs the same strategy as our proof for the composite order case. The attack on the parallel system for $n\ell$ bits of leakage is analogous to the attack for composite order groups. For $n = 2$ (for example), the attack is applicable whenever $2\ell \geq jS(G)$.

B. Correctness

Correctness of the algorithm is verified by observing (e.g. for type 1 keys):

$$\begin{aligned} e_j(C_1, \text{SK}) &= e_j(g^{s_1\vec{x}+s_2\vec{y}}, g^{u_1\vec{x}+u_2\vec{y}}) \\ &= e(g,g)^{(s_1\vec{x}+s_2\vec{y})\cdot(u_1\vec{x}+u_2\vec{y})} \\ &= e(g,g)^{s_1((u_1\vec{x}+u_2\vec{y})\cdot\vec{x})+s_2((u_1\vec{x}+u_2\vec{y})\cdot\vec{y})} \\ &= A_1^{s_1}A_2^{s_2}. \end{aligned}$$

VIII. Discussion

Improving the Compression Factor: We could improve the compression factor for both prime and composite order groups by constructing our systems with T key types, allowing T keys of distinct types to be compressed at once. In composite order groups, this would be done by choosing a group order $N = p_1 p_2 \cdots p_T q$, where q is much larger than each of the p_i's. A key of type i would be in the subgroup G_{p_i}. The product of T keys, one of each type, would yield a single element that could be substituted for *any* of the T input keys in the decryption algorithm.

This system would still be ℓ-leakage-resilient for $\ell \leq \log(q) - k - \tau$, under the analog of the subgroup decision assumption for groups of order $N = p_1 \cdots p_T q$. For the parallel version of the system with $n \geq T$ keys, the attacker can expect roughly $\frac{n}{T}$ keys of each type. More precisely, for any constant $c > 0$, there will be at least $\frac{n}{T} - \frac{c}{T}\sqrt{n}$ keys of each type with probability $\geq 1 - Te^{-\frac{c^2}{2T}}$ (by a Chernoff bound). When this occurred, the attacker could group the n keys into $\geq \frac{n}{T} - \frac{c}{T}\sqrt{n}$ sets of T keys of distinct types, with at most $c\sqrt{n}$ individual keys remaining. With leakage at least $S(G)(\frac{n}{T} + c(1-\frac{1}{T})\sqrt{n})$, the attacker could learn the products of all the groups of keys and all the remaining individual keys, and hence decrypt all the shares of the ciphertext.

For any fixed constant $\gamma > 0$, we can then mount our attack on the parallel system with leakage $\gamma\ell n$ as long as:

$$S(G) \leq \frac{\gamma T}{1 + \frac{c(T-1)}{\sqrt{n}}} \left(\log(q) - k - \tau\right).$$

For prime order groups (of order p), having T key types instead of T would simply require setting $j > 2T + 1$ to create enough space to simulate $T+1$ orthogonal subgroups in $\mathbb{Z}_p^j$ with our method. Our system is leakage resilient up to ℓ bits of leakage for $\ell = (j - (2T + 1)) \log p$, for each value of T. We note the underlying security assumption (decision linear) is independent of the number of simulated subgroups and the value of j. Our attack then requires $jS(G)\left(\frac{n}{T} + c\left(1 - \frac{1}{T}\right)\sqrt{n}\right)$ bits of leakage, and this will be $\leq \gamma n\ell$ for $\ell = (j - (2T + 1)) \log p$ as long as:

$$S(G) \leq \frac{\gamma(j - (2T + 1))T}{j\left(1 + \frac{c(T-1)}{\sqrt{n}}\right)} \log(p).$$

For any fixed $\gamma > 0$, we can choose T and j large enough to meet this requirement. Hence, our counterexample shows that leakage $\Omega(n\ell)$ is not always achieved by parallel repetition.

An Alternate Counterexample for Signatures: One can also obtain a counterexample to parallel repetition for signatures from any multi-signature scheme secure against "rogue-key attacks", as observed by Wichs [41]. One defines an ordinary signature scheme by using the same key generation algorithm, and having a signer simply sign each message under a set of size 1, containing only its own public key. The verification algorithm will accept as valid any signature under a set containing the correct public key which verifies under the multi-signature scheme verification algorithm (note that this accepts signatures which would never be generated by a honest signer). This scheme is secure up to some ℓ bits of leakage (at least logarithmic). The parallel system (with n copies) can then be broken for any leakage exceeding the size of a single signature under the multi-signature scheme (which is independent of n). The attacker simply asks for leakage which is a multi-signature of some message under the set containing all n public keys: this will then verify as a valid signature for each of the n signers individually. This allows the attacker to forge on only one message. In contrast, a counterexample for signatures obtained using our techniques will allow the attacker to forge as many messages as desired.

IX. Future Directions

We now discuss a few approaches for avoiding or extending our counterexample.

Removing the Common Reference String: The assumption of a common setup for the n copies of a parallel system is natural, since it is typical to create several public keys from one group in practical systems. For example, NIST recommends using certain elliptic curves [36]. However, it would be interesting to construct a counterexample to parallel repetition that does not rely on a common setup.

A Black-box Separation: Alwen et. al. [3] suggest the potential direction of showing a black-box separation for parallel repetition. Such a result would rule out the possibility of a general reduction using an attacker who can break the parallel scheme with $n\ell$ bits of leakage to break the original scheme with ℓ bits of leakage. This would be incomparable to our result, since a black-box separation does not imply the existence of a counterexample, and our result relies on unproven (though commonly used) assumptions for security.

Alternate Assumptions: We rely on either a variant of the subgroup decision assumption in composite order bilinear groups *or* the decisional linear assumption in prime order bilinear groups. We also suspect that our results could be adapted to provide a counterexample under lattice-based assumptions, given that many results obtained using bilinear groups have also been instantiated with lattices (e.g. [24], [23], [1], [12]). Obtaining additional counterexamples under a variety of assumptions would provide stronger evidence for the insecurity of parallel repetition as a generic tool.

A Looser Bound: One might ask if parallel repetition holds for some sublinear bound on the leakage as a function of n. In other words, can we take n copies of an ℓ-leakage-resilient system and always build an $f(n, \ell)$-leakage-resilient system for some sufficiently growing function $f(n, \ell) = o(n\ell)$?

Alternative Leakage Models: One can also ask if parallel repetition holds for other interesting leakage models. For instance, we might strengthen the definition of leakage resilience by allowing the leakage function f to be computationally unbounded. Our counterexample no longer applies in this case. If we instead weaken the definition of leakage resilience by restricting the leakage to being a subset of the bits representing the secret key, we recall that parallel repetition *does* hold.

References

[1] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[2] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[3] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[4] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

[5] E. Biham, Y. Carmeli, and A. Shamir. Bug attacks. In *CRYPTO*, pages 221–240, 2008.

[6] E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In *CRYPTO*, pages 513–525, 1997.

[7] D. Boneh and D. Brumley. Remote timing attacks are practical. In *Computer Networks*, volume 48, pages 701–716, 2005.

[8] D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT*, pages 37–51, 1997.

[9] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–341, 2005.

[10] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.

[11] D. Cash, Y. Z. Ding, Y. Dodis, W. Lee, R. J. Lipton, and S. Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In *TCC*, pages 479–498, 2007.

[12] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

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

[14] D. Di Crescenzo, R. J. Lipton, and S. Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.

[15] Y. Dodis, Y. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

[16] Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In *EUROCRYPT*, pages 301–324, 2001.

[17] S. Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.

[18] S. Dziembowski and K. Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.

[19] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

[20] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.

[21] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[22] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *CHES*, number Generators, pages 251–261, 2001.

[23] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple bgn-type cryptosystem from lwe. In *EUROCRYPT*, pages 506–522, 2010.

[24] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[25] A. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J. Applebaum, and E. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.

[26] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

[27] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *FOCS*, pages 92–101, 2003.

[28] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[29] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.

[30] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

[31] Allison Lewko and Brent Waters. On the insecurity of parallel repetition for leakage resilience. Cryptology ePrint Archive, Report 2010/404, 2010.

[32] S. Micali and L. Reyzin. Physically observable cryptography. In *TCC*, pages 278–296, 2004.

[33] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

[34] P. Q. Nguyen and I. Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. volume 15, pages 151–176, 2002.

[35] N. Nisan and D. Zuckerman. Randomness is linear in space. In *J. Comput. Syst. Sci.*, volume 52, pages 43–52, 1996.

[36] National Institute of Standards and Technology. Digital signature standard (dss), June 2009. http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf.

[37] C. Petit, F.X. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *ASIACCS*, pages 56–65, 2008.

[38] K. Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.

[39] J. Quisquater and D. Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.

[40] F.X. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In *EUROCRYPT*, pages 443–461, 2009.

[41] D. Wichs. personal communication, 2010.