Implementation and Evaluation of a Leakage-Resilient ElGamal Key Encapsulation Mechanism

David Galindo[1,2], Johann Großschädl[3], Zhe Liu[3], Praveen Kumar Vadnala[3], and Srinivas Vivek[3]

[1] CNRS, Loria, France
[2] SCYTL Secure Electronic Voting, Spain
[3] University of Luxembourg, Luxembourg

Abstract. Leakage-resilient cryptography aims to extend the rigorous guarantees achieved through the provable security paradigm to physical implementations. The constructions and mechanisms designed on basis of this new approach inevitably suffer from an Achilles heel: a bounded leakage assumption is needed. Currently, a huge gap exists between the theory of such designs and their implementation to confirm the leakage resilience in practice. The present work tries to narrow this gap for the leakage-resilient bilinear ElGamal key encapsulation mechanism (BEG-KEM) proposed by Kiltz and Pietrzak in 2010. Our first contribution is a variant of the bounded-leakage and the only-computation-leaks model that is closer to practice. We weaken the restriction on the image size of the leakage functions in these models and only insist that the inputs to the leakage functions have sufficient min-entropy left, in spite of the leakage, with no limitation on the quantity of this leakage. We provide a novel security reduction for BEG-KEM in this relaxed leakage model using the generic bilinear group axiom. Secondly, we show that a naive implementation of the exponentiation in BEG-KEM makes it impossible to meet the leakage bound. Instead of trying to find an exponentiation algorithm that meets the leakage axiom (which is a non-trivial problem in practice), we propose an advanced scheme, BEG-KEM+, that avoids exponentiation by a secret value, but rather uses an encoding into the base group due to Fouque and Tibouchi. Thirdly, we present a software implementation of BEG-KEM+ based on the Miracl library and provide detailed experimental results. We also assess its (theoretical) resistance against power analysis attacks from a practical perspective, taking into account the state-of-the-art in side-channel cryptanalysis.

Keywords: Secure implementation, side-channel cryptanalysis, leakage-resilient cryptography, security proof, public-key encryption, pairings

1 Introduction

How to secure cryptographic algorithms embedded in devices that can eventually "fall in the hands" of an adversary? Answering this question is probably

the holy grail in cryptography nowadays. Two paths are taken to explore the possible solutions, a destructive one and a constructive one. In the first path, we find the rich contributions of the practice and theory of side-channel attacks. In the second path, we find the no less precious body of countermeasures against the attacks unveiled in the first path. Lately, a novel approach called leakage-resilient cryptography is being studied, which aims at extending the guarantees delivered by the provable security paradigm to the physical world. Despite the clever discoveries and constructions provided by this new approach, it persistently presents an Achilles heel: a bounded leakage assumption is needed. Ensuring this is unfortunately a challenging endeavor on its own and, admittedly, the leakage-resilient cryptography body of work has not significantly helped to argue why this could be a reasonable assumption.

In this work, we consider the *only computation leaks information* (OCL) leakage model by Micali and Reyzin [23]. In this model only actual computations are supposed to leak sensitive information. This captures the usual situation in side-channel attacks, where leakage data only depends on the current state of the target device and some independent randomness [33]. The internal data of the device is divided into two parts, an *active* and a *passive* part, the active part being the input data used in the current computation. Therefore, at a given time frame, only the active data is leaking. The main non-invasive attacks against embedded devices, like the attacks based on power consumption [21], electromagnetic radiations [15] or running-time [20] measurements, belong to this category.

It is currently agreed upon that, not only the OCL model, but also the bounded retrieval/memory leakage models [2, 3] or the auxiliary input model [9], rely on a strange combination of both strong and weak assumptions. On the one side, the information leakage is supposed to be bounded in a somewhat artificial manner; on the other side, the leakage considered is overly general, for instance it might come from any polynomial time function. However, these assumptions are actually far from the reality that practitioners experience in their daily work in a side-channel analysis lab.

Several contemporary works [33, 28, 5] have put forward ways to redefine the above models and bring them closer to practice, for symmetric cryptography primitives. This comes at the cost of algorithmic-level specialization, providing models that are indeed more realistic, but which apply to a more restrained class of primitives (i.e. pseudorandom generators, block ciphers).

We aim at contributing to the challenge of bringing leakage-resilient cryptography closer to the practice. In this work, we do so by analyzing, modifying, implementing and evaluating a previous leakage-resilient key encapsulation mechanism proposed by Kiltz and Pietrzak [18]. This is one of the very few schemes admitting continual leakage (maybe the only one?) that one could dare to implement in an embedded processor, for instance in a smartphone. It is a pairing-based stateful variant of the ElGamal encryption scheme (called BEG-KEM), where the secret key is an element of the pairing base group (essentially a point in the group of points of an elliptic curve). The secret key is divided into

two shares, which are re-shared at each new decryption call by using multiplicative blinding. To decrypt, one takes the first half of the secret key, refreshes it, and uses it as the input to a pairing calculation. In the second step, the second half of the secret key is updated with the blinding used for refreshing; it is then used as the input to a new pairing calculation; and finally the two pairing values are multiplied to obtain a decapsulated symmetric key (for the details see Section 2).

The result proven in [18], which holds under a variant of the generic group model tailored to pairing groups uses a bounded leakage assumption. Roughly speaking, it is required that the *data leaked* against side-channel attacks that satisfy the OCL axiom, shall be significantly smaller than κ for a single measurement, where κ is the security parameter (e.g. $\kappa = 128$). These leakages are modeled as an oracle that answers values $f(\cdot)$ for adaptively chosen arbitrary (but efficiently computable) functions f on input the secret data being used in the calculation. This kind of requirement, that may look reasonable for a theoretician used to study cryptographic primitives in the so-called *black-box model* might seem completely unrealistic to the practitioner. An an example, let us recall the figure gathered in [33], where it is pointed out that the leaking of a block cipher recently reported in [25], consisted of 200000 traces leading to more than 1.5 Gigabits of data storage.

We start our investigation by proposing and testing a relaxation on the requirement of 'bounded leakage size' in the OCL model. We weaken the restriction on the image size of the leakage functions in these models to asking that the random variables used to refresh the secret key shall have enough min-entropy left given the leakage, with no limitation on the 'size' of this leakage. This is an altogether more reasonable leakage bound assumption, which could eventually be met by clever implementations (in fact we provide an implementation candidate). We give a new security reduction using the generic bilinear group axiom for BEG-KEM in this relaxed leakage model, which turns out to be tighter than the original reduction in [18] in the OCL model.

Secondly, we observe that the blinding mechanism originally proposed is susceptible to invalidate the leakage bound assumption. This is because to perform blinding, one computes an exponentiation G^{r_i} for a random integer r_i, which if implemented in a naive way, can almost completely leak r_i, even with a simple power analysis attack (i.e. with a single power trace), as we discuss in Section 4. The authors in [18] did not discuss how exponentiation shall be implemented to meet the leakage bound, nor we can currently find a exponentiation algorithm with these guarantees. Thus, their positive result risks to be void.

This is why we propose an advanced BEG-KEM+, where we avoid blinding by an exponentiation G^{r_i} for a random integer r_i. Our modification is based on the observation that knowledge of the exponent r_i is not needed to perform a successful decryption, but it suffices to build a *random element* in a suitable pairing base group. We propose instead to use a random encoding into asymmetric pairing groups by Fouque and Tibouchi [11]. It turns out that this encoding produces a random element in the base group, and can naturally be implemented in

such a way that the leakage expected against a single measurement is arguably minimal (see Section 4).

Fourthly, we stress that the idea of leakage-resilient cryptography—like any other theoretical concept —can only be brought into practice by actual implementation. For this reason, we implemented BEG-KEM+ in ANSI C on an ARM based microcontroller. BEG-KEM+ is, to our knowledge, the first implementation and evaluation of a public-key scheme from the leakage-resilient literature.

2 Stateful Bilinear ElGamal KEM

In this section we present the stateful bilinear ElGamal Key Encapsulation Mechanism (BEG-KEM) from [18]. First, we recall the basics of the notion of min-entropy. Then we introduce the concept of stateful KEM and security under non-adaptive chosen-ciphertext attacks in the presence of continual min-entropy leakage (CCmLA1). We note again that the class of leakage functions allowed in our model (based on lowering min-entropy) is broader than the bounded length model (CCLA1) used in [18][1].

Min-Entropy

Let X be a finite random variable with probability distribution Pr. The *min-entropy* of X, denoted $\mathbf{H}_\infty(X)$, is defined as $\mathbf{H}_\infty(X) := -\log_2\left(\max_x \Pr[X=x]\right)$. Min-entropy is a standard measure of the worst-case predictability of a random variable. Let Z be a random variable. The *average conditional min-entropy* of X given Z, denoted $\tilde{\mathbf{H}}_\infty(X \mid Z)$, is defined as $\tilde{\mathbf{H}}_\infty(X \mid Z) := -\log_2\Big(\mathbb{E}_{z\leftarrow Z}\Big[\max_x \Pr[X=x \mid Z=z]\Big]\Big)$. Average conditional min-entropy is a measure of the worst-case predictability of a random variable given a correlated random variable.

Lemma 1. [[10]] *Let $f : X \to \{0,1\}^{\lambda'}$ be a function on X. Then $\tilde{\mathbf{H}}_\infty(X \mid f(X)) \geq \mathbf{H}_\infty(X) - \lambda'$.*

The following result is a variant of the Schwartz-Zippel Lemma [29, 38, 13].

Lemma 2. [Schwartz-Zippel; min-entropy version] *Let $\mathsf{F} \in \mathbb{Z}_q[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ be a non-zero polynomial of (total) degree at most d. Let P_i $(i = 1, \ldots, n)$ be probability distributions on $\mathbb{Z}_q$ such that $\mathbf{H}_\infty(P_i) \geq \log q - \lambda'$, where $0 \leq \lambda' \leq \log q$. If $x_i \overset{P_i}{\leftarrow} \mathbb{Z}_q$ $(i = 1, \ldots, n)$ are independent, then $Pr[\mathsf{F}(x_1, \ldots, x_n) = 0] \leq 2^{\lambda'}\frac{d}{q}$.*

Corollary 1. *If $\lambda' < \log q - \omega(\log\log q)$ in Lemma 2, then $Pr[\mathsf{F}(x_1, \ldots, x_n) = 0]$ is negligible (in $\log q$).*

[1] We point out the authors of [18] mention that their results also carry over to a relaxed leakage model, close in spirit to ours. However this model is not fully detailed, and additionally no justification of this fact is given in [18] nor in [19].

Stateful Key Encapsulation Mechanism

Formally, a split-state key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec1}, \mathsf{Dec2})$ consists of four polynomial-time algorithms. Let κ denote the *s*ecurity parameter and λ denote the *leakage parameter*. The key generation procedure $\mathsf{KG}(\kappa, \lambda)$ takes as input κ and λ, and outputs the public key pk, a pair of initial (stateful) secret states (σ_0, σ'_0), and the public parameters $\mathbb{PP}$. The encapsulation procedure $\mathsf{Enc}(pk)$ takes as input pk, and outputs a secret symmetric key K and the corresponding ciphertext C. The stateful decapsulation procedure takes C as an input and outputs $K \in \mathcal{K}$. This procedure is split into two consecutive steps $\mathsf{Dec1}$ and $\mathsf{Dec2}$, where each step accesses distinct parts of the two secret states. The procedures $\mathsf{Dec1}$ and $\mathsf{Dec2}$ may also update the secret key using locally generated fresh randomness:

$$(\sigma_i, w_i) \xleftarrow{r_i} \mathsf{Dec1}(\sigma_{i-1}, C)\ ;\ (\sigma'_i, K) \xleftarrow{r'_i} \mathsf{Dec2}(\sigma'_{i-1}, w_i).$$

The scheme KEM is required to satisfy the following correctness property:

$$\Pr\left[\mathsf{Dec2}\left(\mathsf{Dec1}\left(\mathsf{Enc}\left(pk\right), \sigma_{i-1}\right) \backslash \sigma_i,\ \sigma'_{i-1}\right) = K\ :\right.$$
$$\left.\left(pk, \left(\sigma_{i-1}, \sigma'_{i-1}\right)\right) \leftarrow \left(\mathsf{KG}, \mathsf{Dec1}, \mathsf{Dec2}\right),\ K \leftarrow \mathsf{Enc}\left(pk\right)\right] = 1.$$

The security of the scheme KEM is defined by the following game:

KEM-CCmLA1$_{\mathsf{KEM}}(\mathcal{A}, \kappa, \lambda)$	**KEM-Leak-Oracle** $O^{\mathrm{CCmLA1}}(C, f_i, h_i)$
$(pk, (\sigma_0, \sigma'_0)) \leftarrow \mathsf{KG}(\kappa, \lambda)$	
$i := 1,\ w \leftarrow \mathcal{A}^{O^{\mathrm{CCmLA1}}(\cdot)}(pk)$	$(\sigma_i, w_i) \xleftarrow{r_i} \mathsf{Dec1}(\sigma_{i-1}, C)$
$b \xleftarrow{\$} \{0,1\}$	$(\sigma'_i, K) \xleftarrow{r'_i} \mathsf{Dec2}(\sigma'_{i-1}, w_i)$
$(C, K_0) \leftarrow \mathsf{Enc}(pk)$	$\Lambda_i := f_i(\sigma_{i-1}, r_i)$
$K_1 \xleftarrow{\$} \mathcal{K}$	$\Lambda'_i := h_i(\sigma'_{i-1}, r'_i, w_i)$
$b' \leftarrow \mathcal{A}(w, CK_b)$	$i := i + 1$
	Return $(K, \Lambda_i, \Lambda'_i)$

In the above experiment, $f_i(\sigma_{i-1}, r_i)$ and $h_i(\sigma'_{i-1}, r'_i, w_i)$ are (efficiently computable) leakage functions that the adversary can choose adaptively between the rounds. The functions $f_i(\cdot)$ and $h_i(\cdot)$ are such that the min-entropy of the individual inputs of the leakage functions is decreased by at most λ bits, given the corresponding leakages. More precisely, the requirement on the leakage functions is that

$$\tilde{\mathbf{H}}_\infty\left(\mathsf{t} \mid f_i(\sigma_{i-1}, r_i)\right) \geq \mathbf{H}_\infty\left(\mathsf{t}\right) - \lambda \quad \forall \mathsf{t}\ \in\ \sigma_{i-1} \cup r_i,$$

and

$$\tilde{\mathbf{H}}_\infty\left(\mathsf{t} \mid h_i(\sigma'_{i-1}, r'_i, w_i)\right) \geq \mathbf{H}_\infty\left(\mathsf{t}\right) - \lambda \quad \forall \mathsf{t}\ \in\ \sigma'_{i-1} \cup r'_i \cup w_i.$$

Essentially, the above equations restrict the class of allowed leakage functions to those that do not decrease the min-entropy of each atomic parameter of the secret state by more than λ bits. For instance, if $w_i = \{w_{i,1}, w_{i,2}\}$, then we require that individually $w_{i,1}$ and $w_{i,2}$ have their min-entropy reduced by at most λ bits given the leakages.

Definition 1. [CCmLA1 security for KEM] *A key encapsulation mechanism* KEM *is* secure under non-adaptive chosen-ciphertext attacks in the presence of continual split-state leakage (CCmLA1)*, with min-entropy leakage bound* λ*, if* $\Pr[b' = b]$ *is at most negligibly greater than* $\frac{1}{2}$ *in the Experiment* $\text{KEM-CCmLA1}_{\mathsf{KEM}}(\mathcal{A}, \kappa, \lambda)$ *for any efficient adversary* $\mathcal{A}$*.*

Note that if in the above definition we would force the leakage functions to have output length of at most λ bits, then we would obtain the *CCLA1* security for KEM as defined in [18]. From Lemma 1, we have that the conditional min-entropy of a random variable, given the leakage output of at most λ bits, cannot decrease by more than λ bits. Hence if a KEM is CCLA1 secure, then it is also CCmLA1 secure.

Bilinear Groups

Let $\mathsf{BGen}'(\kappa, \lambda)$ be a probabilistic bilinear group generator that outputs $(\mathbb{G}, \mathbb{G}_T, q, e', g)$ such that:

1. $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are (multiplicatively written) cyclic groups of prime order q with binary operations $\cdot$ and $\star$, respectively. The size of q is κ bits.
2. $e' : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map that is:
   (a) bilinear: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $e'(u^a, v^b) = e'(u, v)^{ab}$.
   (b) non-degenerate: $e'(g, g) \neq 1$.

Such a group $\mathbb{G}$ is said to be a bilinear group if the above properties hold and the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the map e' are efficiently computable. The group $\mathbb{G}$ is called as *base group* and $\mathbb{G}_T$ as *target group*.

2.1 Bilinear ElGamal KEM

The scheme $\mathsf{BEG} = \left(\mathsf{KG}_{\mathsf{BEG}}, \mathsf{Enc}_{\mathsf{BEG}}, \mathsf{Dec1}_{\mathsf{BEG}}, \mathsf{Dec2}_{\mathsf{BEG}}\right)$ is as follows:

1. $\mathsf{KG}_{\mathsf{BEG}}(\kappa)$: Compute $\mathbb{PP} = (\mathbb{G}, \mathbb{G}_T, e', q, g) \leftarrow \mathsf{BGen}'(\kappa, \lambda)$ and randomly choose $x, t_0 \xleftarrow{\$} \mathbb{F}_q$. Set $X = g^x$, $\sigma_0 = g^{t_0}$, $\sigma_0' = g^{x - t_0}$, and $X_T = e'(g, g)^x$. Return (pk, sk_0), where
   (a) the public key is $pk = (\mathbb{PP}, X_T)$.
   (b) the secret state is $sk_0 = (\sigma_0, \sigma_0') \in \mathbb{G} \times \mathbb{G}$.
2. $\mathsf{Enc}_{\mathsf{BEG}}(pk)$*:* Choose a random $r \xleftarrow{\$} \mathbb{F}_q$. Compute the ciphertext $C = g^r$, and the derived key $K = X_T^r$. Return (C, K).
3. $\mathsf{Dec1}_{\mathsf{BEG}}(\sigma_{i-1}, C)$: Choose a random $t_i \xleftarrow{\$} \mathbb{F}_q$, set $\sigma_i = \sigma_{i-1} \cdot g^{t_i}$, $Y_i = e'(\sigma_i, C)$. Return (t_i, Y_i).
4. $\mathsf{Dec2}_{\mathsf{BEG}}(\sigma_{i-1}', (t_i, Y_i), C)$: Set $\sigma_i' = \sigma_{i-1}' \cdot g^{-t_i}$, and $Y_i' = e'(\sigma_i', C)$. Compute the derived key $K = Y_i \cdot Y_i' \in \mathbb{G}_T$. Return K.

The correctness of the scheme follows from the fact that $\sigma_i \cdot \sigma'_i = X \ \forall i \geq 0$ and using the bilinearity of $e'()$.

In [18], the above scheme is shown to be secure in the generic bilinear group model [32, 8] under (non-adaptive) chosen-ciphertext attacks in the presence of continual bounded-size leakage (in short, CCLA1 security). The basic motivation for splitting the decapsulation step into two parts comes from the "only computation leaks information" axiom [23], which states that any leakage of information occurs only from the data that is being currently accessed by the computation.

Theorem 1. [18, Theorem 1] *The scheme* BEG *(also called BEG-KEM) is CCLA1 secure in the generic bilinear group model. The advantage of an* s*-query adversary who gets at most* λ *bits of leakage per each invocation of* $\mathsf{Dec1}_{\mathsf{BEG}}$ *or* $\mathsf{Dec2}_{\mathsf{BEG}}$ *is at most* $\frac{s^3}{q}2^{2\lambda+1}$.

2.2 A CCmLA1 Security Reduction in the Generic Bilinear Group Model

We show that BEG-KEM is also leakage-resilient in the min-entropy leakage model introduced above, where leakage functions are not necessarily size-bounded. The only restriction is that the inputs to the leakage functions shall have enough min-entropy left, as a function of a leakage parameter λ, given the corresponding outputs. Interestingly, by using a different proof technique than [19], we obtain a tighter bound on the adversarial CCLmA1 advantage than the bound claimed in [18] for the adversarial CCLA1 advantage, w.r.t. the number of oracle queries s. In other words, with respect to the previous work, we provide here a new security reduction under a more realistic leakage model, and surprisingly we achieve better tightness.

Theorem 2. *The scheme* BEG-KEM *is CCmLA1 secure in the GBG model. The advantage of an* s*-query adversary with min-entropy leakage bound* λ *is* $\left(\frac{9s^2+3s}{q}\right)2^{2\lambda}$.

At a high level, the proof of this theorem proceeds in two steps as in [13, 12]. First we show in Theorem 3 (contained in Appendix A) that the scheme is secure if there is no leakage, i.e., CCA1 security. Note that the adversary is transparent to the internal details of secret state updates. Then, we complete the proof of CCmLA1 security by analyzing the effect of leakage on the CCA1 security.

The main idea to prove the CCA1 security is that the adversary will not be able to compute the derived symmetric key K_0 even after seeing the challenge ciphertext. To show this we just need to prove that K_0 cannot be written as a "linear combination" of the elements of $\mathbb{G}_T$ that it has got as input or can compute itself using the pairing oracle along with the input elements of $\mathbb{G}$. Hence in the GBG model it will not be able to distinguish the actual derived key or a randomly chosen key in $\mathbb{G}_T$. The challenger simulates the security game

$\mathcal{G}$ to the adversary in the naive way. Also, the challenger simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements.

We then argue that that the proof for the non-leakage setting (i.e. proof of Theorem 3) and that for the leakage setting would be the same conditioned on the fact that the adversary is unable to derive useful relation amongst the elements it has seen or guessed, and that it will not be able to compute and hence leak the full secret key X through the leakage functions, if λ is sufficiently small. Finally, we show that the probability of this event is increased by a factor of at most $2^{2\lambda}$ compared to the non-leakage setting. For the formal proof see Appendix A.

3 BEG-KEM+ : A Leakage-Resilient KEM Closer to Practice

Our choice of BEG-KEM for this investigation is entirely motivated by the fact that a similar leakage-resilience result as that proven in [18] cannot be expected for a pairing-less group, as shown in [14]. This motivates using pairing groups to implement ElGamal.

On the other hand, while Theorem ensures a protection against side-channel attacks that combine traces of different computations (e.g. differential power analysis attacks), we still need protection against single trace attacks, i.e. Simple Power Analysis (SPA). The use of pairing groups can help on this respect, as pointed out by Scott in [30]:

> "[...] it is of interest to consider the resistance of pairing-based protocols to so-called SPA attacks [...] one might with reasonable confidence expect that the power consumption profile of (and execution time for) such protocols will be constant and independent of any secret values."

We continue by proposing a tweak to BEG-KEM with the aim to make the most, from a minimizing leakage perspective, out of our choice of using pairing groups to realize leakage-resilient public key cryptographic primitives.

3.1 An Advanced BEG-KEM+ More Resistant to Side-Channel Attacks

Let us first make the observation that $\mathsf{Dec1}^*_{\mathsf{BEG}}$ is picking a random point in the pairing based group $\mathbb{G}$ by computing an exponentiation g^r for a random r. As is well-known, a naïve implementation of exponentiation can leak the entire exponent r, which would, of course, invalidate the required bound of maximum leakage in our new (as well as in the old) model. This leads us to the question whether it is possible, given the large body of side-channel resistant exponentiation techniques, to find an algorithm that would likely meet the leakage bound for single measurements. In other words, we have to answer the question of whether the exponentiation can be made resistant against SPA attacks.

Exponentiation in a multiplicative group (or scalar multiplication in an elliptic curve group) of large order involves hundreds or even thousands of low-level arithmetic operations such as modular multiplication. Unfortunately, all these low-level operations are (either directly or indirectly) controlled by the secret exponent, which means that each of them can potentially leak sensitive information (see e.g. [36, 35, 34] for further details). Consequently, we need both an SPA-resistant exponentiation algorithm and an SPA-resistant implementation of the underlying multiple-precision operations. The latter is difficult to achieve in software due to side-channel leakage induced by certain micro-architectural features such as the early-termination mechanism of integer multipliers in ARM processors [16]. For example, it was shown in [16] that highly regular exponentiation (resp. scalar multiplication) techniques, which are (in theory) perfectly SPA-resistant, succumb to an SPA attack when exploiting the early-termination mechanism. Therefore, we avoid exponentiation with a secret exponent in our modified scheme[2].

A careful analysis of BEG-KEM reveals that $\mathsf{Dec1}^*_{\mathsf{BEG}}$ only needs to sample uniformly at random an element u of $\mathbb{G}$, and that knowledge of $\log_g u$ is *not necessary*. For this reason, we decided to build a random u in the pairing base group by using a so-called *encoding* to the base group [31, 17, 11]. Roughly speaking, an encoding is a deterministic function mapping an arbitrary string to a point in an elliptic curve. Recently, Fouque and Tibouchi [11] proposed a modification of the Shallue and van de Woestijne encoding into arbitrary elliptic curves [31], that maps arbitrary strings to Barreto-Naehrig asymmetric pairing groups [4]. Let $f : \mathbb{F}_p^* \to E(\mathbb{F}_p)$ be the Fouque-Tibouchi encoding. Then, $(t_1, t_2) \mapsto u = u_1 \cdot_E u_2$ builds a point $u \in E(\mathbb{F}_p)$ distributed uniformly at random if $t_1, t_2 \xleftarrow{\$} \mathbb{F}_p^*$, where $\cdot_E$ is the addition operation in $E(\mathbb{F}_p)$. Additionally, [11] points out that f can be naturally implemented so that its computation is completely independent of the inputs, which clearly helps us towards meeting our desired min-entropy leakage bound.

BEG-KEM+

Let ABGen be an asymmetric bilinear group generator that outputs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, g_1, g_2)$ with $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = q$, where q is a prime, κ be the security parameter, and λ be the leakage parameter. We will again use the multiplicative notation for group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a type 3 pairing map, i.e., e is a non-degenerate bilinear map with no known efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. These groups are instantiated using the BN curves, denoted $E(\mathbb{F}_p)$, of the form $y^2 = x^3 + b$, where $b \in \mathbb{F}_p$ [4].

[2] As mentioned previously, the secret exponent controls a large number of multiple-precision arithmetic operations, which execute an even larger number of `mul` instructions. Each of these `mul` instructions can potentially trigger the early-termination mechanism and, hence, leak information about the secret exponent. In our modified scheme, the secret value is only used as input of a multiple-precision operation and does not control any other operations.

Algorithm 1. Shallue-van de Woestijne encoding to BN curves $y^2 = x^3 + b$ [11]

Input: A random number $t \in \mathbb{F}_p^\star$.
Output: Point $P \in E(\mathbb{F}_p)$
1: $w \leftarrow \sqrt{-3} \cdot t/(1 + b + t^2)$
2: $x_1 \leftarrow (-1 + \sqrt{-3})/2 - tw$
3: $x_2 \leftarrow -1 - x_1$
4: $x_3 \leftarrow 1 + 1/w^2$
5: $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{F}_p^\star$
6: $\alpha \leftarrow \chi_q(r_1^2 \cdot (x_1^3 + b))$
7: $\beta \leftarrow \chi_q(r_2^2 \cdot (x_2^3 + b))$
8: $i \leftarrow [(\alpha - 1) \cdot \beta \bmod 3] + 1$
9: **return** $P[x_i, \chi_q(r_3^2 \cdot t) \cdot \sqrt{(x_i^3 + b)}]$

Also, let G_1 and G_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $f : \mathbb{F}_p^* \to \mathbb{G}_1$ be the Fouque-Tibouchi encoding of the elements of $\mathbb{G}_1$.

The advanced $\mathsf{BEG-KEM+} = \left(\mathsf{KG}^+_{\mathsf{BEG}}, \mathsf{Enc}^+_{\mathsf{BEG}}, \mathsf{Dec1}^+_{\mathsf{BEG}}, \mathsf{Dec2}^+_{\mathsf{BEG}}\right)$ is defined as follows:

1. $\mathsf{KG}^+_{\mathsf{BEG}}(\kappa)$: Compute $\mathbb{PP} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, G_1, G_2) \leftarrow \mathsf{ABGen}(\kappa)$ and randomly choose $x, t_0 \xleftarrow{\$} \mathbb{F}_q$. Set $X = G_1^x$, $\sigma_0 = G_1^{t_0}$, $\sigma_0' = G_1^{(x-t_0)}$, and $X_T = e\left(G_1, G_2\right)^x$. Return (pk, sk_0), where
   (a) the public key is $pk = (\mathbb{PP}, X_T)$.
   (b) the secret state is $sk_0 = (\sigma_0, \sigma_0')$.
2. $\mathsf{Enc}^+_{\mathsf{BEG}}(pk)$*:* Choose a random $r \xleftarrow{\$} \mathbb{F}_p$. Compute the ciphertext $C = G_2^r$, and the derived key $K = X_T^r$. Return (C, K).
3. $\mathsf{Dec1}^+_{\mathsf{BEG}}(\sigma_{i-1}, C)$: Choose random $t_i, z_i \xleftarrow{\$} \mathbb{F}_p^*$, set $u_i = f\left(t_i\right) \cdot f\left(z_i\right)$, and compute $\sigma_i = \sigma_{i-1} \cdot u_i$, $Y_i = e\left(\sigma_i, C\right)$. Return (u_i, Y_i).
4. $\mathsf{Dec2}^+_{\mathsf{BEG}}(\sigma_{i-1}', (u_i, Y_i), C)$: Set $\sigma_i' = \sigma_{i-1}' \cdot (u_i)^{-1}$, and $Y_i' = e\left(\sigma_i', C\right)$. Compute the derived key $K = Y_i \cdot Y_i' \in \mathbb{G}_T$. Return K.

Algorithm 1 describes the constant-time hashing function to BN curves from [11]. As described in the original paper, implementing this algorithm against timing and Simple Power Analysis (SPA) attacks is not difficult to be achieved. In step 6 and 7, instead of computing the values $\chi_q(x_1^3 + b)$ and $\chi_q(x_2^3 + b)$ in a straightforward way, which can leak secret data, the authors suggested to use blinding. Namely, in order to get α and β, we actually evaluate $\chi_q(r_1^2 \cdot (x_1^3 + b))$ and $\chi_q(r_2^2 \cdot (x_2^3 + b))$, where r_1 and r_2 are random field elements generated in Step 5. On the other hand, in order to prevent the leakage while computing the index i, they employ a specific algebraic function $\phi(\alpha, \beta) = [(\alpha - 1) \cdot \beta \bmod 3] + 1$, which runs in constant time.

4 Secure Implementation and Performance Analysis

In this section, we first describe a software implementation of BEG-KEM+ (along with the instantiation of the underlying pairing groups) and present the

execution times we measured on an ARM Cortex M-3 processor. The second part of this section is devoted to a "practical" security evaluation of BEG-KEM+ by analyzing potential sources of information leakage in the underlying arithmetic operations that could be exploited to mount a side-channel attack.

4.1 Implementation Details and Performance Analysis

We implemented both BEG-KEM and BEG-KEM+ in Magma and ANSI C, whereby the former implementation served as a reference for the latter. The C implementation is based on the MIRACL library to ensure an efficient execution of the pairing evaluation and all other arithmetic operations in the diverse groups and fields. We instantiated both BEG-KEM and our improved scheme using the Ate pairing over a 254-bit Barreto-Naehrig (BN) curve. More specifically, our implementations adopts the curve BN254 from [27], which provides a security level roughly comparable to that of 128-bit AES. BN curves are defined by a Weierstrass equation of the form $y^2 = x^3 + b$ over a prime field $\mathbb{F}_q$, whereby q can be written as polynomial $p(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ for some parameter u [4]. In our case, $u = -(2^{62} + 2^{55} + 1) = -0\text{x}4080000000000001$ and, hence, q has a length of 254 bits. The curve BN254 is given by the equation $y^2 = x^3 + 2$ (i.e. $b = 2$) and has prime order with embedding degree $k = 12$.

The execution times for various arithmetic operations in the different fields and groups are summarized in Table 1, whereby all timings are specified in millions of clock cycles. Our prototype platform for performance evaluation is an Arduino Due microcontroller board equipped with an ARM Cortex-M3 CPU. Even though the three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ have the same order, the underlying multiple-precision arithmetic operations are performed with operands of different size. $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve groups over $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$, the elements of which have, in our case, a bitlength of 254 and 508 bits, respectively. The group $\mathbb{G}_T$ is a subgroup of the multiplicative group of the extension field $\mathbb{F}_{q^{12}}$, i.e. the modular multiplications for exponentiation in $\mathbb{G}_T$ are carried out on 3048-bit operands.

Table 1. Running times for field exponentiation, square root, inversion, group exponentiation and pairing operations (in 10^6 clock cycles)

Operation	Running time
Square root $\mathbb{F}_q$	0.7
Inversion $\mathbb{F}_q$	0.087
Encoding to $\mathbb{G}_2$	3.7
Exponentiation $\mathbb{G}_1$	4.5
Exponentiation $\mathbb{G}_2$	10.0
Exponentiation $\mathbb{G}_T$	27.1
Pairing	65.0

Table 2. Comparison of running times for key generation, encapsulation and decapsulation for BEG-KEM and BEG-KEM+ (in 10^6 clock cycles)

Operation	BEG-KEM	BEG-KEM+
KeyGen	108	108
Encryption	34	34
Decryption	131	140

The execution times for key generation, encapsulation as well as decapsulation for both BEG-KEM and BEG-KEM+ are given in Table 2. Our results show that an encapsulation can be carried out in 34 million clock cycles, while the decapsulation takes about 140 million cycles. We observe that our modified decapsulation algorithm is roughly 6% slower than the original one.

4.2 Side-Channel Resistance from a Practical Point of View

One of the fundamental principles of leakage-resilient cryptography is to use a critical secret only once (or a few times), which ensures that an attacker is not able to retrieve the secret key if the per-invocation leakage is in some way "limited" or "bounded." In every invocation of the scheme or function, the secret is either "refreshed" or a completely new secret is generated randomly. The original BEG-KEM scheme from [18], and also our variant BEG-KEM+, follow this principle. As a consequence, all forms of side-channel attack that require several executions of a cryptographic function with one and the same secret key, e.g. Differential Power Analysis (DPA), are obviously not applicable to BEG-KEM+ (and in fact the latter is guaranteed by Theorem 2). However, attacks that aim to recover the secret key from information leaked from a single invocation of a cryptographic function (i.e. Simple Power Analysis (SPA) attacks) may succeed under certain conditions. The group exponentiation computed in the BEG-KEM scheme to derive a random group element $\sigma_0 = g^{t_0}$ serves as a good example. If this exponentiation is implemented in completely straightforward way (e.g. using the square-and-multiply method) an attacker can obtain t_0 if he is able to distinguish group squarings from group products in the power consumption profile. Such SPA attacks on unprotected or insufficiently protected ECC implementations are fairly easy and have been reported extensively in the literature, see e.g. [6, Chapter IV] and the references therein. Therefore, we advocated to replace the afore-mentioned group exponentiation by a deterministic encoding into an elliptic curve group [11].

SPA Resistance of Pairing Evaluation. Section 3.1 quotes a statement of Scott [30, Section 3.1] saying that one can expect the power consumption profile of a pairing-based protocol to be independent of any secret values. An intuitive explanation why pairings are fairly "robust" against SPA leakage is also given in [30]: the target of the attack is a secret point, which is generally much harder to reveal than e.g. a secret scalar or a secret exponent. As mentioned before, our implementation uses the Ate pairing instantiated on a BN curve over a 254-bit prime field $\mathbb{F}_p$. Consequently, the secret is the x and y coordinate of an elliptic curve point, which are in our case simply elements of $\mathbb{F}_p$. The only way in which an attacker can hope to gain information about x and y is by inspecting the power consumption and execution time of the $\mathbb{F}_p$-arithmetic operations (e.g. addition, multiplication) performed on them. However, the operand-related SPA leakage from field-arithmetic operations is generally very small. To explain this in detail, let us use the addition in $\mathbb{F}_p$ as example, which is nothing else than

a modular addition $r = a + b \bmod p$. We assume that a is a secret value and that b is known to the attacker. A modular addition consists of an ordinary addition $s = a + b$, followed by a subtraction if the sum s is equal to or bigger than p. Conventional wisdom from the side-channel community says that such a conditional subtraction causes differences in the power consumption profile (and also execution time), which is observable by an attacker. However, the information content is very small; in fact, when the subtraction is executed the attacker just knows that $a + b \geq p$, i.e. he has learned that $a \geq p - b$.

The situation is similar for multiplication in $\mathbb{F}_p$, which is nothing else than a modular multiplication $r = a \cdot b \bmod p$. Again, we assume that a is the secret value and that b is known to the attacker. A modular multiplication involves a conventional multiplication $t = a \cdot b$, followed by a modular reduction $r = t \bmod p$, which is in pairing-based cryptography typically implemented using Montgomery's algorithm [24]. Both the multiplication and Montgomery reduction are highly regular (i.e. do not have to execute any conditional statements), except for the so-called "final subtraction." Montgomery's reduction technique does not directly compute $t \bmod p$ but produces the following output

$$x = \big(t + (t \cdot p' \bmod 2^n) \cdot p\big)/2^n \tag{1}$$

where $p' = -p^{-1} \bmod 2^n$ and n is the bitlength of p. Note that x may be not fully reduced, which means a final subtraction of p is necessary to get the least non-negative residue as result. An attacker able to observe whether or not this final subtraction is executed learns only whether $x \geq p$ or not, which does not reveal much information about a. The same also holds for subtraction and squaring in $\mathbb{F}_p$. However, a noteworthy exception is the inversion operation, which we will further discuss below. In summary, a straightforward implementation of the arithmetic operations (bar inversion) in $\mathbb{F}_p$ leaks only very little information about the operands, which confirms that pairing evaluation is, in general, not susceptible to SPA attacks. To our knowledge, the recent literature contains only two papers in which SPA attacks on pairings are discussed [26, 37], but both of them are only relevant for pairings over binary fields where the multiplication is implemented in a highly irregular way. The attack from [35] is only applicable to scalar multiplication with a secret scalar, but not to pairings with a secret point.

SPA Resistance of Encoding Function. The encoding function shown in Algorithm 1 consists of a number of basic arithmetic operations (e.g. addition, multiplication) in the field $\mathbb{F}_p$. Furthermore, two inversions are executed, one in step 1 and the other in step 4. The straightforward approach to invert an element of a finite field is the Extended Euclidean Algorithm (EEA). Conventional wisdom from the side-channel community says that the EEA is a highly irregular algorithm, executing many conditional operations, which is likely to leak SPA-relevant information about the operand to be inverted. In order to prevent an SPA attack on the inversion operation, we apply a simple multiplicative masking. That is, instead of inverting a field element v directly, we first multiply it

by a random number r, which yields the product $t = v \cdot r$. Then, we invert this product using the EEA to obtain $1/t = 1/(v \cdot r)$, which we finally multiply again by r to get $1/v$ as result.

The function χ in step 6 and 7 of Algorithm 1 is essentially an evaluation of the Legendre Symbol, which, in turn, consists of an exponentiation using a constant public exponent (i.e. $(p+1)/4$). The input to the χ function is "blinded" by the random value r_1^2 and r_2^2, which means the underlying exponentiation can not leak any SPA-relevant information. As mentioned in Section 3.1, a constant-time algebraic function is adopted for the calculation of the index i in step 8, which also cannot leak.

5 Conclusion

In this paper we aimed to bring the concept of leakage-resilient cryptography closer to practice. Most of the leakage-resilient public key cryptography schemes proposed until now are too inefficient for real-world applications. Even though they provide provable security against a large class of side-channel attacks, they do so under certain leakage models and leakage bound requirements that are far from what we can ensure in practice. On the other hand, the side-channel countermeasures are often ad-hoc and do not provide enough security guarantees. We addressed this problem by bringing best practices from both worlds together. First, we argued that a naive implementation of the pairing group exponentiation in the leakage-resilient ElGamal key encapsulation mechanism proposed by Kiltz and Pietrzak makes it impossible to reach the required leakage bound. To overcome this problem, we have made two additional contributions. On the one hand, we have proposed a relaxed leakage model, that we call min-entropy leakage, that lifts the restriction on the image size of leakage functions, and proposes instead to require that the inputs to the leakage functions have sufficient min-entropy left, in spite of the leakage. On the other hand, we adopted a different mechanism for finding a random point in an elliptic curve group, namely the encoding of Fouque and Tibouchi. We assessed the security of our implementation from both a theoretical and a practical perspective and argued that it is indeed secure in both worlds. BEG-KEM+ is, to our knowledge, the first leakage-resilient public-key scheme that has been successfully implemented and evaluated on an embedded 32-bit processor.

References

1. D. Aggarwal and U. Maurer. The leakage-resilience limit of a computational problem is equal to its unpredictability entropy. In D. H. Lee and X. Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 686–701. Springer, 2011.
2. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In O. Reingold, editor, *TCC*, volume 5444 of *LNCS*, pages 474–495. Springer, 2009.
3. J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

4. P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *LNCS*, pages 319–331. Springer, 2005.
5. S. Belaïd, V. Grosso, and F.-X. Standaert. Masking and leakage-resilient primitives: One, the other(s) or both? Cryptology ePrint Archive, Report 2014/053, 2014. `http://eprint.iacr.org/`.
6. I. F. Blake, G. Seroussi, and N. P. Smart. *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 2005.
7. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
8. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical Identity Based Encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.
9. Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In D. Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 361–381. Springer, 2010.
10. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
11. P.-A. Fouque and M. Tibouchi. Indifferentiable hashing to Barreto-Naehrig curves. In *LATINCRYPT*, pages 1–17, 2012.
12. D. Galindo and S. Vivek. A leakage-resilient pairing-based variant of the Schnorr signature scheme. In M. Stam, editor, *IMA Int. Conf.*, volume 8308 of *LNCS*, pages 173–192. Springer, 2013.
13. D. Galindo and S. Vivek. A practical leakage-resilient signature scheme in the generic group model. In *SAC 2012*, volume 7707 of *LNCS*, pages 50–65. Springer, 2013.
14. D. Galindo and S. Vivek. Limits of a conjecture on a leakage-resilient cryptosystem. *Inf. Process. Lett.*, 114(4):192–196, 2014.
15. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.
16. J. Großschädl, E. Oswald, D. Page, and M. Tunstall. Side-channel analysis of cryptographic software via early-terminating multiplications. In D. Lee and S. Hong, editors, *Information Security and Cryptology — ICISC 2009*, volume 5984 of *LNCS*, pages 176–192. Springer Verlag, 2010.
17. T. Icart. How to hash into elliptic curves. In *CRYPTO*, pages 303–316, 2009.
18. E. Kiltz and K. Pietrzak. Leakage resilient ElGamal encryption. In M. Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 595–612. Springer, 2010.
19. E. Kiltz and K. Pietrzak. Leakage resilient ElGamal encryption. Full version of [18]. `http://homepage.ruhr-uni-bochum.de/Eike.Kiltz/papers/elgamal_leak.pdf`. Last accessed June 04, 2014, 2010.
20. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
21. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
22. U. M. Maurer. Abstract models of computation in cryptography. In N. P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *LNCS*, pages 1–12. Springer, 2005.

23. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In M. Naor, editor, *TCC*, volume 2951 of *LNCS*, pages 278–296. Springer, 2004.
24. P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, Apr. 1985.
25. A. Moradi. Statistical tools flavor side-channel collision attacks. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 428–445. Springer, 2012.
26. D. Page and F. Vercauteren. Fault and side-channel attacks on pairing based cryptography. *IACR Cryptology ePrint Archive*, 2004:283, 2004.
27. G. C. Pereira, M. A. Simplício, M. Naehrig, and P. S. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, Aug. 2011.
28. E. Prouff and M. Rivain. Masking against side-channel attacks: A formal security proof. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.
29. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.
30. M. Scott. Computing the tate pairing. In A. Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 293–304. Springer, 2005.
31. A. Shallue and C. van de Woestijne. Construction of rational points on elliptic curves over finite fields. In F. Hess, S. Pauli, and M. E. Pohst, editors, *ANTS*, volume 4076 of *LNCS*, pages 510–524. Springer, 2006.
32. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.
33. F.-X. Standaert, O. Pereira, and Y. Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO (1)*, volume 8042 of *LNCS*, pages 335–352. Springer, 2013.
34. D. Stebila and N. Thériault. Unified point addition formulæ and side-channel attacks. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *LNCS*, pages 354–368. Springer Verlag, 2006.
35. C. D. Walter. Simple power analysis of unified code for ECC double and add. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems — CHES 2004*, volume 3156 of *LNCS*, pages 191–204. Springer Verlag, 2004.
36. C. D. Walter and S. Thompson. Distinguishing exponent digits by observing modular subtractions. In D. Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, volume 2020 of *LNCS*, pages 192–207. Springer Verlag, 2001.
37. C. Whelan and M. Scott. Side channel analysis of practical pairing implementations: Which path is more secure? In P. Q. Nguyen, editor, *VIETCRYPT*, volume 4341 of *LNCS*, pages 99–114. Springer, 2006.
38. R. Zippel. Probabilistic algorithms for sparse polynomials. In E. W. Ng, editor, *EUROSAM*, volume 72 of *LNCS*, pages 216–226. Springer, 1979.

A Proof of Theorem 2.

First we briefly recollect the notion of generic bilinear groups. We use the notation used in [18, 13] Then we present a formal proof of Theorem 2. Our proof is based on the techniques from [13, 12].

Generic Bilinear Group Model

The generic bilinear group (GBG) model [8] is an extension of the generic group model [32]. The encodings of the elements of $\mathbb{G}$ and $\mathbb{G}_T$ are given by random bijective maps $\xi : \mathbb{Z}_q \to \Xi$ and $\xi_T : \mathbb{Z}_q \to \Xi_T$, respectively, where Ξ and Ξ_T are sets of bit-strings. The group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and evaluation of the bilinear map e are performed by three public oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, respectively, defined as follows. For all $a, b \in \mathbb{Z}_q$

- $\mathcal{O}(\xi(a), \xi(b)) := \xi(a + b \bmod q)$
- $\mathcal{O}_T(\xi_T(a), \xi_T(b)) := \xi_T(a + b \bmod q)$
- $\mathcal{O}_e(\xi(a), \xi(b)) := \xi_T(ab \bmod q)$

We assume that the (fixed) generator g of $\mathbb{G}$ satisfies $g = \xi(1)$, and also the (fixed) generator g_T of $\mathbb{G}_T$ satisfies $g_T = e(g, g) = \xi_T(1)$. The encoding of g is provided to all users of the group oracles. The users can thus efficiently sample random elements in both $\mathbb{G}$ and $\mathbb{G}_T$.

We further assume that $\Xi \cap \Xi_T = \phi$, $|\Xi| = |\Xi_T| = q$, and that the elements of Ξ and Ξ_T are efficiently recognizable. For instance, the encodings in Ξ can comprise of the binary representation of the set $\{0, 1, \ldots, q-1\}$, where every string begins with '0' and all are of uniform length. The encodings in Ξ_T are similarly defined but instead begin with '1'. Since the encodings are efficiently recognizable, the queries to a group oracle with an invalid encoding can be detected and an error can be raised. For simplicity, we assume that the users' queries to the oracles are all valid.

A.1 The Proof of Theorem 2

The proof of this theorem proceeds in two steps as in [13, 12]. First we show in Theorem 3 that the scheme is secure if there is no leakage, i.e., CCA1 security. Note that the adversary is transparent to the internal details of secret state updates. Then, in Section A.2, we complete the proof of CCmLA1 security by analyzing the effect of leakage on the CCA1 security.

Non-Leakage Setting: CCA1 Security

Theorem 3. *The scheme* BEG *is CCA1 secure in the generic bilinear group model, i.e., it is secure against non-adaptive chosen-ciphertext attacks if there is no leakage of the secret states. The advantage of an s-query adversary is at most $\frac{1}{2} + \frac{9s^2}{q}$.*

Proof. Let $\mathcal{A}$ be an s-query adversary that can break the CCA1 security of BEG. Hence $\mathcal{A}$ can make totally at most s group oracle, pairing oracle and decryption oracle queries. Let $s_{\mathcal{O}}$ denote the total number of calls to the oracles $\mathcal{O}$, $\mathcal{O}_T$ and $\mathcal{O}_e$, and s_D denote the number of calls to the decryption oracle O^{CCA1}. Thus

$s_{\mathcal{O}} + s_D \leq s$. Let $\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}}$ denote the success probability of the adversary $\mathcal{A}$ in making a correct guess of b' in the security game. We show that

$$\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}} \leq \frac{1}{2} + \frac{9s^2}{q}.$$

for any s-query adversary $\mathcal{A}$ in the GBG model.

The main idea is to show that $\mathcal{A}$ will not be able to compute the derived symmetric key K_0 even after seeing the challenge ciphertext C^*. To show this we just need to prove that K_0 cannot be written as a "linear combination" of the elements of $\mathbb{G}_T$ that it has got as input or can compute itself using the pairing oracle $\mathcal{O}_e$ along with the input elements of $\mathbb{G}$. Hence in the GBG model it will not be able to distinguish the actual derived key or a randomly chosen key in $\mathbb{G}_T$. The challenger $\mathcal{C}$ simulates the security game $\mathcal{G}$ to $\mathcal{A}$ in the naive way. Also, $\mathcal{C}$ simulates the generic bilinear group oracles in the usual way by maintaining lists of pairs of encodings and polynomials that represent the relation amongst group elements.

We now formally describe the game $\mathcal{G}$. The description of the group oracles is typical for proofs in the generic group model (see [32, 22, 7, 13]).

Description of Game $\mathcal{G}$: Let X, R,$\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ be indeterminates. Intuitively, these (or other) polynomials represent the relation amongst the group elements that are output by a group oracle, or guessed by $\mathcal{A}$. The indeterminate X corresponds to the quantity x (discrete logarithm of the secret key), and R corresponds to the challenge ciphertext. Since $\mathcal{A}$ can query the group oracles with representations (from Ξ and Ξ_T) not previously obtained from the group oracles, in order to accommodate this case, we introduce the indeterminates U_i, V_i. The U_i correspond to the guessed elements of $\mathbb{G}$, whereas V_i correspond to the guessed elements of $\mathbb{G}_T$. We denote the lists $\{\mathsf{U}_i : i \geq 1\}$ and $\{\mathsf{V}_i : i \geq 1\}$ by $\{\mathsf{U}\}$ and $\{\mathsf{V}\}$, respectively.
$\mathcal{C}$ maintains two lists of pairs

$$\mathcal{L} = \{(F_{1,i}\,,\, \xi_{1,i}) : 1 \leq i \leq \tau_1\}, \tag{2}$$

$$\mathcal{L}_T = \{(F_{T,i}\,,\, \xi_{T,i}) : 1 \leq i \leq \tau_T\}. \tag{3}$$

The entries $\mathsf{F}_{1,i} \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$, $\mathsf{F}_{T,i} \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{V}\}]$ are multivariate polynomials over $\mathbb{Z}_q$, whereas $\xi_{1,i}$, and $\xi_{T,i}$ are bit-strings in the encoding sets Ξ (of $\mathbb{G}$) and Ξ_T (of $\mathbb{G}_T$), respectively. The polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to (more precisely, a superset of the) elements of $\mathbb{G}$ and $\mathbb{G}_T$, respectively, that $\mathcal{A}$ will ever be able to compute or guess. The values τ_1 and τ_T denote the respective list counters. In order to simplify the description, we view $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$ as a subring of $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{V}\}]$.

Initially, $\tau_1 = 1$, $\tau_T = 1$, $\mathcal{L} = \{\ (1, \xi_{1,1})\ \}$, and $\mathcal{L}_T = \{\ (\mathsf{X}, \xi_{T,1})\ \}$. The bit-strings $\xi_{1,1}$, $\xi_{T,1}$ are set to random distinct strings from Ξ and Ξ_T, respectively. We assume that there is some ordering among the strings in the sets Ξ and Ξ_T (say, lexicographic ordering), so that given a string $\xi_{1,i}$ or $\xi_{T,i}$, it is possible to efficiently determine its index in the lists, if it exists. The initial state of the lists $\mathcal{L}$ and $\mathcal{L}_T$ correspond to the generator of $\mathbb{G}$ and the public key, respectively. The

game begins by $\mathcal{C}$ providing $\mathcal{A}$ with the string $\xi_{1,1}$ from $\mathcal{L}$, and the string $\xi_{T,1}$ from $\mathcal{L}_T$.

Group Operation of $\mathbb{G}$: The calls made by $\mathcal{A}$ to the group oracle $\mathcal{O}$ are modeled as follows. For group operations in $\mathbb{G}$, $\mathcal{A}$ provides $\mathcal{C}$ with two operands (bit-strings) $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$ and also specifies whether to multiply or divide them. $\mathcal{C}$ answers the query by first incrementing the counter $\tau_1 := \tau_1 + 1$, and computes the polynomial $\mathsf{F}_{1,\tau_1} := F_{1,i} \pm F_{1,j}$. If $\mathsf{F}_{1,\tau_1} = \mathsf{F}_{1,k}$ for some $k < \tau_1$, then $\mathcal{C}$ sets $\xi_{1,\tau_1} := \xi_{1,k}$. Otherwise, ξ_{1,τ_1} is set to a random string distinct from those already present in $\mathcal{L}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$ and $\mathcal{C}$ provides $\mathcal{A}$ with ξ_{1,τ_1}. Note that the (total) degree of the polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ is at most one.

If $\mathcal{A}$ queries $\mathcal{O}$ with an encoding ξ not previously output by the oracle, then $\mathcal{A}$ increments the counter $\tau_1 := \tau_1 + 1$, sets $\xi_{1,\tau_1} := \xi$, and sets $\mathsf{F}_{1,\tau_1} := \mathsf{U}_{\tau_1}$. The pair $(\mathsf{F}_{1,\tau_1}, \xi_{1,\tau_1})$ is appended to $\mathcal{L}$. This step is carried out for each guessed operand.

Group Operation of $\mathbb{G}_T$: The group oracle $\mathcal{O}_T$ is modeled similar to $\mathcal{O}$, instead appropriately updating the counter τ_T, and appending the list $\mathcal{L}_T$ with the output $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$. $\mathcal{C}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. For guessed operands in $\mathbb{G}_T$, a new variable V_{τ_T} is introduced instead.

Pairing Operation: For a pairing operation, $\mathcal{A}$ queries $\mathcal{C}$ with two operands $\xi_{1,i}, \xi_{1,j}$ $(1 \leq i, j \leq \tau_1)$ in $\mathcal{L}$. $\mathcal{C}$ first increments $\tau_T := \tau_T + 1$, and then computes the polynomial $\mathsf{F}_{T,\tau_T} := \mathsf{F}_{1,i} \cdot \mathsf{F}_{1,j}$. Again, if $\mathsf{F}_{T,\tau_T} = \mathsf{F}_{T,k}$ for some $k < \tau_T$, then $\mathcal{C}$ sets $\xi_{T,\tau_T} := \xi_{T,k}$. Otherwise, ξ_{T,τ_T} is set to a random string distinct from those already present in $\mathcal{L}_T$. The pair $(\mathsf{F}_{T,\tau_T}, \xi_{T,\tau_T})$ is appended to $\mathcal{L}_T$, and $\mathcal{C}$ provides $\mathcal{A}$ with ξ_{T,τ_T}. Note that the degree of the polynomials $\mathsf{F}_{T,i}$ in $\mathcal{L}_T$ is at most two.

Decryption: $\mathcal{C}$ answers decryption queries by $\mathcal{A}$ in the normal way by calling the pairing oracle $\mathcal{O}_e$, correspondingly updating the list $\mathcal{L}_T$, and by providing $\mathcal{A}$ the corresponding encoding in Ξ_T.

Challenge: $\mathcal{C}$ chooses a random bit $b \xleftarrow{\$} \{0, 1\}$. $\mathcal{C}$ adds the the polynomial R and gives $\mathcal{A}$ the corresponding random distinct encoding in Ξ. If $b = 0$, $\mathcal{C}$ adds the polynomial XR to $\mathcal{L}_T$, else it adds a new polynomial V_{τ_T} (after incrementing τ_T) to $\mathcal{L}_T$. $\mathcal{A}$ is also given the corresponding encoding in Ξ_T.

End of Game $\mathcal{G}$**:** When $\mathcal{A}$ terminates it outputs a guess b' of b. Next, $\mathcal{C}$ chooses random values x, r, $\{u\}$, $\{v\}$, $\{r\} \leftarrow \mathbb{Z}_q$ for the indeterminates X, R, $\{\mathsf{U}\}$, $\{\mathsf{V}\}$, respectively. Then it evaluates the polynomials in lists $\mathcal{L}$ and $\mathcal{L}_T$.

Note that the adversary $\mathcal{A}$ will not be able to compute the polynomial XR from polynomials in $\mathcal{L}$ and $\mathcal{L}_T$ if XR was not given to it in the challenge step. $\mathcal{A}$ is said to have won the game $\mathcal{G}$ if:

1. $F_{1,i}(x, r, \{u\}) = F_{1,j}(x, r, \{u\})$ in $\mathbb{Z}_q$, for some two polynomials $F_{1,i} \neq F_{1,j}$ in $\mathcal{L}$.
2. $F_{T,i}(x, r, \{u\}, \{v\}) = F_{T,j}(x, r, \{u\}, \{v\})$ in $\mathbb{Z}_q$, for some two polynomials $F_{T,i} \neq F_{T,j}$ in $\mathcal{L}_T$.
3. $b' = b$.

This completes the description of the game $\mathcal{G}$ and simulator $\mathcal{C}$.

Let $\mathsf{Collision}$ denote either of the events 1 and 2 above, i.e. a *collision* occurring in lists $\mathcal{L}$ and/or $\mathcal{L}_T$. Denote the event 3 above by $\mathsf{Success}$.

Analysis of $\mathbf{Pr}^{\mathbf{CCA1}}_{\mathcal{A},\mathsf{BEG}}$: The success probability $\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}}$ of $\mathcal{A}$ in the actual CCA1 game satisfies

$$\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}} \leq \Pr\left[\mathsf{Success} \mid \overline{\mathsf{Collision}}\right] + \Pr\left[\mathsf{Collision}\right]. \tag{4}$$

This is because the event $\overline{\mathsf{Collision}}$ ensures that $\mathcal{A}$ will get to see only distinct group elements in the actual interaction. In other words, $\mathcal{A}$ is unable to cause *collisions* among group elements. As long as the event $\mathsf{Collision}$ does not occur, then the view of $\mathcal{A}$ is identical in the game $\mathcal{G}$ and the actual interaction. Hence if $\mathcal{A}$ is unable to provoke collisions, then adaptive strategies are no more powerful than non-adaptive ones (see [22, Lemma 2 on pp. 12], also [32]). This observation allows us to choose group elements and their representations independently of the strategy of $\mathcal{A}$.

First we bound $\Pr\left[\mathsf{Collision}\right]$. The τ_1 polynomials $\mathsf{F}_{1,i}$ in $\mathcal{L}$ have degree at most one. Note that $\mathsf{F}_{1,i} \neq \mathsf{F}_{1,j} \Leftrightarrow \mathsf{F}_{1,i} - \mathsf{F}_{1,j} \neq 0$ as polynomials. From Lemma 2 (with $\lambda' = 0$), the probability that two distinct polynomials in $\mathcal{L}$ evaluate to the same value for randomly and independently chosen values for the indeterminates is at most $\frac{1}{q}$. Summing up over at most $\binom{\tau_1}{2}$ distinct pairs (i,j), the probability that the condition 1 above holds is at most $\binom{\tau_1}{2} \cdot \frac{1}{q}$. Similarly, the probability that the condition 2 above holds is at most $\binom{\tau_T}{2} \cdot \frac{2}{q}$. Since $\mathcal{A}$ makes at most $s_{\mathcal{O}} < s$ group oracle queries and that in each query $\mathcal{A}$ can guess at most two new elements, it is easy to see that lists $\mathcal{L}$ and $\mathcal{L}_T$ together have at most $3(s_{\mathcal{O}} + s_D) \leq 3s$ elements. Hence we obtain

$$\Pr\left[\mathsf{Collision}\right] \leq \binom{\tau_1}{2} \cdot \frac{1}{q} + \binom{\tau_T}{2} \cdot \frac{2}{q} \leq \frac{1}{q}(\tau_1 + \tau_T)^2 \leq \frac{9s^2}{q}. \tag{5}$$

Next, to bound $\Pr\left[\mathsf{Success} \mid \overline{\mathsf{Collision}}\right]$, we note that the adversary $\mathcal{A}$ will not be able to compute the polynomials XR or R from polynomials in $\mathcal{L}$ and $\mathcal{L}_T$ if XR or R was not given to it in the challenge step. Hence the event of no collision ensures that $\mathcal{A}$ will not be able to compute the representation of the element corresponding to XR or R. Hence $\Pr\left[\mathsf{Success} \mid \overline{\mathsf{Collision}}\right] = \frac{1}{2}$. Therefore, from (5) and (4), we get

$$\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}} \leq \frac{1}{2} + \frac{9s^2}{q}. \tag{6}$$

Hence if $s = \mathrm{poly}(\log q)$, then $\Pr^{\mathrm{CCA1}}_{\mathcal{A},\mathsf{BEG}}$ is negligible. This completes the proof of Theorem 3. □

A.2 Leakage Setting: Completing Proof of Theorem 2.

Let us first briefly sketch the main ideas of the proof. Working on the lines of (4), the advantage of $\mathcal{A}$ is bounded by its success probabilities conditioned on the

event whether or not a collision has occurred in the lists consisting of elements of $\mathbb{G}$ and $\mathbb{G}_T$. It is important to note that the proof for the non-leakage setting (i.e. proof of Theorem 3) and the leakage setting would be the same conditioned on the fact that a collision has not occurred, and that the leakage functions will not be able to compute the "polynomial X" corresponding to the secret key nor guess the correct representations of the group elements for which it only partially obtains information through the leakage functions. The reason is that in the event of no collision, the adversary gets to see only distinct group elements and hence it will not have enough information on the relation amongst the group elements it can compute. The fact that the leakage functions cannot compute the full secret key shows that the adversary will never be able to continually leak the whole of the secret key. Hence leakage on the secret state will not be useful in this case. Hence the success probability of $\mathcal{A}$ is the same in the event of no collision (that includes the event of guessing the representations of group elements using partial information about them).

However the probability that a collision occurs in the leakage setting is increased by a factor of at most $2^{2\lambda}$. This is because when $\mathcal{A}$ has access to leakage output $f_i(\sigma_{i-1}, t_i)$ and $h_i(\sigma'_{i-1}, (t_i, Y_i))$ during i^{th} decryption query, then in adversary's view the parameters t_i $(i \geq 1)$ are no longer uniformly distributed even though they are still independent. Hence $\mathcal{A}$ can now cause collisions among polynomials (in Conditions 1-2 on page 19) with increased probability. Since t_i appears in both $f_i()$ and $h_i()$, its (average conditional) min-entropy will be reduced by at most 2λ bits.

The only useful information that the leakage functions can provide to $\mathcal{A}$ is about the secret key X. This is because the values t_i are independent of the derived shared secret key. However $\mathcal{A}$ can use the leakages of t_i to eventually leak X. If $\mathcal{A}$ is able to compute X, then it can trivially compute the symmetric key corresponding to the challenge ciphertext. The event of no collision, and the fact that X is not a "linear combination" of the inputs to the leakage functions, guarantees that $\mathcal{A}$ is unable to compute X. Note that because the representations of group elements in the GBG model are randomized, the probability of guessing the complete representations of each of σ_{i-1}, σ'_{i-1} and Y_i, given the leakages, is increased by a factor of at most $2^{2\lambda}$.

Proof. Let $\mathcal{A}$ be an s-query adversary that can break the security of the scheme BEG. Hence $\mathcal{A}$ can make totally at most s group oracle and pairing oracle queries $(s_{\mathcal{O}})$ and decryption oracle queries (s_D). In the count of s, even group oracle queries by leakage functions f_i, h_i $(i \geq 1)$ specified by $\mathcal{A}$ are also included. Let the adversary $\mathcal{A}$ play the game $\mathcal{G}'$ described below. This game is an extension of game $\mathcal{G}$ described in the proof of Theorem 3. To avoid repetition, we only describe here the extensions that are not part of game $\mathcal{G}$. Let $\{T\}$ denote the list of indeterminates $\{T_i : 1 \leq i \leq s_D\}$ that correspond to the values t_i in BEG.

Game $\mathcal{G}'$: For each leakage function $f_i(\sigma_{i-1}, t_i)$ and $h_i(\sigma'_{i-1}, (t_i, Y_i))$, $\mathcal{A}$ maintains a pair of lists $\left(\mathcal{L}^{f_i}, \mathcal{L}_T^{f_i}\right)$ and $\left(\mathcal{L}^{h_i}, \mathcal{L}_T^{h_i}\right)$, respectively. These lists contain polynomial and bit-string pairs. The polynomials in $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ belong to

$\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{T}\}]$, and the corresponding bit-strings are from the encoding set Ξ of group $\mathbb{G}$. The polynomials in $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ are in the ring $\mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{T}\}]$, and the corresponding bit-strings are from the encoding set Ξ_T of group $\mathbb{G}_T$. Intuitively, the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ correspond to the elements of group $\mathbb{G}$ that can be computed by f_i and h_i, respectively, whereas the lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ correspond to the elements of $\mathbb{G}_T$.

Every polynomial in $\mathcal{L}^{f_i}$ is of the form $c_{1,i}\mathsf{T}_i + c_{2,i}\sum_{j=0}^{i-1}\mathsf{T}_j + c_{3,i}\mathsf{D}_i$, where $c_{1,i}$, $c_{2,i}$, $c_{3,i} \in \mathbb{Z}_q$ are chosen by $\mathcal{A}$ and $\mathsf{D}_i \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$ is in $\mathcal{L}$ (cf. (2)). Every polynomial in $\mathcal{L}^{h_i}$ is of the form

$$d_{1,i}\mathsf{T}_i + d_{2,i}\left(\mathsf{X} - \sum_{j=0}^{i-1}\mathsf{T}_j\right) + d_{4,i}\mathsf{W}_i, \tag{7}$$

where $d_{1,i}$, $d_{2,i}$, $d_{3,i}$, $d_{4,i} \in \mathbb{Z}_q$ are also chosen by $\mathcal{A}$ and $\mathsf{W}_i \in \mathbb{Z}_q[\mathsf{X}, \mathsf{R}, \{\mathsf{U}\}]$ is in the list $\mathcal{L}$. Note that the polynomials in lists $\mathcal{L}^{f_i}$ and $\mathcal{L}^{h_i}$ are of degree at most one, and that they do not contain the monomial X. The polynomials in lists $\mathcal{L}_T^{f_i}$ and $\mathcal{L}_T^{h_i}$ are of degree at most two.

The game $\mathcal{G}'$ proceeds exactly as game $\mathcal{G}$ except that $\mathcal{A}$ can also obtain leakage through functions f_i and h_i in the i^{th} decryption query. The leakages on the representations of the group elements are simulated in the naive way, whereas for the leakages on t_i, a temporary random value is chosen for each t_i and the leakage on this value is given to the adversary. When $\mathcal{A}$ terminates it outputs a guess b' of b. Let us denote by $\mathsf{Success}^*$ the event of successful guess of the bit b by $\mathcal{A}$. Let $\mathsf{Collision}^*$ denote the event of a collision occurring in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \leq i \leq s_D)$ and also the event of successful guessing of the partially leaked representations. The polynomials are now evaluated with values chosen from independent distributions with min-entropy $\log q - 2\lambda$, not necessarily from an uniform distribution. The exact distribution depends on the leakage functions chosen by $\mathcal{A}$. Since we are only interested to upper bound the collision probability, we can safely assume that the simulator chooses the right distribution. Note that even in the leakage setting, adaptive strategies are no more powerful than non-adaptive ones, as observed in [1, pp. 691]. This completes the description of the game $\mathcal{G}'$.

Let $\Pr^{\text{CCmLA1}}_{\mathcal{A},\mathsf{BEG}}$ denote the probability of the event $\mathsf{Success}^*$. On the lines of (4), we can write

$$\Pr^{\text{CCmLA1}}_{\mathcal{A},\mathsf{BEG}} \;\leq\; \Pr\left[\mathsf{Success}^* \mid \overline{\mathsf{Collision}^*}\right] + \Pr\left[\mathsf{Collision}^*\right]. \tag{8}$$

As mentioned before, conditioned on the event $\overline{\mathsf{Collision}^*}$, the view of the adversary $\mathcal{A}$ will be same in both the games $\mathcal{G}'$ and $\mathcal{G}$. This is because in both the cases $\mathcal{A}$ will get to see only distinct group elements. Also, we are conditioning on the event that $\mathcal{A}$ will not be able to guess the correct representations of any of the at most $2\lambda s$ group elements it obtains through the leakage functions. Hence,

on the lines of (6), we have

$$\Pr\left[\mathsf{Success}^* \mid \overline{\mathsf{Collision}^*}\right] \;=\; \frac{1}{2}. \tag{9}$$

Lemma 3. $\Pr\left[\mathsf{Collision}^*\right] \leq \left(\frac{9s^2+2\lambda s}{q}\right) 2^{2\lambda}$.

Proof. To compute the required probability, the polynomials in lists $\mathcal{L}$, $\mathcal{L}_T$, $\mathcal{L}^{f_i}$, $\mathcal{L}^{h_i}$, $\mathcal{L}_T^{f_i}$, $\mathcal{L}_T^{h_i}$ $(1 \leq i \leq s_D)$ are evaluated by choosing values from $\mathbb{Z}_q$ according to (independent) distributions with min-entropy at least $\log q - 2\lambda$. This is because $\mathcal{A}$ can obtain at most 2λ bits of leakage about t_i $(i = 1, \ldots, s_D)$. According to Lemma 1, the values t_i have min-entropy at least $\log q - 2\lambda$ in the view of $\mathcal{A}$. The total length of all the lists is at most $3(\tau_1 + \tau_T) \leq 3s)$ (c.f. 3). Working exactly on the lines of (5), and using Lemma 2 (with $\lambda' = 2\lambda$), we obtain this probability as $\frac{9s^2}{q}2^{2\lambda}$. The probability of the event that $\mathcal{A}$ will guess the complete representations of any of the at most $3s$ group elements, for which it can possibly obtain partial information on their representations through the leakage functions, is at most $\frac{3s \cdot 2^{2\lambda}}{q}$. Hence $\Pr\left[\mathsf{Collision}^*\right] \leq \left(\frac{9s^2+3s}{q}\right) 2^{2\lambda}$. □

From (8), (9) and Lemma 3, we have $\Pr^{\mathrm{CCmLA1}}_{\mathcal{A},\mathsf{BEG}} \leq \frac{1}{2} + \left(\frac{9s^2+3s}{q}\right) 2^{2\lambda}$. This completes the proof of Theorem 2. □