# Exposure-Resilience for Free: The Hierarchical ID-based Encryption Case


Yevgeniy Dodis
Department of Computer Science
New York University
Email: dodis@cs.nyu.edu

Moti Yung
Department of Computer Science
Columbia University
Email: moti@cs.columbia.edu



## Abstract

*In the problem of* gradual key exposure *[7] (which is very closely related to the problem of* proactive security *[27]), the secret key is assumed to be slowly compromised over time, so that more and more information about a secret key is eventually leaked. This models the general situation in the real world where memory, storage systems and devices cannot perfectly hide all information for long time (due to physical and operational leakages). In this setting, in order to protect against exposure threats, the secret key is represented in an "exposure-resilient" form, which is periodically refreshed with the following guarantee: as long as the adversary does not learn "too much" information about the current representation of the secret between successive refreshes, the system should remain secure.*

*To measure the efficiency of a given solution, one considers the "natural" secret key representation $A$, the "exposure-resilient" representation $B$, and examines the following three measures: (1)* space loss *which is the extra space required by $B$ over $A$; (2)* time loss *which is the operation slowdown when $B$ is used in place of $A$; and (3)* exposure-resilience *which is the fraction of $B$ which can be "safely leaked". All the current solutions to the problem — including proactive secret sharing [27], all-or-nothing transforms and exposure-resilient functions [7] — always suffered from non-trivial losses in both space and time in order to achieve varying levels of exposure-resilience. It was, therefore, informally believed that these losses are inevitable in every reasonable application, since a "natural" representation $A$ is unlikely to offer any exposure-resilience. Perhaps surprisingly, we show this belief is false for the elegant "hierarchical identity-based encryption" (*HIBE*) of Gentry and Silverberg [16], which is the* only *known fully functional* HIBE *up to date. Specifically, we show that the natural secret key representation for the* HIBE *of [16] admits a simple and efficient refresh operation, which offers very high level of exposure-resilience, while incurring absolutely no space or time losses for decryption. We argue that this simple fact is quite powerful from a key storage security perspective, is highly applicable for such tasks as threshold decryption, and that it further makes* HIBE *a much more attractive alternative in various real life scenarios. On a philosophical level, while previous techniques [7] protected against gradual key exposure in a generic way, oblivious to the application, we show that in certain situations one might achieve much better parameters by concentrating on the application at hand.*



**Keywords:** cryptographic key storage, key storage protection, gradual key exposure, exposure resilience, key redundancy, hierarchical id-based encryption, bilinear Diffie-Hellman.


## 1 Introduction

A great deal of cryptography can be seen as finding ways to leverage the possession of a small but totally secret piece of knowledge (a key) into the ability to perform many useful and complex actions: from encryption and decryption to identification and message authentication. But what happens if our most *basic* assumption breaks down — that is, if the secrecy of our key becomes partially compromised?

Indeed, exposure of secret keys is perhaps the most debilitating attack on a cryptosystem since it typically implies that all security guarantees are lost. This problem is emerging as an ever-greater threat as cryptographic primitives are deployed on inexpensive, lightweight, and mobile devices; in these cases, it is typically much easier for an adversary to break into the device and obtain the secret keys than to crack the computational assumptions on which the system is based. Clearly, concerns about key exposure must be addressed in a satisfactory manner by the research community.

Recognizing the need to address these concerns, a long line of research has focused on dealing with the threat of key exposure. Methods to prevent key exposure entirely (e.g., by using tamper-resistant devices) seem cost-prohibitive and impractical for most common applications. Thus, much research has focused on making key exposures more difficult, or, alternately, minimizing the damage when (partial) key exposure occurs while utilizing regular computing devices and memory modules of servers that hold keys.

Two classes of methods exist to deal with this problem: those based on some form of *key evolution*, and those based on some form of *secret sharing* (and the combination of the two). The approach of key evolution [2] assumes that the timeline is divided into different periods, and a *different* secret key is used from one period to the next. This somewhat recent approach has already led to many useful notions, including those of forward-secure [3, 24, 1, 20, 26, 8], key-insulated [11, 12, 4] and intrusion-resilient [21, 13] cryptosystems. While very powerful, the disadvantage of this approach is the need to introduce "global time" and the issue of what to do with documents produced outside of the "current" period.

The last, older approach of secret sharing [28, 5, 25] typically does not change the secret over time, but rather stores the secret in a "redundant" form, such that the exposure of most (but not all) of such a representation still guarantees the security of the actual, "embedded" secret. We will call this property "exposure-resilience" from now on. The secret sharing approach has led to many applications, including the development of threshold [10, 9], proactive [27, 17] and exposure-resilient [7, 14] cryptography. One of the main disadvantages of this approach, though, is the fact that the new "exposure-resilient" representation of the secret is typically longer than the actual secret, and working with this "redundant" representation typically incurs a large loss of efficiency. Moreover, when the secret is split among many servers, special distributed protocols have to be designed to jointly perform the needed set of operations like signing or decrypting. These inefficiencies are believed, and usually are, inevitable, since it is unlikely that a "natural" representation of the secret offers any level of exposure-resilience.

## 1.1 Our Contribution

Surprisingly, we show that the above belief may sometimes be false. Specifically, we show that the only fully functional implementation of *hierarchical identity-based encryption* (HIBE), due to [16], naturally offers very high level of exposure-resilience. We recall that HIBE is a natural and very powerful extension of a regular identity-based encryption (which was originally formalized by Shamir [29] and recently solved by Boneh and Franklin [6]). Intuitively, HIBE allows to organize the users into a tree hierarchy. Each user gets the secret key from its parent in the hierarchy (and all the users share a few global parameters). Now, anybody can encrypt message to any given user by only knowing *its position in the hierarchy*. In particular, no "public key" of the user is needed, only user's identity and the "global public key" are used for encryption! The concept of HIBE was recently introduced by Horwitz and Lynn [18], but the only fully functional implementation is due to Gentry and Silverberg [16]. In this implementation, each user at "depth" $t$ has $t$ pieces of secret information. We show that any $t-1$ of these pieces give no information to the adversary, and therefore do not have to be carefully protected (thus reducing the requirement for secure storage). Moreover, we show that each user can easily perform (by itself) periodic *refreshes* of its secret key. Each such refresh is oblivious to the outside world, as the new key is as functional as the old one. However, it completely randomizes any $t-1$ out of $t$ shares of the user's secret key.

Our finding is simple, yet it has several natural and powerful applications in the area of cryptographic key storage. First, it gives natural protection against the *gradual key exposure* problem introduced by [7]. In this problem, the secret key is assumed to be slowly compromised over time, so that more and more information about a secret key is eventually leaked. As long as the user refreshes its HIBE key frequently enough, no security is lost. Secondly, it shows that the *secure* storage for the HIBE of [16] is the same as in the regular IBE of [6], since all but one pieces of the secret can be made public. Thirdly, it leads to more efficient implementations of threshold and proactive implementations of HIBE. Namely, rather than share all $t$ pieces of its secret, we show that the user can share only one piece among some number of servers, which results in much more efficient threshold decryption protocols. Finally, we believe that our observation will be useful in many more complex schemes which are based on the HIBE of [16]. Indeed, our technique was recently used by [13] in constructing the first intrusion-resilient encryption scheme.

We note that from a technical point of view, the crux of our contribution is carefully defining the adversarial setting and proving the security of the refresh procedure within this setting.

From a systems design perspective, what we show is that the current HIBE possesses a real advantage in the area of "cryptographic key storage protection." In fact, storing its keys may require much less "secure memory" while replacing the rest of the key storage area with memory modules that are "safe" or "trusted" but not necessarily concealing. This may ease the cost and design effort of an architecture for cryptographic key storage. From an engineering practice point of view, when designing a real life cryptographic system, we note that the issue of protection of keys (and their memory modules) should always be considered in the design process (and should not be left as an afterthought design). Thus, the notions of "key exposure" and key protection in general, have to be considered in the design. What is shown here is that while, theoretically, HIBE may be considered a solution which requires "heavy" keying storage (and thus dis-advantageous in many respects), it actually becomes a much more attractive solution when one has to cope with potential partial key exposure by the key storage media.

## 2 Cryptographic Assumptions

The security of the HIBE of [16] is based on the difficulty of the bilinear Diffie-Hellman (BDH) problem as recently formalized by Boneh and Franklin [6] (see also [23, 22]). We review the relevant definitions as they appear in [6]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order $q$, where $\mathbb{G}_1$ is represented additively and $\mathbb{G}_2$ is represented multiplicatively. We use a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ for which the following hold:

1. The map $\hat{e}$ is *bilinear*; that is, for all $P_0, P_1 \in \mathbb{G}_1$ and all $x, y \in \mathbb{Z}_q$ we have
$$\hat{e}(xP_0, yP_1) = \hat{e}(yP_0, xP_1) = \hat{e}(P_0, P_1)^{xy} \tag{1}$$
2. There is an efficient algorithm to compute $\hat{e}(P_0, P_1)$ for any $P_0, P_1 \in \mathbb{G}_1$.
3. The map is non-degenerate, i.e. $\hat{e}(P, P) \neq 1$ for some $P \in \mathbb{G}_1$.

A *BDH parameter generator* $\mathcal{I}$ is a randomized algorithm that takes a security parameter $1^k$, runs in polynomial time, and outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and a map $\hat{e}$ satisfying the above conditions. We define the *BDH problem with respect to* $\mathcal{I}$ as the following: given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ output by $\mathcal{I}$ along with random $P, aP, bP, cP \in \mathbb{G}_1$, compute $\hat{e}(P, P)^{abc}$. We say that $\mathcal{I}$ *satisfies the BDH assumption* if the following is negligible (in $k$) for all PPT algorithms $A$:

$$\Pr[\quad (\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{I}(1^k); P \leftarrow \mathbb{G}_1; a, b, c \leftarrow \mathbb{Z}_q : \\ A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc} \quad ]$$

We note that BDH parameter generators for which the BDH assumption is believed to hold can be constructed from Weil and Tate pairings associated with supersingular elliptic curves or Abelian varieties. As our results do not depend on any specific instantiation, we refer the interested reader to [6] for details.

## 3 Hierarchical ID-Based Encryption

Recall, HIBE allows to organize the users into a tree hierarchy. Each user gets the secret key from its parent in the hierarchy (and all the users share a few global parameters). Now, anybody can encrypt message to any given user by only knowing *its position in the hierarchy*. In particular, no "public key" of the user is needed! Below we briefly describe the functionality of general HIBE, followed by the specific HIBE scheme of [16].

### 3.1 General HIBE

Each user of the system is identified by its position in the hierarchy, $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, also referred as its *ID-tuple*. This means that the user is located at level $t$ and its ancestors, starting from the parent down to the root, are $(\mathrm{ID}_1, \ldots, \mathrm{ID}_{t-1})$, …, $(\mathrm{ID}_1)$, root. A HIBE is specified by five efficient randomized algorithms described below: Root Setup, Lower-level Setup, Extraction, Encryption and Decryption.

- **Root Setup:** Given a security parameter $K$, it returns the global public key $PK$ available to everybody, and the master secret key $SK^*$ available to the super-user root.
- **Lower-level Setup:** Not important for us.
- **Extraction:** Any user with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ ($t = 0$ corresponds to root) may compute, using its secret key, the secret key for any of its children with ID-tuple $(\mathrm{ID}_1, \ldots \mathrm{ID}_t, \mathrm{ID}_{t+1})$.
- **Encryption:** Given the global public key $PK$, the recipient's ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ and a message $M$, it returns the encryption $C$ of $M$ intended for user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$.
- **Decryption:** Given the ciphertext $C$ and its secret key, the user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ can recover the plaintext $M$.

As expected, the correctness property states that the user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ should always correctly recover messages encrypted for him.

SECURITY. Intuitively, security of HIBE states that only the designated user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ and its ancestors can decrypt messages sent to this user, while no other user of the system can. We briefly define it more formally, referring the reader to [16] for a more detailed description. We only describe the basic semantic security since dealing with chosen ciphertext security presents no additional problems using the technique of Fujisaki and Okamoto [15].

At the beginning of the game, the adversary is given $PK$. At any point of the game, the adversary is also given oracle access to the extraction procedure. Namely, given any ID-tuple of adversary's choice, the adversary will learn the secret key of this user. At some point, the adversary chooses an ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ and two message $M_0, M_1$. A random bit $b$ is chosen and the adversary gets the hierarchical encryption $C$ of $M_b$ for user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. At the end, the adversary has to output a guess $b'$. Adversary wins if $b' = b$ and the adversary did not call the extraction oracle on $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ for any $i \leq t$; i.e., no ancestor of $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ was "corrupted". The HIBE is semantically secure if no PPT adversary can win with probability non-negligibly more than $1/2$. Due to the technical reason, Gentry and Silverberg [16] got asymptotically good bounds for their scheme only for the case of so called "non-adaptive" adversary. This adversary is the same as the one we consider except that it chooses its "target" $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ at the beginning of its run (i.e., independently of its extraction queries). To get the same good results for our extension, we will also concentrate on such "non-adaptive" adversary (of course, our results extend to adaptive adversary, but in this case we get the same poor exact security as [16]).

### 3.2 The HIBE of Gentry and Silverberg [16]

We can now describe the scheme of [16] using the notation developed in Section 2.

- **Root Setup:** Runs $\mathcal{I}(1^K)$ to get $\mathbb{G}_1, G_2, \hat{e}$, picks a random $s_0 \in \mathbb{Z}_q$, $P_0 \in \mathbb{G}_1$, sets $Q_0 = s_0 P_0$, and outputs $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2)$, $SK^* = s_0$. Here $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ are cryptographic hash functions, modeled as random oracles (i.e., they output a truly random string on every input), and $n$ is the length of the messages encrypted.
- **Extraction:** Every user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ at level $t \geq 0$ will have a secret point $S_t \in \mathbb{G}_1$ (see be-

low; we assume that the root has $S_0 = 0_{\mathbb{G}_1}$), and $(t-1)$ "translation points" $Q_1 \dots Q_{t-1} \in \mathbb{G}_1$ (notice, $Q_0$ is in the public key). Recursively, to assign the secret key to its child $\mathrm{ID}_{t+1}$, the parent $(\mathrm{ID}_1, \dots, \mathrm{ID}_t)$ computes $P_{t+1} = H_1(\mathrm{ID}_1 \dots \mathrm{ID}_{t+1}) \in \mathbb{G}_1$, picks a random $s_t \in \mathbb{Z}_q$, sets the child's secret point $S_{t+1} = S_t + s_t P_{t+1}$, the child's final translation point $Q_t = s_t P_0$, and sends to the child the values $S_{t+1}, Q_t$ together with its own $t-1$ translation points $Q_1 \dots Q_{t-1}$. Unwrapping the notation, the child's secret key is

$$(S_{t+1} = \sum_{i=1}^{t+1} s_{i-1} P_i, Q_1 = s_1 P_0, \dots, Q_t = s_t P_0)$$

- **Encryption:** To encrypt a message $M \in \{0,1\}^n$ for $(\mathrm{ID}_1, \dots, \mathrm{ID}_t)$ using public value $Q_0$, compute $P_i = H_1(\mathrm{ID}_1 \dots \mathrm{ID}_i) \in \mathbb{G}_1$ for all $1 \le i \le t$, choose a random $r \in \mathbb{Z}_q$, set $g = \hat{e}(Q_0,\ rP_1) \in \mathbb{G}_2$ and return

$$C = [rP_0,\ M \oplus H_2(g),\ rP_2, \dots,\ rP_t] \quad (2)$$

  Intuitively, the first two components correspond to the standard "ElGamal"-like encryption for the top-level user $(\mathrm{ID}_1)$. Unfortunately, user $(\mathrm{ID}_1, \dots, \mathrm{ID}_t)$ cannot quite decrypt it using its "translated" secret point $S_{t+1}$, so additional values $rP_2, \dots, rP_t$ are given. Combining them with secret translation points $Q_1 \dots Q_{t-1}$, the message $M$ is recovered. This is described below.

- **Decryption:** To decrypt $C = [U_0, V, U_2, \dots, U_t]$ using $S_t$ and $Q_1 \dots Q_{t-1}$, set $f_0 = \hat{e}(U_0, S_t)$, $f_i = \hat{e}(Q_{i-1}, U_i)$ for $2 \le i \le t$ and output

$$M = V \oplus H_2\left(\frac{f_0}{f_2 \dots f_t}\right) \quad (3)$$

  To see the correctness of the decryption, notice that

$$\begin{aligned} f_0 &= \hat{e}(U_0, S_t) \\ &= \hat{e}(rP_0,\ \sum_{i=1}^{t} s_{i-1} P_i) \\ &= \prod_{i=1}^{t} \hat{e}(rP_0,\ s_{i-1} P_i) \\ &\stackrel{(1)}{=} \prod_{i=1}^{t} \hat{e}(s_{i-1} P_0,\ rP_i) \\ &= \hat{e}(Q_0,\ rP_1) \cdot \prod_{i=2}^{t} \hat{e}(Q_{i-1},\ U_i) \\ &= g \cdot f_2 \cdots f_t \end{aligned}$$

## 4 Exposure-Resilience For Free

Notice, the secret key of a user at level $t$ is of the form

$$S_t = \sum_{i=1}^{t} s_{i-1} P_i,\ Q_1 = s_1 P_0,\ \dots\ ,\ Q_{t-1} = s_{t-1} P_0,$$

where $P_0, P_1, \dots P_t \in \mathbb{G}_1$ are all random (the latter since $H_1$ is a random oracle), and so are $s_0 \dots s_{t-1} \in \mathbb{Z}_q$. Among these last values, only $s_0$ is "fixed" by the public key $Q_0 = s_0 P_0$; the values $s_1, \dots, s_{t-1}$ can be *arbitrary* and the scheme will still work. This suggests the following very simple procedure to refresh the current secret key $(S_t, Q_1, \dots, Q_{t-1})$.

- **Refresh:** Pick random $s'_1, \dots s'_{t-1} \in Z_q$, and reset:

$$\begin{aligned} S_t &:= S_t + \sum_{i=2}^{t} s'_{i-1} P_i, \\ Q_i &:= Q_i + s'_i P_0; \quad \text{for } 1 \le i < t \end{aligned}$$

It is easy to see that the new key is as functional as the old one, requires no extra storage or decryption time, but any $(t-1)$ out of $t$ "old values" (resp. "new values") $S_t, Q_1, \dots, Q_{t-1}$ reveal absolutely no information about any of the "new values" (resp. "old values") due to the fresh randomness of $s'_1 \dots s'_{t-1}$. Also, we will assume that each user immediately performs a refresh operation upon receiving his key from its parent, so that any $(t-1)$ user's shares are random and completely independent from all the secret keys of its ancestors. We then show the following result:

**Theorem 1** *Under the* BDH *assumption, our* HIBE *scheme remains semantically secure for any user at level $t > 1$, even if he leaks any $(t-1)$ out of its $t$ secret values between every pair of successive refreshes.*

**Proof:** Before proceeding, let us first extend the definition of semantic security to model the repeated exposure of $(t-1)$ out $t$ secret shares for a given user. In addition to his usual capabilities, the adversary $A$ can pick any user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ and learn any $(t-1)$ out of $t$ pieces of its secret key, without declaring this user "corrupted". Moreover, the adversary can also ask any user to refresh its secret key, after which it is allowed to again learn any $(t-1)$ out of $t$ "new" shares of this user's secret key. However, we already argued that any $(t-1)$ old/new values reveal no information about any of the new/old values. Thus, we can assume that each user is asked to reveal its $(t-1)$ shares at most ones. Since we consider non-adaptive adversaries, let $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ be the specific user the adversary will be targeting. In our simulation, we will explicitly know the secret keys of all the users beside the ancestors $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ (for $i \leq t$) of the target user, so all the corruption requests for such users will be easy to handle (see below). Thus, we will assume without loss of generality that the adversary wants to learn all but one share of the secret keys for all ancestors of $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. Notice, however, since the adversary $A$ is not allowed to corrupt any of the ancestors $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ of $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, $A$ gets a challenge only for the target user, and each ancestor $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ immediately performed a key refresh operation, the $i-1$ shares of any such ancestor are just $(i-1)$ totally random and independent group elements. Thus, they give no information to the adversary.

To summarize, we may reduce our game to the following. The adversary chooses the target user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, learns some $(t-1)$ out of its $t$ secret shares, arbitrarily corrupts any users besides the ancestors of $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ (as we said, in our simulation this will be trivial), chooses $M_0$ and $M_1$, gets the challenge, and has to guess which message was encrypted for $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. So assume some $A$ succeeds in this game with probability $1/2 + \varepsilon$. We construct $B$ which succeeds in breaking the BDH assumption with probability roughly $\Omega(\varepsilon / q_{H_2})$, where $q_{H_2}$ is the number of hash queries asked to the random oracle $H_2$. For simplicity of notation, we only consider the case when the values $Q_1 \ldots Q_{t-1}$ are leaked to $A$ (while $S_t$ is secure). The other case (when one of the $Q_i$'s is secure) is completely analogous.

So assume $B$ is given an input $P_0, s_0 P_0, \alpha_1 P_0, r P_0$ and tries to compute the value $g = \hat{e}(P_0, P_0)^{s_0 r \alpha_1}$ (the strange choice of notation will be clear soon). $B$ also knows the user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ that $A$ is going to target. $B$ will set the public key $PK = (P_0, Q_0 = s_0 P_0)$ and give it to $A$. It will also set $P_1 = H_1(\mathrm{ID}_1) = \alpha_1 P_0$ (where it does not know $\alpha_1$), choose random $\alpha_2, \ldots, \alpha_t$ and set $P_i = H_1((\mathrm{ID}_1, \ldots, \mathrm{ID}_i)) = \alpha_i P_0$ for $2 \leq i \leq t$. $B$ also chooses random $s_1, \ldots, s_{t-1}$ and sets the translation points $Q_1 = s_1 P_0, \ldots, Q_{t-1} = s_{t-1} P_0$, which it also gives to the adversary as $(t-1)$ shares of the user's secret key. Next, to $H_1$ queries of the form $(\mathrm{ID}'_1)$, where $\mathrm{ID}'_1 \neq \mathrm{ID}_1$, $B$ chooses a random $a$ and responds with $aP_0$ (remembering $a$). Notice, this ensures that $B$ "knows" the secret key of $\mathrm{ID}'_1$ (and, hence, of all its descendants) as $s_0 H_1(\mathrm{ID}'_1) = s_0 a P_0 = a Q_0$. Next, for inputs $(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-1}, \mathrm{ID}'_i)$ to $H_1$, where $\mathrm{ID}'_i \neq \mathrm{ID}_i$ (and $2 \leq i \leq t+1$), $B$ picks random value $a$ and responds with $(aP_0 - s_{i-1}^{-1} P_1)$ (remembering $a$; in case $i = t+1$, a fresh random $s_t$ is chosen as well). Notice also that the returned value is indeed random, since $a$ is random. We claim that this ensures that $B$ "knows" a legal secret key of $(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-1}, \mathrm{ID}'_i)$ (and thus, of its descendants). Indeed, we can set the secret point to

$$S'_i = a s_{i-1} Q_0 + \sum_{j=2}^{i-1} s_{j-1} P_j$$

and translation points to earlier defined $Q_1, \ldots, Q_{i-2}$, followed by $Q'_{i-1} = s_{i-1} Q_0$ (which is also equal to $s_{i-1} s_0 P_0$, so that the supposed coefficient is $s_{i-1} s_0$; this coefficient is unknown to $B$ since $B$ does not know $s_0$, but this is fine as long as the equation below holds). Indeed, the "supposed" value of the secret point $S'_i$ corresponding to the translation points $Q_1 = s_1 P_0, \ldots, Q_{i-2} = s_{i-2} P_0, Q'_i = s_{i-1} s_0 P_0$ should have been

$$s_0 P_1 + \sum_{j=2}^{i-1} s_{j-1} P_j + (s_{i-1} s_0) H(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-1}, \mathrm{ID}'_i)$$

Thus, we we only need to check that the part of the secret point $a s_{i-1} Q_0$ is consistent with its "supposed" value $s_0P_1 + (s_{i-1}s_0)H(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-1}, \mathrm{ID}'_i)$. But, indeed,

$$\begin{aligned} s_0P_1 + (s_{i-1}s_0)H(\mathrm{ID}_1, \ldots, \mathrm{ID}_{i-1}, \mathrm{ID}'_i) &= \\ s_0P_1 + s_{i-1}s_0(aP_0 - s_{i-1}^{-1}P_1) &= \\ as_{i-1}(s_0P_0) &= \\ as_{i-1}Q_0 \end{aligned}$$

so the secret key is valid. Thus, $B$ can easily produce valid secret keys for any ID-tuple different from the ancestors of the target user $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, which means that $B$ can easily handle all the extraction queries of $A$ (of course, $B$ will return refreshed versions of the secret keys in this case since this is what $A$ expects; notice also that all other random oracle queries to $H_1$ are answered at random).

When $A$ outputs messages $M_0$ and $M_1$, $B$ set $U_0 = rP_0$ (remember, $r$ is unknown, so it takes this value from the BDH input). For $2 \leq i \leq t$, $B$ now has to set the value $U_i = rP_i = r\alpha_iP_0 = \alpha_iU_0$, which it can easily do as it knows the $\alpha_i$'s. Finally, $B$ picks a truly random $V$, and outputs challenge ciphertext $[U_0, V, U_2, \ldots, U_t]$. Notice, $V$ was supposed to be equal to $M_b \oplus H_2(g)$, where

$$g = \hat{e}(Q_0, rP_1) = \hat{e}(s_0P_0, r\alpha_1P_0) = \hat{e}(P_0, P_0)^{s_0r\alpha_1}$$

which is exactly our goal for BDH. Since $H_2$ is a random oracle, the only way $A$ can get any advantage is if it queried $H_2$ (which, by the way, $B$ always simulates by returning a random value) on input $g$ with non-negligible probability (actually, probability at least $\varepsilon$). Thus, at the end of $A$'s run it suffices for $B$ to output a random input to $H_2$, which makes $B$ succeed in the BDH problem with probability $\Omega(\varepsilon/q_{H_2})$, as claimed. $\square$

### 4.1 Consequences and Implications

As a corollary, even though the user at level $t$ needs to store $t$ values, only *one* of these values (e.g., $S_t$) has to be kept secret (e.g., on a smartcard); the other $(t-1)$ values are needed for functionality, but not for the security, and can be kept insecurely (or even publicly!).

In particular, to distribute the decryption process, the user can secret share (using Shamir's secret sharing [28] over $\mathbb{G}_1$) only the value $S_t$, keeping $Q_1, \ldots, Q_{t-1}$ locally. When obtaining ciphertext $C = [U_0, V, U_2, \ldots, U_t]$, the user can compute the values $f_i = \hat{e}(Q_{i-1}, U_i)$ (for $2 \leq i \leq t$) locally, and only needs servers' help in computing $f_0 = \hat{e}(U_0, S_t)$. However, the servers can now jointly compute $\hat{e}(U_0, S_t)$ by simply performing standard Lagrange interpolation (using their polynomial shares and the linearity of $\hat{e}$). Thus, we get threshold decryption for the user at level $t$ with the same communication complexity as that for the user of level 1. The only dependence on $t$ comes in the *local* computation by the user. This shows that the "real" dependence on the level in the hierarchy is very minimal when distributing the HIBE of [16].

Also, to refresh the value $S_t$ for proactive security, the user locally updates $Q_1 \ldots, Q_{t-1}$ by adding random $Q'_1 = s'_1P_0, \ldots, Q'_{t-1} = s'_{t-1}P_0$, and then secret shares (again, using polynomial secret sharing) the corresponding "added value" $S'_t = \sum_{i=1}^{t-1} s'_iP_{i+1}$ among the servers. The servers then locally add the received share of $S'_t$ to the old share of $S_t$, thus obtaining a fresh, totally random sharing.

Finally, we recall that the implications to the design of protected cryptographic key storage systems are discussed at the end of Section 1.1.

## References


[1] M. Abdalla and L. Reyzin. A New Forward-Secure Digital Signature Scheme. Asiacrypt 2000.

[2] R. Anderson. Two Remarks on Public-Key Cryptology. Invited lecture, CCCS 1997. URL: `http://www.cl.cam.ac.uk/users/rja14/`.

[3] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. Crypto 1999.

[4] M. Bellare and A. Palacio. Protecting against Key Exposure: Strongly Key-Insulated Encryption with Optimal Threshold. URL: `http://eprint.iacr.org`.

[5] G. Blackley. Safeguarding Cryptographic Keys. In *Proc. of AFIPS 1979 National Computer Conference*, 1979.

[6] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. Crypto 2001. Full version to appear in *SIAM J. Computing* and available at `http://eprint.iacr.org/2001/090/`.

[7] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-Or-Nothing-Transforms. Eurocrypt 2000.

[8] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. Preliminary version available at `http://eprint.iacr.org/2002/060/`.

[9] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC 1994.

[10] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto 1989.

[11] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public-Key Cryptosystems. Eurocrypt 2002.

[12] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. PKC 2003.

[13] Y. Dodis, M. Franklin, J. Katz, A. Miajyi and M. Yung. Intrusion-Resilient Public-Key Encryption. RSA 2003.

[14] Y. Dodis, A. Sahai and A. Smith. On Perfect and Adaptive Security in Exposure-Resilient Cryptography. EuroCrypt 2001.

[15] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. Crypto 1999.

[16] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. Asiacrypt 2002.

[17] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Public-Key and Signature Schemes. CCCS 1997.

[18] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. Eurocrypt 2002.

[19] G. Itkis. Intrusion-Resilient Signatures: Generic Constructions, or Defeating a Strong Adversary with Minimal Assumptions. SCN 2002.

[20] G. Itkis and L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. Crypto 2001.

[21] G. Itkis and L. Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. Crypto 2002.

[22] A. Joux. The Weil and Tate Pairing as Building Blocks for Public-Key Cryptosystems. ANTS 2002.

[23] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups. Manuscript, Jan. 2001. Available at `http://eprint.iacr.org`.

[24] H. Krawczyk. Simple Forward-Secure Signatures From any Signature Scheme. CCCS 2000.

[25] H. Krawczyk. Secret Sharing Made Short. Crypto 1993.

[26] T. Malkin, D. Micciancio, and S. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number of Time Periods. Eurocrypt 2002.

[27] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC 1991.

[28] A. Shamir. How to share a secret. In *Communic. of the ACM*, 22:612-613, 1979.

[29] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. Crypto 1984.