

**Security.** Next, we study the security of this basic scheme. The following theorem shows that **BasicIdent** is a semantically secure identity based encryption scheme (IND-ID-CPA) assuming BDH is hard in groups generated by  $\mathcal{G}$ .

**Theorem 4.1.** *Suppose the hash functions  $H_1, H_2$  are random oracles. Then **BasicIdent** is a semantically secure identity based encryption scheme (IND-ID-CPA) assuming BDH is hard in groups generated by  $\mathcal{G}$ . Concretely, suppose there is an IND-ID-CPA adversary  $\mathcal{A}$  that has advantage  $\epsilon(k)$  against the scheme **BasicIdent**. Suppose  $\mathcal{A}$  makes at most  $q_E > 0$  private key extraction queries and  $q_{H_2} > 0$  hash queries to  $H_2$ . Then there is an algorithm  $\mathcal{B}$  that solves BDH in groups generated by  $\mathcal{G}$  with advantage at least:*

$$Adv_{\mathcal{G}, \mathcal{B}}(k) \geq \frac{2\epsilon(k)}{e(1 + q_E) \cdot q_{H_2}}$$

Here  $e \approx 2.71$  is the base of the natural logarithm. The running time of  $\mathcal{B}$  is  $O(\text{time}(\mathcal{A}))$ .

To prove the theorem we first define a related Public Key Encryption scheme (not an identity based scheme), called **BasicPub**. **BasicPub** is described by three algorithms: **keygen**, **encrypt**, **decrypt**.

**keygen:** Given a security parameter  $k \in \mathbb{Z}^+$ , the algorithm works as follows:

Step 1: Run  $\mathcal{G}$  on input  $k$  to generate two prime order groups  $\mathbb{G}_1, \mathbb{G}_2$  and a bilinear map  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ . Let  $q$  be the order of  $\mathbb{G}_1, \mathbb{G}_2$ . Choose a random generator  $P \in \mathbb{G}_1$ .

Step 2: Pick a random  $s \in \mathbb{Z}_q^*$  and set  $P_{pub} = sP$ . Pick a random  $Q_{ID} \in \mathbb{G}_1^*$ .

Step 3: Choose a cryptographic hash function  $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$  for some  $n$ .

Step 4: The public key is  $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{ID}, H_2 \rangle$ . The private key is  $d_{ID} = sQ_{ID} \in \mathbb{G}_1^*$ .

**encrypt:** To encrypt  $M \in \{0, 1\}^n$  choose a random  $r \in \mathbb{Z}_q^*$  and set the ciphertext to be:

$$C = \langle rP, M \oplus H_2(g^r) \rangle \quad \text{where} \quad g = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*$$

**decrypt:** Let  $C = \langle U, V \rangle$  be a ciphertext created using the public key  $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{ID}, H_2 \rangle$ .

To decrypt  $C$  using the private key  $d_{ID} \in \mathbb{G}_1^*$  compute:

$$V \oplus H_2(\hat{e}(d_{ID}, U)) = M$$

This completes the description of **BasicPub**. We now prove Theorem 4.1 in two steps. We first show that an IND-ID-CPA attack on **BasicIdent** can be converted to a IND-CPA attack on **BasicPub**. This step shows that private key extraction queries do not help the adversary. We then show that **BasicPub** is IND-CPA secure if the BDH assumption holds.

**Lemma 4.2.** *Let  $H_1$  be a random oracle from  $\{0,1\}^*$  to  $\mathbb{G}_1^*$ . Let  $\mathcal{A}$  be an IND-ID-CPA adversary that has advantage  $\epsilon(k)$  against **BasicIdent**. Suppose  $\mathcal{A}$  makes at most  $q_E > 0$  private key extraction queries. Then there is a IND-CPA adversary  $\mathcal{B}$  that has advantage at least  $\epsilon(k)/e(1+q_E)$  against **BasicPub**. Its running time is  $O(\text{time}(\mathcal{A}))$ .*

**Proof.** We show how to construct an IND-CPA adversary  $\mathcal{B}$  that uses  $\mathcal{A}$  to gain advantage  $\epsilon/e(1+q_E)$  against **BasicPub**. The game between the challenger and the adversary  $\mathcal{B}$  starts with the challenger first generating a random public key by running algorithm **keygen** of **BasicPub**. The result is a public key  $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{ID}, H_2 \rangle$  and a private key  $d_{ID} = sQ_{ID}$ . As usual,  $q$  is the order of  $\mathbb{G}_1, \mathbb{G}_2$ . The challenger gives  $K_{pub}$  to algorithm  $\mathcal{B}$ . Algorithm  $\mathcal{B}$  is supposed to output two messages  $M_0$  and  $M_1$  and expects to receive back the **BasicPub** encryption of  $M_b$  under  $K_{pub}$  where  $b \in \{0,1\}$ . Then algorithm  $\mathcal{B}$  outputs its guess  $b' \in \{0,1\}$  for  $b$ .

Algorithm  $\mathcal{B}$  works by interacting with  $\mathcal{A}$  in an IND-ID-CPA game as follows ( $\mathcal{B}$  simulates the challenger for  $\mathcal{A}$ ):

**Setup:** Algorithm  $\mathcal{B}$  gives  $\mathcal{A}$  the **BasicIdent** system parameters  $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2 \rangle$ . Here  $q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_2$  are taken from  $K_{pub}$ , and  $H_1$  is a random oracle controlled by  $\mathcal{B}$  as described below.

**$H_1$ -queries:** At any time algorithm  $\mathcal{A}$  can query the random oracle  $H_1$ . To respond to these queries algorithm  $\mathcal{B}$  maintains a list of tuples  $\langle ID_j, Q_j, b_j, c_j \rangle$  as explained below. We refer to this list as the  $H_1^{list}$ . The list is initially empty. When  $\mathcal{A}$  queries the oracle  $H_1$  at a point  $ID_i$  algorithm  $\mathcal{B}$  responds as follows:

1. If the query  $ID_i$  already appears on the  $H_1^{list}$  in a tuple  $\langle ID_i, Q_i, b_i, c_i \rangle$  then Algorithm  $\mathcal{B}$  responds with  $H_1(ID_i) = Q_i \in \mathbb{G}_1^*$ .
2. Otherwise,  $\mathcal{B}$  generates a random  $coin \in \{0,1\}$  so that  $\Pr[coin = 0] = \delta$  for some  $\delta$  that will be determined later.
3. Algorithm  $\mathcal{B}$  picks a random  $b \in \mathbb{Z}_q^*$ .  
If  $coin = 0$  compute  $Q_i = bP \in \mathbb{G}_1^*$ . If  $coin = 1$  compute  $Q_i = bQ_{ID} \in \mathbb{G}_1^*$ .
4. Algorithm  $\mathcal{B}$  adds the tuple  $\langle ID_i, Q_i, b, coin \rangle$  to the  $H_1^{list}$  and responds to  $\mathcal{A}$  with  $H_1(ID_i) = Q_i$ .  
Note that either way  $Q_i$  is uniform in  $\mathbb{G}_1^*$  and is independent of  $\mathcal{A}$ 's current view as required.

**Phase 1:** Let  $ID_i$  be a private key extraction query issued by algorithm  $\mathcal{A}$ . Algorithm  $\mathcal{B}$  responds to this query as follows:

1. Run the above algorithm for responding to  $H_1$ -queries to obtain a  $Q_i \in \mathbb{G}_1^*$  such that  $H_1(ID_i) = Q_i$ . Let  $\langle ID_i, Q_i, b_i, coin_i \rangle$  be the corresponding tuple on the  $H_1^{list}$ . If  $coin_i = 1$  then  $\mathcal{B}$  reports failure and terminates. The attack on **BasicPub** failed.
2. We know  $coin_i = 0$  and hence  $Q_i = b_iP$ . Define  $d_i = b_iP_{pub} \in \mathbb{G}_1^*$ . Observe that  $d_i = sQ_i$  and therefore  $d_i$  is the private key associated to the public key  $ID_i$ . Give  $d_i$  to algorithm  $\mathcal{A}$ .

**Challenge:** Once algorithm  $\mathcal{A}$  decides that Phase 1 is over it outputs a public key  $ID_{ch}$  and two messages  $M_0, M_1$  on which it wishes to be challenged. Algorithm  $\mathcal{B}$  responds as follows:

1. Algorithm  $\mathcal{B}$  gives its challenger the messages  $M_0, M_1$ . The challenger responds with a **BasicPub** ciphertext  $C = \langle U, V \rangle$  such that  $C$  is the encryption of  $M_c$  for a random  $c \in \{0,1\}$ .
2. Next,  $\mathcal{B}$  runs the algorithm for responding to  $H_1$ -queries to obtain a  $Q \in \mathbb{G}_1^*$  such that  $H_1(ID_{ch}) =$

$Q$ . Let  $\langle \text{ID}_{ch}, Q, b, \text{coin} \rangle$  be the corresponding tuple on the  $H_1^{list}$ . If  $\text{coin} = 0$  then  $\mathcal{B}$  reports failure and terminates. The attack on **BasicPub** failed.

3. We know  $\text{coin} = 1$  and therefore  $Q = bQ_{\text{ID}}$ . Recall that when  $C = \langle U, V \rangle$  we have  $U \in \mathbb{G}_1^*$ . Set  $C' = \langle b^{-1}U, V \rangle$ , where  $b^{-1}$  is the inverse of  $b \bmod q$ . Algorithm  $\mathcal{B}$  responds to  $\mathcal{A}$  with the challenge ciphertext  $C'$ . Note that  $C'$  is a proper **BasicIdent** encryption of  $M_c$  under the public key  $\text{ID}_{ch}$  as required. To see this first observe that, since  $H_1(\text{ID}_{ch}) = Q$ , the private key corresponding to  $\text{ID}_{ch}$  is  $d_{ch} = sQ$ . Second, observe that

$$\hat{e}(b^{-1}U, d_{ch}) = \hat{e}(b^{-1}U, sQ) = \hat{e}(U, sb^{-1}Q) = \hat{e}(U, sQ_{\text{ID}}) = \hat{e}(U, d_{\text{ID}}).$$

Hence, the **BasicIdent** decryption of  $C'$  using  $d_{ch}$  is the same as the **BasicPub** decryption of  $C$  using  $d_{\text{ID}}$ .

**Phase 2:** Algorithm  $\mathcal{B}$  responds to private key extraction queries as in Phase 1.

**Guess:** Eventually algorithm  $\mathcal{A}$  outputs a guess  $c'$  for  $c$ . Algorithm  $\mathcal{B}$  outputs  $c'$  as its guess for  $c$ .

**Claim:** If algorithm  $\mathcal{B}$  does not abort during the simulation then algorithm  $\mathcal{A}$ 's view is identical to its view in the real attack. Furthermore, if  $\mathcal{B}$  does not abort then  $|\Pr[c = c'] - \frac{1}{2}| \geq \epsilon$ . The probability is over the random bits used by  $\mathcal{A}, \mathcal{B}$  and the challenger.

*Proof of claim.* The responses to  $H_1$ -queries are as in the real attack since each response is uniformly and independently distributed in  $\mathbb{G}_1^*$ . All responses to private key extraction queries are valid. Finally, the challenge ciphertext  $C'$  given to  $\mathcal{A}$  is the **BasicIdent** encryption of  $M_c$  for some random  $c \in \{0, 1\}$ . Therefore, by definition of algorithm  $\mathcal{A}$  we have that  $|\Pr[c = c'] - \frac{1}{2}| \geq \epsilon$ .  $\square$

To complete the proof of Lemma 4.2 it remains to calculate the probability that algorithm  $\mathcal{B}$  aborts during the simulation. Suppose  $\mathcal{A}$  makes a total of  $q_E$  private key extraction queries. Then the probability that  $\mathcal{B}$  does not abort in phases 1 or 2 is  $\delta^{q_E}$ . The probability that it does not abort during the challenge step is  $1 - \delta$ . Therefore, the probability that  $\mathcal{B}$  does not abort during the simulation is  $\delta^{q_E}(1 - \delta)$ . This value is maximized at  $\delta_{opt} = 1 - 1/(q_E + 1)$ . Using  $\delta_{opt}$ , the probability that  $\mathcal{B}$  does not abort is at least  $1/e(1 + q_E)$ . This shows that  $\mathcal{B}$ 's advantage is at least  $\epsilon/e(1 + q_E)$  as required.  $\square$

The analysis used in the proof of Lemma 4.2 uses a similar technique to Coron's analysis of the Full Domain Hash signature scheme [9]. Next, we show that **BasicPub** is a semantically secure public key system if the BDH assumption holds.

**Lemma 4.3.** *Let  $H_2$  be a random oracle from  $\mathbb{G}_2$  to  $\{0, 1\}^n$ . Let  $\mathcal{A}$  be an IND-CPA adversary that has advantage  $\epsilon(k)$  against **BasicPub**. Suppose  $\mathcal{A}$  makes a total of  $q_{H_2} > 0$  queries to  $H_2$ . Then there is an algorithm  $\mathcal{B}$  that solves the BDH problem for  $\mathcal{G}$  with advantage at least  $2\epsilon(k)/q_{H_2}$  and a running time  $O(\text{time}(\mathcal{A}))$ .*

**Proof.** Algorithm  $\mathcal{B}$  is given as input the BDH parameters  $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$  produced by  $\mathcal{G}$  and a random instance  $\langle P, aP, bP, cP \rangle = \langle P, P_1, P_2, P_3 \rangle$  of the BDH problem for these parameters, i.e.  $P$  is random in  $\mathbb{G}_1^*$  and  $a, b, c$  are random in  $\mathbb{Z}_q^*$  where  $q$  is the order of  $\mathbb{G}_1, \mathbb{G}_2$ . Let  $D = \hat{e}(P, P)^{abc} \in \mathbb{G}_2$  be the solution to this BDH problem. Algorithm  $\mathcal{B}$  finds  $D$  by interacting with  $\mathcal{A}$  as follows:

**Setup:** Algorithm  $\mathcal{B}$  creates the **BasicPub** public key  $K_{pub} = \langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, Q_{\text{ID}}, H_2 \rangle$  by setting  $P_{pub} = P_1$  and  $Q_{\text{ID}} = P_2$ . Here  $H_2$  is a random oracle controlled by  $\mathcal{B}$  as described below. Algorithm  $\mathcal{B}$  gives  $\mathcal{A}$  the **BasicPub** public key  $K_{pub}$ . Observe that the (unknown) private key associated to  $K_{pub}$  is  $d_{\text{ID}} = aQ_{\text{ID}} = abP$ .

**$H_2$ -queries:** At any time algorithm  $\mathcal{A}$  may issue queries to the random oracle  $H_2$ . To respond to these queries  $\mathcal{B}$  maintains a list of tuples called the  $H_2^{list}$ . Each entry in the list is a tuple of the form  $\langle X_j, H_j \rangle$ . Initially the list is empty. To respond to query  $X_i$  algorithm  $\mathcal{B}$  does the following:

1. If the query  $X_i$  already appears on the  $H_2^{list}$  in a tuple  $\langle X_i, H_i \rangle$  then respond with  $H_2(X_i) = H_i$ .
2. Otherwise,  $\mathcal{B}$  just picks a random string  $H_i \in \{0, 1\}^n$  and adds the tuple  $\langle X_i, H_i \rangle$  to the  $H_2^{list}$ . It responds to  $\mathcal{A}$  with  $H_2(X_i) = H_i$ .

**Challenge:** Algorithm  $\mathcal{A}$  outputs two messages  $M_0, M_1$  on which it wishes to be challenged. Algorithm  $\mathcal{B}$  picks a random string  $R \in \{0, 1\}^n$  and defines  $C$  to be the ciphertext  $C = \langle P_3, R \rangle$ . Algorithm  $\mathcal{B}$  gives  $C$  as the challenge to  $\mathcal{A}$ . Observe that, by definition, the decryption of  $C$  is  $R \oplus H_2(\hat{e}(P_3, d_D)) = R \oplus H_2(D)$ .

**Guess:** Algorithm  $\mathcal{A}$  outputs its guess  $c' \in \{0, 1\}$ . At this point  $\mathcal{B}$  picks a random tuple  $\langle X_j, H_j \rangle$  from the  $H_2^{list}$  and outputs  $X_j$  as the solution to the given instance of BDH.

Algorithm  $\mathcal{B}$  is simulating a real attack environment for algorithm  $\mathcal{A}$  (it simulates the challenger and the oracle for  $H_2$ ). We show that algorithm  $\mathcal{B}$  outputs the correct answer  $D$  with probability at least  $2\epsilon/q_{H_2}$  as required. The proof is based on comparing  $\mathcal{A}$ 's behavior in the simulation to its behavior in a real IND-CPA attack game (against a real challenger and a real random oracle for  $H_2$ ).

Let  $\mathcal{H}$  be the event that algorithm  $\mathcal{A}$  issues a query for  $H_2(D)$  at some point during the simulation above (this implies that at the end of the simulation  $D$  appears in some tuple on the  $H_2^{list}$ ). We show that  $\Pr[\mathcal{H}] \geq 2\epsilon$ . This will prove that algorithm  $\mathcal{B}$  outputs  $D$  with probability at least  $2\epsilon/q_{H_2}$ . We also study event  $\mathcal{H}$  in the real attack game, namely the event that  $\mathcal{A}$  issues a query for  $H_2(D)$  when communicating with a real challenger and a real random oracle for  $H_2$ .

**Claim 1:**  $\Pr[\mathcal{H}]$  in the simulation above is equal to  $\Pr[\mathcal{H}]$  in the real attack.

Proof of claim. Let  $\mathcal{H}_\ell$  be the event that  $\mathcal{A}$  makes a query for  $H_2(D)$  in one of its first  $\ell$  queries to the  $H_2$  oracle. We prove by induction on  $\ell$  that  $\Pr[\mathcal{H}_\ell]$  in the real attack is equal to  $\Pr[\mathcal{H}_\ell]$  in the simulation for all  $\ell \geq 0$ . Clearly  $\Pr[\mathcal{H}_0] = 0$  in both the simulation and in the real attack. Now suppose that for some  $\ell > 0$  we have that  $\Pr[\mathcal{H}_{\ell-1}]$  in the simulation is equal to  $\Pr[\mathcal{H}_{\ell-1}]$  in the real attack. We show that the same holds for  $\mathcal{H}_\ell$ . We know that:

$$\begin{aligned} \Pr[\mathcal{H}_\ell] &= \Pr[\mathcal{H}_\ell \mid \mathcal{H}_{\ell-1}] \Pr[\mathcal{H}_{\ell-1}] + \Pr[\mathcal{H}_\ell \mid \neg\mathcal{H}_{\ell-1}] \Pr[\neg\mathcal{H}_{\ell-1}] \\ &= \Pr[\mathcal{H}_{\ell-1}] + \Pr[\mathcal{H}_\ell \mid \neg\mathcal{H}_{\ell-1}] \Pr[\neg\mathcal{H}_{\ell-1}] \end{aligned} \tag{1}$$

We argue that  $\Pr[\mathcal{H}_\ell \mid \neg\mathcal{H}_{\ell-1}]$  in the simulation is equal to  $\Pr[\mathcal{H}_\ell \mid \neg\mathcal{H}_{\ell-1}]$  in the real attack. To see this observe that as long as  $\mathcal{A}$  does not issue a query for  $H_2(D)$  its view during the simulation is identical to its view in the real attack (against a real challenger and a real random oracle for  $H_2$ ). Indeed, the public-key and the challenge are distributed as in the real attack. Similarly, all responses to  $H_2$ -queries are uniform and independent in  $\{0, 1\}^n$ . Therefore,  $\Pr[\mathcal{H}_\ell \mid \neg\mathcal{H}_{\ell-1}]$  in the simulation is equal to  $\Pr[\mathcal{H}_\ell \mid \neg\mathcal{H}_{\ell-1}]$  in the real attack. It follows by (1) and the inductive hypothesis that  $\Pr[\mathcal{H}_\ell]$  in the real attack is equal to  $\Pr[\mathcal{H}_\ell]$  in the simulation. By induction on  $\ell$  we obtain that  $\Pr[\mathcal{H}]$  in the real attack is equal to  $\Pr[\mathcal{H}]$  in the simulation.  $\square$

**Claim 2:** In the real attack we have  $\Pr[\mathcal{H}] \geq 2\epsilon$ .

Proof of claim. In the real attack, if  $\mathcal{A}$  never issues a query for  $H_2(D)$  then the decryption of  $C$  is independent of  $\mathcal{A}$ 's view (since  $H_2(D)$  is independent of  $\mathcal{A}$ 's view). Therefore, in the real attack  $\Pr[c = c' \mid \neg\mathcal{H}] = 1/2$ . By definition of  $\mathcal{A}$ , we know that in the real attack  $|\Pr[c = c'] - 1/2| \geq \epsilon$ .

We show that these two facts imply that  $\Pr[\mathcal{H}] \geq 2\epsilon$ . To do so we first derive simple upper and lower bounds on  $\Pr[c = c']$ :

$$\begin{aligned} \Pr[c = c'] &= \Pr[c = c' | \neg \mathcal{H}] \Pr[\neg \mathcal{H}] + \Pr[c = c' | \mathcal{H}] \Pr[\mathcal{H}] \leq \\ &\leq \Pr[c = c' | \neg \mathcal{H}] \Pr[\neg \mathcal{H}] + \Pr[\mathcal{H}] = \frac{1}{2} \Pr[\neg \mathcal{H}] + \Pr[\mathcal{H}] = \frac{1}{2} + \frac{1}{2} \Pr[\mathcal{H}] \\ \Pr[c = c'] &\geq \Pr[c = c' | \neg \mathcal{H}] \Pr[\neg \mathcal{H}] = \frac{1}{2} - \frac{1}{2} \Pr[\mathcal{H}] \end{aligned}$$

It follows that  $\epsilon \leq |\Pr[c = c'] - 1/2| \leq \frac{1}{2} \Pr[\mathcal{H}]$ . Therefore, in the real attack  $\Pr[\mathcal{H}] \geq 2\epsilon$ .  $\square$

To complete the proof of Lemma 4.3 observe that by Claims 1 and 2 we know that  $\Pr[\mathcal{H}] \geq 2\epsilon$  in the simulation above. Hence, at the end of the simulation,  $D$  appears in some tuple on the  $H_2^{\text{list}}$  with probability at least  $2\epsilon$ . It follows that  $\mathcal{B}$  produces the correct answer with probability at least  $2\epsilon/q_{H_2}$  as required.  $\square$

We note that one can slightly vary the reduction in the proof above to obtain different bounds. For example, in the ‘Guess’ step above one can avoid having to pick a random element from the  $H_2^{\text{list}}$  by using the random self reduction of the BDH problem. This requires running algorithm  $\mathcal{A}$  multiple times (as in Theorem 7 of [42]). The success probability for solving the given BDH problem increases at the cost of also increasing the running time.

**Proof of Theorem 4.1.** The theorem follows directly from Lemma 4.2 and Lemma 4.3. Composing both reductions shows that an IND-ID-CPA adversary on BasicIdent with advantage  $\epsilon(k)$  gives a BDH algorithm for  $\mathcal{G}$  with advantage at least  $2\epsilon(k)/e(1 + q_E)q_{H_2}$ , as required.  $\square$