Distributed Private-Key Generators for Identity-Based Cryptography[⋆]

Aniket Kate and Ian Goldberg

Cheriton School of Computer Science
University of Waterloo, Canada
{akate,iang}@cs.uwaterloo.ca

Abstract. An identity-based encryption (IBE) scheme can greatly reduce the complexity of sending encrypted messages. However, an IBE scheme necessarily requires a private-key generator (PKG), which can create private keys for clients, and so can passively eavesdrop on all encrypted communications. Although a distributed PKG has been suggested as a way to mitigate this key escrow problem for Boneh and Franklin's IBE scheme, the security of this distributed protocol has not been proven. Further, a distributed PKG has not been considered for any other IBE scheme.

In this paper, we design distributed PKG setup and private key extraction protocols for three important IBE schemes; namely, Boneh and Franklin's BF-IBE, Sakai and Kasahara's SK-IBE, and Boneh and Boyen's BB_1-IBE. We give special attention to the applicability of our protocols to all possible types of bilinear pairings and prove their IND-ID-CCA security in the random oracle model against a Byzantine adversary. Finally, we also perform a comparative analysis of these protocols and present recommendations for their use.

1 Introduction

In 1984, Shamir [2] introduced the notion of identity-based cryptography (IBC) as an approach to simplify public-key and certificate management in a public-key infrastructure (PKI) and presented an open problem to provide an identity-based encryption (IBE) scheme. After seventeen years, Boneh and Franklin [3] proposed the first practical and secure IBE scheme (BF-IBE) using bilinear maps. After this seminal work, in the last few years, significant progress has been made in IBC (for details, refer to a recent book on IBC [4] and references therein).

In an IBC system, a client chooses an arbitrary string such as her e-mail address to be her public key. With a standardized public-key string format, an IBC scheme completely eliminates the need for public-key certificates. As an example, in an IBE scheme, a sender can encrypt a message for a receiver knowing just the identity of the receiver and importantly, without obtaining and verifying the receiver's public-key certificate. Naturally, in such a system, a client herself is not capable of generating a private key for her identity. There is a trusted party called a *private-key generator* (PKG) which performs the system setup, generates a secret called the *master key* and

[⋆] An extended version of this paper is avaiable [1].

provides private keys to clients using it. As the PKG computes a private key for a client, it can decrypt all of her messages passively. This inherent *key escrow* property asks for complete trust in the PKG, which is difficult to find in many realistic scenarios.

Importantly, the amount of trust placed in the holder of an IBC master key is far greater than that placed in the holder of the private key of a certifying authority (CA) in a PKI. In a PKI, in order to attack a client, the CA has to actively generate a fake certificate for the client containing a fake public key. In this case, it is often possible for the client to detect and prove the malicious behaviour of the CA. The CA cannot perform any passive attack; specifically, it cannot decrypt a message encrypted for the client using a client-generated public key and it cannot sign some document for the client, if the verifier gets a correct certificate from the client. On the other hand, in IBC, 1) knowing the master key, the PKG can decrypt or sign the messages for any client, without any active attack and consequent detection, 2) the PKG can make clients' private keys public without any possible detection, and 3) in a validity-period-based key revocation system [3], bringing down the PKG is sufficient to bring the system to a complete halt (*single point of failure*), once the current validity period ends. Therefore, the PKG in IBC needs to be far more trusted than the CA in a PKI. This has been considered as a reason for the slow adoption of IBC schemes outside of closed organizational settings.

Boneh and Franklin [3] suggest distributing a PKG in their BF-IBE scheme to solve these problems. In an *(n, t)-distributed PKG*, the master key is distributed among n PKG nodes such that a set of nodes of size t or smaller cannot compute the master key, while a client extracts her private key by obtaining private-key shares from any $t + 1$ or more nodes; she can then use the system's public key to verify the correctness of her thus-extracted key. Boneh and Franklin [3] propose *verifiable secret sharing* (VSS) [5] of the master key among multiple PKGs to design a distributed PKG and also hint towards a completely distributed approach using the distributed (shared) key generation (DKG) schemes of Gennaro et al. [6]; however, they do not provide a formal security model and a proof. Further, none of the IBE schemes defined after [3] consider the design of a distributed PKG.

Although various proposed practical applications using IBE, such as pairing-based onion routing [7] or verifiable random functions from identity-based key encapsulation [8], require a distributed PKG as a fundamental need, there is no distributed PKG available for use yet. This practical need forms the motivation of this work.

Related Work. Although we are defining protocols for IBE schemes, as we are concentrating on distributed cryptographic protocols and due to space constraints, we do not include a comprehensive account of IBE. We refer readers to [9] for a detailed discussion on the various IBE schemes and frameworks defined in the literature. Pursuant to this survey, we work in the random oracle model for efficiency and practicality reasons.

None of the IBE schemes except BF-IBE considered distributed PKG setup and key extraction protocols in their design. Recently, Geisler and Smart [10] defined a distributed PKG for Sakai and Kasahara's SK-IBE [11]; however, their solution against a Byzantine adversary has an exponential communication complexity and a formal security proof is also not provided. We overcome both of these barriers in our distributed PKG for SK-IBE: our scheme is secure against a Byzantine adversary and has the same polynomial-time communication complexity as their scheme, which is secure only against an honest-but-curious adversary; we also provide a formal security proof.

Other than [10], there have been a few other efforts in the literature to counter the inherent key escrow and single point of failure issues in IBE. Al-Riyami and Paterson [12] introduce *certificateless public-key cryptography* (CL-PKC) to address the key escrow problem by combining IBC with public-key cryptography. Their elegant approach, however, does not address the single point of failure problem. Although it is possible to solve the problem by distributing their PKG using a VSS (which employs a trusted dealer to generate and distribute the key shares), which is inherently cheaper than a DKG-based PKG by a linear factor, it is impossible to stop a dealer's active attacks without completely distributed master-key generation. Further, as private-key extractions are less frequent than encryptions, it is certainly advisable to use more efficient options during encryption rather than private-key extraction. Finally, with the requirement of online access to the receiver's public key, CL-PKC becomes ineffective for systems without continuous network access, where IBC is considered to be an important tool. Lee et al. [13] and Gangishetti et al. [14] propose variants of the distributed PKG involving a more trustworthy key generation centre (KGC) and other key privacy authorities (KPAs). As observed by Chunxiang et al. [15] for [13], these approaches are, in general, vulnerable to passive attack by the KGC. In addition, the trust guarantees required by a KGC can be unattainable in practice. Goyal [16] reduces the required trust in the PKG by restricting its ability to distribute a client's private key. This does not solve the problem of single point of failure. Further, the PKG in his system still can decrypt the clients' messages passively, which leaves a secure and practical implementation of distributed PKGs wanting.

Threshold versions of signature schemes obtained from some IBE schemes using the Naor transform have been proposed and proved previously [17, 18]. However, these solutions do not work for the corresponding IBE scheme. This is due to the inherent secret nature of a client's private keys and corresponding shares as compared to the inherent public nature of signatures and corresponding signature shares. While designing IBE schemes with a distributed PKG, we have to make sure that a PKG node cannot derive more information than the private-key share it generates for a client and that private-key shares are not available in public as commitments.

Our Contributions. We present distributed PKGs for all three important IBE frameworks: namely, full-domain-hash IBEs, exponent-inversion IBEs and commutative-blinding IBEs [9]. We propose distributed PKG setups and distributed private-key extraction protocols for BF-IBE [3], SK-IBE [11], and Boneh and Boyen's (modified) BB_1-IBE [9, 19] schemes. The novelty of our protocols lies in achieving the secrecy of a client private key from the generating PKG nodes without compromising the efficiency. We realize this with an appropriate use of non-interactive proofs of knowledge, pairing-based verifications, and DKG protocols with and without the uniform randomness property. Based on the choice of the DKG protocol, our distributed PKGs can work in the synchronous or asynchronous communication model. In terms of feasibility, we ensure that our protocols work for all three pairing types defined by Galbraith et al. [20].

We prove the adaptive chosen ciphertext security (IND-ID-CCA) of the defined schemes in the random oracle model. Interestingly, compared to the security proofs for the respective IBE schemes with a single PKG, there are no additional security reduction factors in our proofs, even though the underlying DKG protocol used in the distributed PKGs does not provide a guarantee about the uniform randomness for the generated master secrets. To the best of our knowledge, there is no threshold cryptographic protocol available in the literature where a similar tight security reduction has been proven while using a DKG without the (more expensive) uniform randomness property. Finally, using operation counts, key sizes, and possible pairing types, we compare the performance of three distributed PKGs we define.

2 Preliminaries

2.1 Cryptographic Background

Bilinear Pairings. For three cyclic groups $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_T$ (all of which we shall write multiplicatively) of the same prime order p, an *admissible bilinear pairing* e is a map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ with the *bilinearity*, *non-degeneracy* and *admissibility* properties. For a detailed mathematical discussion of bilinear pairings refer to [21]. We consider all three types of pairings [20] for prime order groups: namely, type 1, 2, and 3. In *type* 1 or *symmetric* pairings, an isomorphism $\phi : \hat{\mathbb{G}} \to \mathbb{G}$ as well as its inverse ϕ^{-1} are efficiently computable. In *type* 2 pairings, only the isomorphism ϕ, but not ϕ^{-1}, is efficiently computable. In *type* 3 pairings, neither ϕ nor ϕ^{-1} can be efficiently computed. The efficiency of the pairing computation improves from type 1 to type 2 to type 3 pairings. For a detailed discussion of the performance aspects of pairings refer to [20, 22].

Non-interactive Proofs of Knowledge. As we assume the random oracle model in the paper, we can use non-interactive zero-knowledge proofs of knowledge (NIZKPK) based on the Fiat-Shamir methodology [23]. In particular, we use a variant of NIZKPK of a discrete logarithm (DLog) and one for proof of equality of two DLogs.

We employ a variant of NIZKPK of a DLog where given a DLog commitment ($\mathcal{C}_{\langle g\rangle}(s) = g^s$) and a Pedersen commitment [24] ($\mathcal{C}_{\langle g,h\rangle}(s,r) = g^s h^r$) to the same value s for generators $g, h \in \mathbb{G}$ and $s, r \in \mathbb{Z}_p$, a prover proves that she knows s and r such that $\mathcal{C}_{\langle g\rangle}(s)$ and $\mathcal{C}_{\langle g,h\rangle}(s,r)$. We denote this proof as

$$\text{NIZKPK}_{\equiv Com}(s, r, \mathcal{C}_{\langle g\rangle}(s), \mathcal{C}_{\langle g,h\rangle}(s,r)) = \pi_{\equiv Com} \in \mathbb{Z}_p^3. \qquad (1)$$

It is nearly equivalent to proving knowledge of two DLogs separately.

We use another NIZKPK (proof of equality) of discrete logs [25] such that given commitments $\mathcal{C}_{\langle g\rangle}(s) = g^s$ and $\mathcal{C}_{\langle h\rangle}(s) = h^s$, a prover proves equality of the associated DLogs. We denote this proof as

$$\text{NIZKPK}_{\equiv DLog}(s, \mathcal{C}_{\langle g\rangle}(s), \mathcal{C}_{\langle h\rangle}(s)) = \pi_{\equiv DLog} \in \mathbb{Z}_p^2. \qquad (2)$$

Note that g and h can belong two different groups of the same order. Refer to the extended version of the paper [1] for the descriptions of the above proofs.

There exists an easier way to prove this equality of DLogs if a pairing between the groups generated by g and h is available. Using a method due to Joux and Nguyen [26] to solve the decisional Diffie-Hellman (DDH) problem over pairing-friendly groups, given g^x and $h^{x'}$ the verifier checks if $e(g, h^{x'}) \overset{?}{=} e(g^x, h)$. However, when using a type 3 pairing, in the absence of an efficient isomorphism between $\mathbb{G}$ and $\hat{\mathbb{G}}$, if both g and h belong to the same group then the pairing-based scheme does not work. NIZKPK$_{\equiv DLog}$ provides a completely practical alternative there.

2.2 Assumptions

System Assumptions. Except for the steps involving DKG in some form, all other steps in our distributed PKG protocols are independent of the communication model used. As distributedness of PKG is important in IBC outside closed organizational settings, we suggest the asynchronous communication model as it closely models the Internet. In particular, we follow the system model of the DKG protocol in [27]. In a synchronous communication network, it is straightforward to replace this asynchronous DKG with a more efficient protocol such as the Joint Feldman DKG (JF-DKG) [28].

We assume a standard t-Byzantine adversary in a system with $n \geq 3t + 1$ nodes $P_1, P_2, \ldots, P_n$, where any t nodes are compromised or crashed by the adversary. In the synchronous communication model, the above resiliency bound becomes $n \geq 2t + 1$. Further, when the communication model is synchronous, we assume a *rushing* adversary. It can wait for the messages of the uncorrupted players to be transmitted, then decide on its computation and communication for that round, and still get its messages delivered to the honest parties on time. The adversary is also *static* as all of the efficient VSS and DKG schemes that we use are proved secure only against a static adversary, which can choose its t compromisable nodes before a protocol run. They are not considered secure against an adaptive adversary because their security proofs do not go through when the adversary can corrupt nodes adaptively. [28, §4.4] Canetti et al. [29] presented a DKG scheme provably secure against adaptive adversaries with at least two more communication rounds as compared to JF-DKG. Due to the inefficiency of adaptive (provably) secure DKG protocols, we stick to protocols provably secure only against a static adversary. However, it possible to easily use the DKG protocol in [29] and obtain security against the adaptive adversary.

Cryptographic Assumptions. Our adversary is computationally bounded with a security parameter κ. We assume an instance of a pairing framework e of groups $\mathbb{G}$, $\hat{\mathbb{G}}$ and $\mathbb{G}_T$, whose common prime order p is such that the adversary has to perform 2^κ operations to break the system. Let $\mathcal{G} = \langle e, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T \rangle$. Following [9], we work in the random oracle model for efficiency reasons. For the security of the IBE schemes, we use the *bilinear Diffie-Hellman* (BDH) [30] and *bilinear Diffie-Hellman inversion* (BDHI) [31, 32] assumptions. Here, we recall their definitions for asymmetric pairings from [9].

BDH Assumption: Given a tuple $(g, \hat{g}, g^a, \hat{g}^a, g^b, \hat{g}^c)$ in a bilinear group $\mathcal{G}$, the BDH problem is to compute $e(g, \hat{g})^{abc}$. The BDH assumption then states that it is infeasible to solve a random instance of the BDH problem, with non-negligible probability, in time polynomial in the size of the problem instance description.

BDHI Assumption: Given two tuples $(g, g^x, g^{x^2}, \ldots, g^{x^q})$ and $(\hat{g}, \hat{g}^x, \hat{g}^{x^2}, \ldots, \hat{g}^{x^q})$ in a bilinear group $\mathcal{G}$, the q-BDHI problem is to compute $e(g, \hat{g})^{1/x}$. The BDHI assumption for some polynomially bounded q states that it is infeasible to solve a random instance of the q-BDHI problem, with non-negligible probability, in time polynomial in the size of the problem instance description.

2.3 Distributed Computation

We next describe the distributed computation primitives that are required to design our distributed PKGs in an network of n nodes with a t-limited Byzantine adversary. Note that these distributed computation primitives are the efficient versions of the their original forms in [33–35, 28, 36, 27] that utilize the presence of random oracles and the pairing-based DDH problem solving technique [26].

DKG over $\mathbb{Z}_p$**.** Pedersen [24] introduced the concept of DKG and developed a DKG protocol. Unlike VSS, where a dealer chooses a secret and distributes its shares among the nodes, DKG requires *no trusted dealer*. In an (n, t)-*DKG* protocol over $\mathbb{Z}_p$, a set of n nodes generates an element $s \in \mathbb{Z}_p$ in a distributed fashion with its shares $s_i \in \mathbb{Z}_p$ spread over the n nodes such that any subset of size greater than a threshold t can reveal or use the shared secret, while smaller subsets cannot. We mandate the following correctness and secrecy properties for a DKG protocol.

Correctness (DKG-C). There exists an efficient algorithm that on input shares from $2t + 1$ nodes and the public information, outputs the same unique value s, even if up to t shares are submitted by malicious nodes.

Secrecy (DKG-S). The adversary with t shares and the public parameters cannot compute the secret s.

In the synchronous and asynchronous communication models, respectively JF-DKG in [28] and the DKG protocol in [27] achieve these properties and are suitable for our use. For ease of exposition, we avoid crash-recoveries used in the DKG protocol in [27].

The shared secret in the above DKG protocols may not be *uniformly* random; this is a direct effect of using only DLog commitments having only computational secrecy. (See [28, §3] for a possible adversary attack.) In many cases, we do not need a uniformly random secret key; the security of these schemes relies on the assumption that the adversary cannot compute the secret. Most of our schemes similarly only require the assumption that it is infeasible to compute the secret given public parameters and we stick with DLog commitments those cases. However, we do indeed need a uniformly random shared secret in few protocols. We mandate the following stronger correctness and secrecy properties based on the DKG correctness and secrecy defined in [28, §4.1].

Strong Correctness (DKG-sC). Along with the DKG-C property, s is now uniformly distributed in $\mathbb{Z}_n$.

Strong Secrecy (DKG-sS). No information about s can be learnt by the adversary except for what is implied by the public parameters.

In this case, we use Pedersen commitments, but we do not employ the methodology defined by Gennaro et al. [6], which increases the number of rounds in the protocol. We observe that with the random oracle assumption at our disposal, the communicationally demanding technique by Gennaro et al. can be replaced with the much simpler computational non-interactive zero-knowledge proof of equality of committed values

NIZKPK$_{\equiv Com}$ described in Eq. 1. The simulator-based proof for the above is similar to that in [28, §4.3] and is included in [1]. We represent DKG protocols using the DLog and Pedersen commitments as $\mathsf{DKG}_{\mathsf{DLog}}$ and $\mathsf{DKG}_{\mathsf{Ped}}$ respectively. For node P_i,

$$\left(\mathcal{C}^{(s)}_{\langle g\rangle}, s_i\right) = \mathsf{DKG}_{\mathsf{DLog}}(n, t, \tilde{t}, g, \alpha_i) \tag{3}$$

$$\left(\mathcal{C}^{(s,s')}_{\langle g,h\rangle}, [\mathcal{C}^{(s)}_{\langle g\rangle}, \text{NIZKPK}_{\equiv Com}], s_i, s'_i\right) = \mathsf{DKG}_{\mathsf{Ped}}(n, t, \tilde{t}, g, h, \alpha_i, \alpha'_i) \tag{4}$$

Here, $\tilde{t}$ is the number of VSS instances to be chosen ($t < \tilde{t} \leq 2t+1$), $g, h \in \mathbb{G}$ are commitment generators and $\alpha_i, \alpha'_i \in \mathbb{Z}_p$ are respectively a secret and randomness shared by P_i. For $\psi, \psi' \in \mathbb{Z}_p[x]$ of degree t with $\psi(0) = s$ and $\psi'(0) = s'$, $\mathcal{C}^{(s)}_{\langle g\rangle} = [g^s, g^{\psi(1)}, \cdots, g^{\psi(n)}]$ and $\mathcal{C}^{(s,s')}_{\langle g,h\rangle} = [g^s h^{s'}, g^{\psi(1)} h^{\psi'(1)}, \cdots, g^{\psi(n)} h^{\psi'(n)}]$ are respectively DLog and Pedersen commitment vectors. The optional NIZKPK$_{\equiv Com}$ is a vector of proofs that the entries of $\mathcal{C}^{(s)}_{\langle g\rangle}$ and $\mathcal{C}^{(s,s')}_{\langle g,h\rangle}$ commit to the same values.

In the most basic form of DKG, nodes generate shares of a secret z chosen jointly at random from $\mathbb{Z}_p$. Here, every node generates a random $r_i \in \mathbb{Z}_p$ and shares that using the DKG protocol with DLog or Pedersen commitments as $\mathsf{DKG}(n, t, \tilde{t} = t+1, g, [h], r_i, [r'_i])$ where the generator h and randomness r'_i are only required if Pedersen commitments are used. We represent the corresponding protocols as follows:

$$\left(\mathcal{C}^{(z)}_{\langle g\rangle}, z_i\right) = \mathsf{Random}_{\mathsf{DLog}}(n, t, g) \tag{5}$$

$$\left(\mathcal{C}^{(z,z')}_{\langle g,h\rangle}, [\mathcal{C}^{(z)}_{\langle g\rangle}, \text{NIZKPK}_{\equiv Com}], z_i, z'_i\right) = \mathsf{Random}_{\mathsf{Ped}}(n, t, g, h). \tag{6}$$

Distributed Addition over $\mathbb{Z}_p$**.** Let $\alpha, \beta \in \mathbb{Z}_p$ be two secrets shared among n nodes using the DKG protocol. Let polynomials $f(x), g(x) \in \mathbb{Z}_p[x]$ be the respectively associated degree-t polynomials and let $c \in \mathbb{Z}_p$ be a non-zero constant. Due to the linearity of Shamir's secret sharing [37], a node P_i with shares α_i and β_i can locally generate shares of $\alpha+\beta$ and $c\alpha$ by computing $\alpha_i+\beta_i$ and $c\alpha_i$, where $f(x)+g(x)$ and $cf(x)$ are the respective polynomials. $f(x)+g(x)$ is random if either one of $f(x)$ or $g(x)$ is, and $cf(x)$ is random if $f(x)$ is. Commitment entries for the resultant shares respectively are $\left(\mathcal{C}^{(\alpha+\beta)}_{\langle g\rangle}\right)_i = \left(\mathcal{C}^{(\alpha)}_{\langle g\rangle}\right)_i \left(\mathcal{C}^{(\beta)}_{\langle g\rangle}\right)_i$ and $\left(\mathcal{C}^{(c\alpha)}_{\langle g\rangle}\right)_i = \left(\mathcal{C}^{(\alpha)}_{\langle g\rangle}\right)^c_i$.

Distributed Multiplication over $\mathbb{Z}_p$**.** Local distributed multiplication of two shared secrets α and β looks unlikely. We use a distributed multiplication protocol against a computational adversary by Gennaro et al. [36, §4]. However, instead of their interactive zero-knowledge proof, we utilize the pairing-based DDH problem solving technique to verify the correctness of the product value shared by a node non-interactively. For shares α_i and β_i with DLog commitments g^{α_i} and $\hat{g}^{\beta_i}$, given a commitment $g^{\alpha_i\beta_i}$ of the shared product, other nodes can verify its correctness by checking if $e(g^{\alpha_i}, \hat{g}^{\beta_i}) \stackrel{?}{=} e(g^{\alpha_i\beta_i}, \hat{g})$ provided the groups of g and $\hat{g}$ are pairing-friendly. We observe that it is also possible to perform this verification when one of the involved commitments is a Pedersen commitment. However, if both commitments are Pedersen commitments, then we have to compute DLog commitments for one of the values and employ NIZKPK$_{\equiv Com}$ to prove its correctness in addition to using the pairing-based verification. In such a case, the choice between the latter technique and the non-interactive version of zero-knowledge proof suggested by Gennaro et al. [36] depends upon implementation efficiencies of the group operation and pairing computations.

In our IBC schemes, we always use the multiplication protocol with at least one DLog commitment. We denote the multiplication protocol involving two DLog commitments as $\mathsf{Mul}_{\mathsf{DLog}}$ and the one involving a combination of the two types of commitments as $\mathsf{Mul}_{\mathsf{Ped}}$. For the protocol correctness, along with recoverability to a unique value (say s), protocol Mul also requires that $s = \alpha\beta$. For the protocol secrecy, along with the secrecy of $\alpha\beta$, the protocol should not provide any additional information about the individual values of α or β once $\alpha\beta$ is reconstructed.

$$\left(\mathcal{C}^{(\alpha\beta)}_{\langle g^*\rangle}, (\alpha\beta)_i\right) = \mathsf{Mul}_{\mathsf{DLog}}(n, t, g^*, \left(\mathcal{C}^{(\alpha)}_{\langle g\rangle}, \alpha_i\right), \left(\mathcal{C}^{(\beta)}_{\langle \hat{g}\rangle}, \beta_i\right)) \tag{7}$$

$$\left(\mathcal{C}^{(\alpha\beta,\alpha\beta')}_{\langle \hat{g},\hat{h}\rangle}, (\alpha\beta)_i, (\alpha\beta')_i\right) = \mathsf{Mul}_{\mathsf{Ped}}(n, t, \hat{g}, \hat{h}, \left(\mathcal{C}^{(\alpha)}_{\langle g\rangle}, \alpha_i\right), \left(\mathcal{C}^{(\beta,\beta')}_{\langle \hat{g},\hat{h}\rangle}, \beta_i, \beta'_i\right)) \tag{8}$$

For $\mathsf{Mul}_{\mathsf{DLog}}$, $g^* = g$ or $\hat{g}$. For $\mathsf{Mul}_{\mathsf{Ped}}$, without loss of generality, we assume that β is distributed with the Pedersen commitment. If instead α uses Pedersen commitment, then the Pedersen commitment groups for $(\alpha\beta)$ change to g and h instead of $\hat{g}$ and $\hat{h}$.

Briefly, the protocol works as follows. Every honest node runs the $\mathsf{DKG}(n, t, 2t + 1, \hat{g}, [\hat{h}], \alpha_i\beta_i, [\alpha_i\beta'_i])$ from Eq. 3 or 4. As discussed above, pairing-based DDH solving is used to verify that the shared value is equal to the product of α_i and β_i.[1] At the end, instead of adding the subshares of the selected VSS instances, every node interpolates them at index 0 to get the new share $(\alpha\beta)_i$ of $\alpha\beta$.

The above Mul protocols can be seamlessly extended for distributed computation of any expression having binary products (BPs). For ℓ shared secrets $x_1, \cdots, x_\ell$, and their DLog commitments $\mathcal{C}^{(x_1)}_{\langle g\rangle}, \cdots, \mathcal{C}^{(x_\ell)}_{\langle g\rangle}$, shares of any binary product $x' = \sum_{i=1}^{m} k_i x_{a_i} x_{b_i}$ with known constants k_i and indices a_i, b_i can be easily computed by extending the protocol in Eq. 7. We denote this generalization as follows.

$$\left(\mathcal{C}^{(x')}_{\langle g^*\rangle}, x'_i\right) = \mathsf{Mul}_{\mathsf{BP}}(n, t, g^*, \{(k_i, a_i, b_i)\}, \left(\mathcal{C}^{(x_1)}_{\langle g\rangle}, (x_1)_i\right), \cdots, \left(\mathcal{C}^{(x_\ell)}_{\langle g\rangle}, (x_\ell)_i\right)) \tag{9}$$

Node P_j shares $\sum_i k_i (x_{a_i})_j (x_{a_i})_j$. For a type 1 pairing, the correctness of the sharing is verified by other nodes as $e(g^{\sum_i k_i (x_{a_i})_j (x_{b_i})_j}, g) \overset{?}{=} \prod_i e((g^{(x_{a_i})_j})^{k_i}, g^{(x_{b_i})_j})$. For type 2 and 3 pairings, NIZKPK$_{\equiv DLog}$ is used to provide DLog commitments to the $(x_{b_i})_j$ with generator $\hat{g}$, and then a pairing computation like the above is used. We use $\mathsf{Mul}_{\mathsf{BP}}$ in Eq. 9 during distributed private-key extraction in the BB$_1$-IBE scheme in §3.5.

Sharing the Inverse of a Shared Secret. Given an (n, t)-distributed secret α, computing shares of its inverse α^{-1} in distributed manner (without reconstructing α) can be done trivially but inefficiently using a distributed computation of α^{p-1}; this involves $O(\log p)$ distributed multiplications. However, using a technique by Bar-Ilan and Beaver [33], this can be done using just one Random and one Mul protocol. This protocol involves interpolation of the product of the secret α with a distributed random element z. If z

[1] For type 3 pairings, a careful selection of commitment generators is required to make the pairing-based verification possible.

is created using DLog commitments and is not uniformly random, the product αz may leak some information about α. We avoid this by using Pedersen commitments while generating z. For a generator g^*, we represent this protocol as follows:

$$\left(\mathcal{C}^{(\alpha^{-1})}_{\langle g^* \rangle}, (\alpha^{-1})_i\right) = \mathsf{Inverse}(n, t, \hat{g}, \hat{h}, \left(\mathcal{C}^{(\alpha)}_{\langle g \rangle}, \alpha_i\right)) \tag{10}$$

The protocol secrecy is the same as that of DKG except it is defined in the terms of α^{-1} instead of α; for the correctness property, along with recoverability to a unique value s, this protocol additionally mandates that $s = \alpha^{-1}$. For a distributed secret $\left(\mathcal{C}^{(\alpha)}_{\langle g \rangle}, \alpha_i\right)$, protocol $\mathsf{Inverse}$ works as follows: Every node P_i runs $\left(\mathcal{C}^{(z,z')}_{\langle \hat{g}, \hat{h} \rangle}, z_i, z'_i\right) = \mathsf{Random}_{\mathsf{Ped}}(n, t, \hat{g}, \hat{h})$ and computes shares of $(w, w') = (\alpha z, \alpha z')$ as $\left(\mathcal{C}^{(w,w')}_{\langle \hat{g}, \hat{h} \rangle}, w_i, w'_i\right) = \mathsf{Mul}_{\mathsf{Ped}}(n, t, \hat{g}, \hat{h}, \left(\mathcal{C}^{(\alpha)}_{\langle g \rangle}, \alpha_i\right), \left(\mathcal{C}^{(z,z')}_{\langle \hat{g}, \hat{h} \rangle}, z_i, z'_i\right))$. It then sends (w_i, w'_i) to each node and interpolates w using the correct received shares. If $w = 0$, repeats the above two steps, else locally computes $(\alpha^{-1})_i = w^{-1} z_i$. Finally, it computes the commitment $\mathcal{C}^{(\alpha^{-1})}_{\langle g^* \rangle}$ using w^{-1}, $\mathcal{C}^{(z,z')}_{\langle \hat{g}, \hat{h} \rangle}$, and if required, any of the NIZKPK techniques. A modified form of this protocol is used in the distributed PKG for SK-IBE in §3.4.

3 Distributed PKG for IBE

We present distributed PKG setup and private key extraction protocols for three IBE schemes: BF-IBE [3], SK-IBE [11], and modified BB_1-IBE [9]. Each of these schemes represents a distinct important category of an IBE classification defined by Boyen [38]. They respectively belong to *full-domain-hash* IBE schemes, *exponent-inversion* IBE schemes, and *commutative-blinding* IBE schemes. The distributed PKG architectures that we develop for each of the three schemes apply to every scheme in their respective categories. Our above choice of IBE schemes is influenced by a recent identity-based cryptography standard (IBCS) [19] and also a comparative study by Boyen [9], which finds the above three schemes to be the most practical IBE schemes in their respective categories. In his classification, Boyen [38] also includes another category for quadratic-residuosity-based IBE schemes; however, none of the known schemes in this category are practical enough to consider here.

The role of a PKG in an IBE scheme ends with a client's private-key extraction and the distributed form of the PKG does not affect the encryption and decryption steps of IBE. Consequently, we define only the distributed PKG setup and private-key extraction steps of the three IBE schemes under consideration. We recall the original encryption and decryption steps in the extended version of the paper [1].

3.1 Bootstrapping Procedure

Each scheme under consideration here requires the following three bootstrapping steps.

1. Determine the node group size n and the security threshold t such that $n \geq 3t + 1$ (the asynchronous case) or $n \geq 2t + 1$ (the synchronous case).

2. Choose the pairing type to be used and compute groups $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_T$ of prime order p such that there exists a pairing e of the decided type with $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$. The security parameter κ determines the group order p.
3. Choose two generators $g \in \mathbb{G}$ and $\hat{g} \in \hat{\mathbb{G}}$ required to generate public parameters as well as the commitments. With a type 1 or 2 pairing, set $g = \phi(\hat{g})$.

Any untrusted entity can perform these offline tasks. Honest DKG nodes can verify the correctness of the tuple (n, t) and confirm the group choices $\mathbb{G}$, $\hat{\mathbb{G}}$, and $\mathbb{G}_T$ as the first step of their distributed PKG setup. If unsatisfied, they may decline to proceed.

3.2 Formal Security Model

An IBE scheme with an (n, t)-distributed PKG consists of the following components:

- A *distributed PKG setup protocol* for node P_i that takes the above bootstrapped parameters n, t and $\mathcal{G}$ as input and outputs a share s_i of a master secret s and a public-key vector K_{pub} of a master public key and n public-key shares.
- A *distributed private key-extraction protocol* for node P_i that takes a client identity $\mathtt{ID}$, the public key vector K_{pub} and the master-secret share s_i as input and outputs a verifiable private-key share $d_{\mathtt{ID}_i}$. The client computes the private key $d_{\mathtt{ID}}$ after verifying the received shares $d_{\mathtt{ID}_i}$.
- An *encryption algorithm* that takes a receiver identity $\mathtt{ID}$, the master public key and a plaintext message M as input and outputs a ciphertext C.
- A *decryption algorithm* for client with identity $\mathtt{ID}$ that takes a ciphertext C and the private key $d_{\mathtt{ID}}$ as input and outputs a plaintext M.

Note that the above distributed PKG setup protocol does not require any *dealer* and that we mandate verifiability for the private-key shares rather than obtaining robustness using error-correcting techniques. During private-key extractions, we insist on minimal interaction between clients and PKG nodes—transferring identity credentials from the client at the start and private-key shares from the nodes at the end.

To define security against an IND-ID-CCA attack, we consider the following game that a challenger plays against a polynomially bounded t-limited Byzantine adversary.

Setup: The adversary chooses to corrupt a fixed set of t nodes and the challenger simulates the remaining $n - t$ nodes to run a distributed PKG setup protocol. At the end of the protocol execution, the adversary receives t shares of a shared master secret for its t nodes and a public key vector K_{pub}. The challenger knows the remaining $n - t$ shares and can derive the master secret as $n - t \geq t + 1$ in any communication setting.

Phase 1: The adversary adaptively issues private-key extraction and decryption queries to the challenger. For a private-key extraction query $\langle \mathtt{ID} \rangle$, the challenger simulates the distributed key extraction protocol for its $n - t$ nodes and sends verifiable private-key shares for its $n - t$ nodes. For a decryption query $\langle \mathtt{ID}, C \rangle$, the challenger decrypts C by generating the private key $d_{\mathtt{ID}}$ or using the master secret.

Challenger: The adversary chooses two equal-length plaintexts M_0 and M_1, and a challenge identity $\mathtt{ID}_{ch}$ such that $\mathtt{ID}_{ch}$ does not appear in any private-key extraction query in Phase 1. The challenger chooses $b \in_R \{0, 1\}$ and encrypts M_b for $\mathtt{ID}_{ch}$ and K_{pub}, and gives the ciphertext C_{ch} to the adversary.

Phase 2: The adversary adaptively issues more private-key extraction and decryption queries to the challenger except for key extraction query for $\langle \mathtt{ID}_{ch} \rangle$ and decryption queries for $\langle \mathtt{ID}_{ch}, C_{ch} \rangle$.
Guess: Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

Security against IND-ID-CCA attacks means that, for any polynomially bounded adversary, $b' = b$ with probability negligibly greater than $1/2$.

3.3 Boneh and Franklin's BF-IBE

BF-IBE [3] belongs to the full-domain-hash IBE family. In a BF-IBE setup, a PKG generates a master key $s \in \mathbb{Z}_p$ and a public key $g^s \in \mathbb{G}$, and derives private keys for clients using their identities and s. A client with identity $\mathtt{ID}$ receives the private key $d_{\mathtt{ID}} = (H_1(\mathtt{ID}))^s = h^s_{\mathtt{ID}} \in \hat{\mathbb{G}}$, where $H_1 : \{0,1\}^* \to \hat{\mathbb{G}}^*$ is a full-domain cryptographic hash function. ($\hat{\mathbb{G}}^*$ denotes the set of all elements in $\hat{\mathbb{G}}$ except the identity.)

Distributed PKG Setup. This involves generation of the system master key and the system public-key tuple in the (n, t)-distributed form among n nodes. Each node P_i participates in a common DKG over $\mathbb{Z}_p$ to generate its share $s_i \in \mathbb{Z}_p$ of the distributed master key s. The system public-key tuple is of the form $\mathcal{C}^{(s)}_{\langle g \rangle} = [g^s, g^{s_1}, \cdots, g^{s_n}]$. We obtain this using our $\mathsf{Random}_{\mathsf{DLog}}$ protocol from Eq. 5 as $\left(\mathcal{C}^{(s)}_{\langle g \rangle}, s_i\right) = \mathsf{Random}_{\mathsf{DLog}}(n, t, g)$.

Private-key Extraction. As a client needs $t+1$ correct shares, it is sufficient for her to contact any $2t+1$ nodes (say set $\mathcal{Q}$). The private-key extraction works as follows.

1. Once a client with identity $\mathtt{ID}$ contacts every node in $\mathcal{Q}$, every honest node $P_i \in \mathcal{Q}$ authenticates the client's identity and returns a private-key share $h^{s_i}_{\mathtt{ID}} \in \hat{\mathbb{G}}$ over a secure and authenticated channel.
2. Upon receiving $t+1$ valid shares, the client can construct her private key $d_{\mathtt{ID}}$ as $d_{\mathtt{ID}} = \prod_{P_i \in \mathcal{Q}} (h^{s_i}_{\mathtt{ID}})^{\lambda_i} \in \hat{\mathbb{G}}$, where the Lagrange coefficient $\lambda_i = \prod_{P_j \in \mathcal{Q} \setminus \{i\}} \frac{j}{j-i}$. The client can verify the correctness of the computed private key $d_{\mathtt{ID}}$ by checking $e(g, d_{\mathtt{ID}}) \overset{?}{=} e(g^s, h_{\mathtt{ID}})$. If unsuccessful, she can verify the correctness of each received $h^{s_i}_{\mathtt{ID}}$ by checking if $e(g, h^{s_i}_{\mathtt{ID}}) \overset{?}{=} e(g^{s_i}, h_{\mathtt{ID}})$. An equality proves the correctness of the share, while an inequality indicates misbehaviour by the node P_i and its consequential removal from $\mathcal{Q}$.

In asymmetric pairings, elements of $\mathbb{G}$ generally have a shorter representation than those of $\hat{\mathbb{G}}$. Therefore, we put the more frequently accessed system public-key shares in $\mathbb{G}$, while the occasionally transferred client private-key shares belong to $\hat{\mathbb{G}}$. This also leads to a reduction in the ciphertext size. However, for type 2 pairings, an efficient hash-to-$\hat{\mathbb{G}}$ is not available for the group $\hat{\mathbb{G}}$ [20]; in that case we compute the system public key shares in $\hat{\mathbb{G}}$ and use the more feasible group $\mathbb{G}$ for the private key shares.

Proof of Security. Using the encryption and decryption steps of the $\mathsf{FullIdent}$ version of BF-IBE [3, §4.2] along with the above distributed setup and key extraction protocols, we prove the IND-ID-CCA security of BF-IBE with the (n, t)-distributed PKG ((n, t)-$\mathsf{FullIdent}$) based on the BDH assumption. Hereafter, q_E, q_D and q_{H_i} denote the number of extraction, decryption and random oracle H_i queries respectively.

Theorem 1. *Let H_1, H_2, H_3 and H_4 be random oracles. Let $\mathcal{A}_1$ be an IND-ID-CCA adversary that has advantage $\epsilon_1(\kappa)$ in running time $t_1(\kappa)$ against (n,t)-$\mathsf{FullIdent}$ making at most q_E, q_D, q_{H_1}, q_{H_2}, q_{H_3}, and q_{H_4} queries. Then, there exists an algorithm $\mathcal{B}$ that solves the BDH problem in $\mathcal{G}$ with advantage roughly equal to $\epsilon_1(\kappa)/(q_{H_1}q_{H_2}(q_{H_3}+q_{H_4}))$ and running time $O(t_1(\kappa), q_E, q_D, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4})$.*

For their proof, Boneh and Franklin define two additional public key encryption schemes: $\mathsf{BFBasicPub}$ [3, Sec. 4.1], and its IND-CCA secure version $\mathsf{BFBasicPub}^{hy}$ [3, Sec. 4.2] and prove the security of $\mathsf{FullIdent}$ in the following proof sequence: $\mathsf{FullIdent} \rightarrow \mathsf{BFBasicPub}^{hy} \rightarrow \mathsf{BFBasicPub} \rightarrow \mathsf{BDH}$. We use distributed versions of these encryption schemes: (n,t)-$\mathsf{BFBasicPub}^{hy}$ and (n,t)-$\mathsf{BFBasicPub}$ respectively, and prove the proof sequence (n,t)-$\mathsf{FullIdent} \rightarrow (n,t)$-$\mathsf{BFBasicPub}^{hy} \rightarrow (n,t)$-$\mathsf{BFBasicPub} \rightarrow \mathsf{BDH}$. For the complete proof, refer to the extended version of the paper. [1]

3.4 Sakai and Kasahara's SK-IBE

SK-IBE [11] belongs to the exponent-inversion IBE family. Here, the PKG generates a master key $s \in \mathbb{Z}_p$ and a public key $g^s \in \mathbb{G}$ just as in BF-IBE. However, the key-extraction differs significantly. Here, a client with identity $\mathtt{ID}$ receives the private key $d_{\mathtt{ID}} = \hat{g}^{\frac{1}{s+H'_1(\mathtt{ID})}} \in \hat{\mathbb{G}}$, where $H'_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$.

Distributed PKG Setup. The distributed PKG setup remains the exactly same as that of BF-IBE, where $s_i \in \mathbb{Z}_p$ is the master-key share for node P_i and $\mathcal{C}^{(s)}_{\langle g\rangle} = [g^s, g^{s_1}, \cdots, g^{s_n}]$ is the system public-key tuple.

Private-key Extraction. The private-key extraction for SK-IBE is not as straightforward as that for BF-IBE. We modify the $\mathsf{Inverse}$ protocol described in §2.3; specifically, here a private-key extracting client receives w_i from the node in step 3 and instead of nodes, the *client* performs the interpolation. In step 4, instead of publishing, nodes forward $\hat{g}^{z_i}$ and the associated NIZKPK$_{\equiv Com}$ directly to the client, which computes $\hat{g}^z$ and then $d_{\mathtt{ID}} = (\hat{g}^z)^{w^{-1}}$. The reason behind this is to avoid possible key escrow if the node computes both $\hat{g}^z$ and w. Further, the nodes precompute another generator $\hat{h} \in \hat{\mathbb{G}}$ for Pedersen commitments using $\left(\mathcal{C}^{(r)}_{\langle\hat{g}\rangle}, r_i\right) = \mathsf{Random}_{\mathsf{DLog}}(n,t,\hat{g})$, and set $\hat{h} = \left(\mathcal{C}^{(r)}_{\langle\hat{g}\rangle}\right)_0 = \hat{g}^r$.

1. Once a client with identity $\mathtt{ID}$ contacts all n nodes the system, every node P_i authenticates the client's identity, runs $\left(\mathcal{C}^{(z,z')}_{\langle\hat{g},\hat{h}\rangle}, z_i, z'_i\right) = \mathsf{Random}_{\mathsf{Ped}}(n,t,\hat{g},\hat{h})$ and computes $s_i^{\mathtt{ID}} = s_i + H'_1(\mathtt{ID})$ and for $0 \le j \le n$, $\left(\mathcal{C}^{(s^{\mathtt{ID}})}_{\langle g\rangle}\right)_j = \left(\mathcal{C}^{(s)}_{\langle g\rangle}\right)_j g^{H'_1(\mathtt{ID})} = g^{s_j+H'_1(\mathtt{ID})}$. $\mathsf{Random}_{\mathsf{Ped}}$ makes sure that z is uniformly random.
2. P_i performs $\left(\mathcal{C}^{(w,w')}_{\langle\hat{g},\hat{h}\rangle}, w_i, w'_i\right) = \mathsf{Mul}_{\mathsf{Ped}}(n,t,\hat{g},\hat{h},\left(\mathcal{C}^{(s^{\mathtt{ID}})}_{\langle g\rangle}, s_i^{\mathtt{ID}}\right), \left(\mathcal{C}^{(z,z')}_{\langle\hat{g},\hat{h}\rangle}, z_i, z'_i\right))$, where $w = s^{\mathtt{ID}}z = (s+H'_1(\mathtt{ID}))z$ and $w' = (s+H'_1(\mathtt{ID}))z'$ and sends $\left(\mathcal{C}^{(w,w')}_{\langle\hat{g},\hat{h}\rangle}, w_i\right)$ along with NIZKPK$_{\equiv Com}(w_i, w'_i, \left(\mathcal{C}^{(w)}_{\langle\hat{g}\rangle}\right)_i, \left(\mathcal{C}^{(w,w')}_{\langle\hat{g},\hat{h}\rangle}\right)_i)$ to the client, which upon

receiving $t+1$ verifiably correct shares (w_i) reconstructs w using interpolation. If $w \neq 0$, then it computes w^{-1} or else starts again from step 1.

3. Node P_i sends $\left(\mathcal{C}^{(z)}_{\langle \hat{g} \rangle}\right)_i = \hat{g}^{z_i}$ and $\mathcal{C}^{(z,z')}_{\langle \hat{g},\hat{h} \rangle}$ along with $\mathrm{NIZKPK}_{\equiv Com}(z_i, z'_i, \left(\mathcal{C}^{(z)}_{\langle \hat{g} \rangle}\right)_i, \left(\mathcal{C}^{(z,z')}_{\langle \hat{g},\hat{h} \rangle}\right)_i)$ to the client.
4. The client verifies $\left(\mathcal{C}^{(z)}_{\langle \hat{g} \rangle}\right)_i$ using the received $\mathrm{NIZKPK}_{\equiv Com}$, interpolates $t+1$ valid $\hat{g}^{z_i}$ to compute $\hat{g}^z$ and derives her private key $(\hat{g}^z)^{w^{-1}} = \hat{g}^{\frac{1}{(s+H(\mathtt{ID}))}}$.

This protocol can be used without any modification with any type of pairing. Further, online execution of the $\mathsf{Random}_{\mathsf{Ped}}$ computation can be eliminated using batch precomputation of distributed random elements $\left(\mathcal{C}^{(z,z')}_{\langle \hat{g},\hat{h} \rangle}, z_i, z'_i\right)$.

Proof of Security. The security of SK-IBE with a distributed PKG ((n,t)-SK-IBE) is based on the BDHI assumption.

Theorem 2. *Let H'_1, H_2, H_3 and H_4 be random oracles. Let $\mathcal{A}_1$ be an IND-ID-CCA adversary that has advantage $\epsilon_1(\kappa)$ in running time $t_1(\kappa)$ against (n,t)-SK-IBE making at most q_E, q_D, $q_{H'_1}$, q_{H_2}, q_{H_3}, and q_{H_4} queries. Then, there exists an algorithm $\mathcal{B}$ that solves the BDHI problem in $\mathcal{G}$ with advantage roughly equal to $\epsilon_1(\kappa)/(q_{H'_1} q_{H_2}(q_{H_3} + q_{H_4}))$ and running time $O(t_1(\kappa), q_E, q_D, q_{H'_1}, q_{H_2}, q_{H_3}, q_{H_4})$.*

Chen and Cheng [39] prove the security of the original SK-IBE protocol in a proof sequence: SK-IBE → $\mathsf{SKBasicPub}^{hy}$ → $\mathsf{SKBasicPub}$ → BDHI, where $\mathsf{SKBasicPub}$ and $\mathsf{SKBasicPub}^{hy}$ [39, §3.2] are public key encryption schemes based on SK-IBE. We prove Theorem 2 by showing (n,t)-SK-IBE → $\mathsf{SKBasicPub}^{hy}$. For the complete proof, refer to the extended version of the paper [1].

3.5 Boneh and Boyen's BB$_1$-IBE

BB$_1$-IBE belongs to the commutative-blinding IBE family. Boneh and Boyen [32] proposed the original scheme with a security reduction to the decisional BDH assumption [40] in the standard model against selective-identity attacks. However, with a practical requirement of IND-ID-CCA security, in the recent IBCS standard [19], Boyen and Martin proposed a modified version, which is IND-ID-CCA secure in the random oracle model under the BDH assumption. In [9], Boyen rightly claims that for practical applications, it would be preferable to rely on the random-oracle assumption rather than using a less efficient IBE scheme with a stronger security assumption or a weaker attack model. We use the modified BB$_1$-IBE scheme as described in [9] and [19].

In the BB$_1$-IBE setup, the PKG generates a master-key triplet $(\alpha, \beta, \gamma) \in \mathbb{Z}_p^3$ and an associated public key tuple $(g^\alpha, g^\gamma, e(g, \hat{g})^{\alpha\beta})$. A client with identity $\mathtt{ID}$ receives the private key tuple $d_{\mathtt{ID}} = (\hat{g}^{\alpha\beta + (\alpha H'_1(\mathtt{ID}) + \gamma)r}, \hat{g}^r) \in \hat{\mathbb{G}}^2$.

Distributed PKG Setup. In [9], Boyen does not include the parameters $\hat{g}$ and $\hat{g}^\beta$ from the original BB$_1$ scheme [32] in his public key, as they are not required during key extraction, encryption or decryption (they are not omitted for security reasons). In the distributed setting, we in fact need those parameters to be public for efficiency reasons; a verifiable distributed computation of $e(g, \hat{g})^{\alpha\beta}$ becomes inefficient otherwise. To avoid key escrow of clients' private-key components ($\hat{g}^r$), we also need $\hat{h}$ and $\mathcal{C}^{(\beta)}_{\langle \hat{h} \rangle}$; otherwise, parts of clients' private keys would appear in public commitment vectors. As in SK-IBE in §3.4, this extra generator $\hat{h} \in \hat{\mathbb{G}}$ is precomputed using the $\mathsf{Random}_{\mathsf{DLog}}$ protocol. Distributed PKG setup of BB_1 involves distributed generation of the master-key tuple (α, β, γ). Distributed PKG node P_i achieves this using the following three $\mathsf{Random}_{\mathsf{DLog}}$ protocol invocations: $\left(\mathcal{C}^{(\alpha)}_{\langle g \rangle}, \alpha_i\right) = \mathsf{Random}_{\mathsf{DLog}}(n, t, g)$, $\left(\mathcal{C}^{(\beta)}_{\langle \hat{g} \rangle}, \beta_i\right) = \mathsf{Random}_{\mathsf{DLog}}(n, t, \hat{g})$, and $\left(\mathcal{C}^{(\gamma)}_{\langle g \rangle}, \gamma_i\right) = \mathsf{Random}_{\mathsf{DLog}}(n, t, g)$.

Here, $(\alpha_i, \beta_i, \gamma_i)$ is the tuple of master-key shares for node P_i. We also need $\mathcal{C}^{(\beta)}_{\langle \hat{h} \rangle}$; each node P_i provides this by publishing $\left(\mathcal{C}^{(\beta)}_{\langle \hat{h} \rangle}\right)_i = \hat{h}^{\beta_i}$ and the $\mathrm{NIZKPK}_{\equiv DLog}$ $(\beta_i, \hat{g}^{\beta_i}, \hat{h}^{\beta_i})$. The tuple $\left(\mathcal{C}^{(\alpha)}_{\langle g \rangle}, e(g, \hat{g})^{\alpha\beta}, \mathcal{C}^{(\gamma)}_{\langle g \rangle}, \mathcal{C}^{(\beta)}_{\langle \hat{h} \rangle}\right)$ forms the system public key, where $e(g, \hat{g})^{\alpha\beta}$ can computed from the public commitment entries. The vector $\mathcal{C}^{(\beta)}_{\langle \hat{g} \rangle}$, although available publicly, is not required for any further computation.

Private-key Extraction. The most obvious way to compute a BB_1 private key seems to be for P_i to compute $\alpha_i\beta_i + (\alpha_i H'_1(\mathtt{ID}) + \gamma_i)r_i$ and provide the corresponding $\hat{g}^{\alpha_i\beta_i + (\alpha_i H'_1(\mathtt{ID}) + \gamma_i)r_i}, \hat{g}^{r_i}$ to the client, who now needs $2t+1$ valid shares to obtain her private key. However, $\alpha_i\beta_i + (\alpha_i H'_1(\mathtt{ID}) + \gamma_i)r_i$ here is not a share of a random degree-$2t$ polynomial. The possible availability of $\hat{g}^{r_i}$ to the adversary creates a suspicion about secrecy of the master-key share with this method. For private-key extraction in BB_1-IBE with a distributed PKG, we instead use the $\mathsf{Mul}_{\mathsf{BP}}$ protocol in which the client is provided with $\hat{g}^{w_i}$, where $w_i = (\alpha\beta + (\alpha H'_1(\mathtt{ID}) + \gamma)r)_i$ is a share of random degree t polynomial. The protocol works as follows.

1. Once a client with identity $\mathtt{ID}$ contacts all n nodes the system, every node P_i authenticates the client's identity and runs $\left(\mathcal{C}^{(r,r')}_{\langle \hat{h}, \hat{g} \rangle}, [\mathcal{C}^{(r)}_{\langle \hat{h} \rangle}, \mathrm{NIZKPK}_{\equiv Com}], r_i, r_i\right) = \mathsf{Random}_{\mathsf{Ped}}(n, t, f, \hat{h}, \hat{g})$. $\mathsf{Random}_{\mathsf{Ped}}$ makes sure that r is uniformly random.
2. P_i computes its share w_i of $w = \alpha\beta + (\alpha H'_1(\mathtt{ID}) + \gamma)r$ using $\mathsf{Mul}_{\mathsf{BP}}$ in Eq. 9.

   $$\left(\mathcal{C}^{(w)}_{\langle g^* \rangle}, w_i\right) = \mathsf{Mul}_{\mathsf{BP}}(n, t, f, g^*, desc, \left(\mathcal{C}^{(\alpha)}_{\langle g \rangle}, \alpha_i\right), \left(\mathcal{C}^{(\beta)}_{\langle \hat{h} \rangle}, \beta_i\right), \left(\mathcal{C}^{(\gamma)}_{\langle g \rangle}, \gamma_i\right), \left(\mathcal{C}^{(r)}_{\langle \hat{h} \rangle}, r_i\right)).$$

   Here, $desc = \{(1,1,2), (H'_1(\mathtt{ID}), 1, 4), (1,3,4)\}$ is the description of the required binary product under the ordering $(\alpha, \beta, \gamma, r)$ of secrets. To justify our choices of commitment generators, we present the pairing-based verification in protocol $\mathsf{Mul}_{\mathsf{BP}}$: $e(g^{\alpha_i\beta_i + (\alpha_i H'_1(\mathtt{ID}) + \gamma_i)r_i}, \hat{h}) \stackrel{?}{=} e(g^{\alpha_i}, \hat{h}^{\beta_i}) e((g^{\alpha_i})^{H'_1(\mathtt{ID})} g^{\gamma_i}, \hat{h}^{r_i})$. For type 2 and 3 pairings, $g^* = g$, as there is no efficient isomorphism from $\mathbb{G}$ to $\hat{\mathbb{G}}$. For type 1 pairings, we use $g^* = \hat{h} = \phi^{-1}(h)$. Otherwise, the resultant commitments for w (which are public) will contain the private-key part $g^{\alpha\beta + (\alpha H'_1(\mathtt{ID}) + \gamma)r}$.
3. Once the $\mathsf{Mul}_{\mathsf{BP}}$ protocol has succeeded, Node P_i generates $\hat{g}^{w_i}$ and $\hat{g}^{r_i}$ and sends those to the client over a secure and authenticated channel.
4. The client generates her private key $(\hat{g}^{\alpha\beta + (\alpha H'_1(\mathtt{ID}) + \gamma)r}, \hat{g}^r)$ by interpolating the valid received shares. For type 1 and type 2 pairings, the client can use the pairing-based DDH solving to check the validity of the shares. However, for type 3 pairings,

without an efficient mapping from $\hat{\mathbb{G}}$ to $\mathbb{G}$, pairing-based DDH solving can only be employed to verify $\hat{g}^{w_i}$. As a verification of $\hat{g}^{r_i}$, node P_i includes a NIZKPK$_{\equiv DLog}$ $(r_i, \hat{h}^{r_i}, \hat{g}^{r_i})$ along with $\hat{g}^{w_i}$ and $\hat{g}^{r_i}$.

As in SK-IBE in §3.4, online execution of the $\mathsf{Random}_{\mathsf{DLog}}$ computation can be eliminated using batch precomputation of distributed random elements $\left(\mathcal{C}^{(r)}_{\langle\hat{h}\rangle}, r_i\right)$.

Proof of Security. Along with the above distributed setup and private-key extraction protocols, we prove IND-ID-CCA security of BB$_1$-IBE with the (n,t)-distributed PKG ((n,t)-BB$_1$-IBE) based on the BDH assumption. To the best of our knowledge, an IND-ID-CCA security proof for the modified BB$_1$-IBE scheme has not been published yet.

Theorem 3. *Let H'_1, H_2, H_3 and H'_4 be random oracles. Let $\mathcal{A}$ be an IND-ID-CCA adversary that has advantage $\epsilon(\kappa)$ in running time $t(\kappa)$ against (n,t)-BB$_1$-IBE making at most q_E, q_D, $q_{H'_1}$, q_{H_2}, $q_{H'_3}$, and q_{H_4} queries. Then, there exists an algorithm $\mathcal{B}$ that solves the BDH problem in $\mathcal{G}$ with advantage roughly equal to $\epsilon(\kappa)/(q_{H'_1} q_{H'_3})$ and running time $O(t(\kappa), q_E, q_D, q_{H'_1}, q_{H_2}, q_{H'_3}, q_{H_4})$.*

For the proof, refer to the extended version of the paper [1].

Using a more expensive DKG protocol with uniformly random output, all of our proofs would become relatively simpler. However, note that our use of DKG without uniformly random output does not affect the security reduction factor in any proof. This is something not achieved for the known previous protocols with non-uniform DKG such as threshold Schorr signatures [28]. Further, we do not discuss the liveness and agreement properties for our asynchronous protocols as liveness and agreement of all the distributed primitives provides liveness and agreement for the distributed PKG setup and distributed key extraction protocols. Finally, for simplicity of the discussion, it would have been better to combine three proofs. However, that looks difficult, if not impossible, as the distributed computation tools used in these distributed PKGs and the original IBE security proofs vary a lot from a scheme to scheme.

Finally, observing the importance of proactiveness and a capability to handle group dynamics in any practical system, we also discuss the proactive security and group modification primitives for our distributed PKGs in the extended version of the paper [1].

4 Comparing Distributed PKGs

In this section, we discuss the performance of the setup and key extraction protocols of the above three distributed PKGs. For a detailed comparison of the encryption and decryption steps of BF-IBE, SK-IBE and BB$_1$-IBE, refer to [9]. The general recommendations from this survey are to avoid SK-IBE and other exponent-inversion IBEs due to their reliance on the strong BDHI assumption, and that BB$_1$-IBE and BF-IBE both are good, but BB$_1$-IBE can be a better choice due to BF-IBE's less efficient encryption.

Table 1 provides a detailed operation count and key size comparison of three distributed PKGs. We count DKG instances, pairings, NIZKPKs, interpolations and public and private key sizes. We leave aside the comparatively small exponentiations and other group operations. As mentioned in §3.5, for BB$_1$-IBE, with pairings of type 1 and 2,

Table 1. Operation count and key sizes for distributed PKG setups and distributed private-key extractions (per key)

	BF-IBE		**SK-IBE**		**BB$_1$-IBE**	
	Setup	Extraction	Setup	Extraction	Setup	Extraction
Operation Count						
Generator h or $\hat{h}$	X		✓		✓	
DKG[a] (precomputed)	-	0	-	1^P	-	1^P
DKG (online)	1^D	0	1^D	1^P	3^D	1^D
Parings @PKG Node	0	0	0	$2n$	1[b]	$2n$
Parings @Client	-	$2(2t+2)$	-	0	-	$2n$[b]
NIZKPK	0	0	0	$2n$	n[b]	$2n$[b]
Interpolations	0	1	0	2	1	2
Key Sizes						
PKG Public Key	$(n+2)\mathbb{G}$[c]		$(n+3)\mathbb{G}$		$(2n+3)\mathbb{G}$, $(n+2)\hat{\mathbb{G}}$, $(1)\mathbb{G}_T$	
Private-key Shares	$(2t+1)\hat{\mathbb{G}}$[c]		$(3n)\mathbb{Z}_p$, $(3n+1)\hat{\mathbb{G}}$		$(2n)\mathbb{Z}_p$[b], $(2n)\hat{\mathbb{G}}$	

[a] For DKG, D indicates use of DLog commitments, while P indicates Pedersen commitments.

[b] For type 1 and 2 pairings, $2n$ extra pairings replace n NIZKPKs. Further, the $2n$ $\mathbb{Z}_p$ elements are omitted from the private-key shares.

[c] For type 2 parings, the groups used for the PKG public key and the private-key shares are interchanged.

there is a choice that can be made between using n NIZKPKs and $2n$ pairing computations. The table shows the NIZKPK choice (the only option for type 3 pairings), and footnote b shows where NIZKPKs can be traded off for pairings. An efficient algorithm for hash-to-$\hat{\mathbb{G}}$ is not available for type 2 pairing curves and we interchange the groups used for the public key and client private-key shares. Footnote c indicates how that affects the key sizes.

In Table 1, we observe that the distributed PKG setup and the distributed private-key extraction protocols for BF-IBE are significantly more efficient than those for SK-IBE and BB$_1$-IBE. Importantly, for BF-IBE, distributed PKG nodes can extract a key for a client without interacting with each other, which is not possible in the other two schemes; both BB$_1$-IBE and SK-IBE require at least one DKG instance for every private-key extraction; the second required instance can be batch precomputed. Therefore, for IBE applications in the random oracle model, we suggest the use of the BF-IBE scheme, except in situations where private-key extractions are rare and efficiency of the encryption step is critical to the system. For such applications, we suggest BB$_1$-IBE as the small efficiency gains in the distributed PKG setup and extraction protocols of SK-IBE do not well compensate for the strong security assumption required. BB$_1$-IBE is also more suitable for type 2 pairings, where an efficient map-to-group hash function H_1 is not available. Further, BB$_1$-IBE can also be proved secure in the standard model with selective-identity attacks. For applications demanding security in the standard model, our distributed PKG for BB$_1$-IBE also provides a solution to the key escrow and single point of failure problems, using pairings of type 1 or 2.

5 Concluding Remarks

We have designed and compared distributed PKG setup and private key extraction protocols for BF-IBE, SK-IBE, and BB_1-IBE. We observed that the distributed PKG protocol for BF-IBE is the most efficient among all and we suggest its use when the system can support its relatively costly encryption step. For systems requiring a faster encryption, we suggest the use of BB_1-IBE instead. However, during every distributed private key extraction, it requires a DKG and consequently, interaction among PKG nodes. That being said, during private-key extractions, we successfully avoid any interaction between clients and PKG nodes except the necessary identity at the start and key share transfers at the end. Finally, each of the above schemes represents a separate IBE framework and our designs can be applied to other schemes in those frameworks as well.

Acknowledgements. This work is supported by NSERC, MITACS, and a David R. Cheriton Graduate Scholarship. We specially thank Sanjit Chatterjee for his suggestions regarding the pairing types and the IBE literature. We also thank Alfred Menezes, Kenny Paterson and the anonymous reviewers for helpful discussions and suggestions.

References

1. Kate, A., Goldberg, I.: Asynchronous Distributed Private-Key Generators for Identity-Based Cryptography. Cryptology ePrint Archive, Report 2009/355 at `http://eprint.iacr.org/2009/355` (April 2010)
2. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: CRYPTO'84. (1984) 47–53
3. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: CRYPTO'01. (2001) 213–229
4. Joye, M., Neven, G.: Identity-Based Cryptography - Volume 2 Cryptology and Information Security Series. IOS Press, Amsterdam, The Netherlands, The Netherlands (2008)
5. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: FOCS'87. (1987) 427–437
6. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. In: EUROCRYPT'99. (1999) 295–310
7. Kate, A., Zaverucha, G.M., Goldberg, I.: Pairing-Based Onion Routing. In: PETS'07. (2007) 95–112
8. Abdalla, M., Catalano, D., Fiore, D.: Verifiable Random Functions from Identity-Based Key Encapsulation. In: EUROCRYPT'09. (2009) 554–571
9. Boyen, X.: A Tapestry of Identity-based Encryption: Practical Frameworks Compared. IJACT **1**(1) (2008) 3–21
10. Geisler, M., Smart, N.P.: Distributing the Key Distribution Centre in Sakai-Kasahara Based Systems. In: IMA Int. Conf. on Cryptography and Coding. (2009) 252–262
11. Sakai, R., Kasahara, M.: ID based Cryptosystems with Pairing on Elliptic Curve. Cryptology ePrint Archive, Report 2003/054 (2003)
12. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: ASIACRYPT'03. (2003) 452–473
13. Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., Yoo, S.: Secure key issuing in ID-based cryptography. In: ACSW Frontiers'04. (2004) 69–74
14. Gangishetti, R., Gorantla, M.C., Das, M., Saxena, A.: Threshold key issuing in identity-based cryptosystems. Computer Standards & Interfaces **29**(2) (2007) 260–264

15. Chunxiang, X., Junhui, Z., Zhiguang, Q.: A Note on Secure Key Issuing in ID-based Cryptography. Technical report (2005) http://eprint.iacr.org/2005/180.
16. Goyal, V.: Reducing Trust in the PKG in Identity Based Cryptosystems. In: CRYPTO'07. (2007) 430–447
17. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: PKC'03. (2003) 31–46
18. Wang, H., Zhang, Y., Feng, D.: Short Threshold Signature Schemes Without Random Oracles. In: INDOCRYPT'03. (2005) 297–310
19. Boyen, X., Martin, L.: Identity-Based Cryptography Standard (IBCS) (Version 1), Request for Comments (RFC) 5091. http://www.ietf.org/rfc/rfc5091.txt (2007)
20. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics **156**(16) (2008) 3113–3121
21. Blake, I., Seroussi, G., Smart, N.P., eds.: Advances in Elliptic Curve Cryptography. Number 317 in London Mathematical Society Lecture Note Series. (2005) 183–252.
22. Chatterjee, S., Menezes, A.: On Cryptographic Protocols Employing Asymmetric Pairings - The Role of Ψ Revisited. CACR 2009-34 at http://www.cacr.math.uwaterloo.ca/techreports/2007/cacr2009-34.pdf (2009)
23. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO'86. (1986) 186–194
24. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: CRYPTO'91. (1991) 129–140
25. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: CRYPTO. (1992) 89–105
26. Joux, A., Nguyen, K.: Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. Journal of Cryptology **16**(4) (2003) 239–247
27. Kate, A., Goldberg, I.: Distributed Key Generation for the Internet. In: ICDCS'09. (2009) 119–128
28. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. Journal of Cryptology **20**(1) (2007) 51–83
29. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive Security for Threshold Cryptosystems. In: CRYPTO'99. (1999) 98–115
30. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. In: ANTS-IV. (2000) 385–394
31. Mitsunari, S., Sakai, R., Kasahara, M.: A New Traitor Tracing. IEICE Transactions **E85-A**(2) (2002) 481–484
32. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: EUROCRYPT'04. (2004) 223–238
33. Bar-Ilan, J., Beaver, D.: Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction. In: PODC'89. (1989) 201–209
34. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC'88. (1988) 1–10
35. Cachin, C., Kursawe, K., A.Lysyanskaya, Strobl, R.: Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. In: ACM CCS'02. (2002) 88–97
36. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and Fact-Track Multiparty Computations with Applications to Threshold Cryptography. In: PODC'98. (1998) 101–111
37. Shamir, A.: How to Share a Secret. Commun. ACM **22**(11) (1979) 612–613
38. Boyen, X.: General *Ad Hoc* Encryption from Exponent Inversion IBE. In: EUROCRYPT'07. (2007) 394–411
39. Chen, L., Cheng, Z.: Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. In: IMA Int. Conf. (2005) 442–459
40. Joux, A.: The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In: ANTS-V. (2002) 20–32