A preliminary version of this paper appears in the proceedings of the 10th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2007, Lecture Notes in Computer Science Vol. ???, Tatsuaki Okamoto and Xiaoyun Wang eds, Springer Verlag, 2007. This is the full version.

# Chosen-Ciphertext Secure Key-Encapsulation Based on Gap Hashed Diffie-Hellman

Eike Kiltz

CWI Amsterdam, The Netherlands

**Abstract**

We propose a practical key encapsulation mechanism with a simple and intuitive design concept. Security against chosen-ciphertext attacks can be proved in the standard model under a new assumption, the Gap Hashed Diffie-Hellman (GHDH) assumption. The security reduction is tight and simple.

Secure key encapsulation, combined with an appropriately secure symmetric encryption scheme, yields a hybrid public-key encryption scheme which is secure against chosen-ciphertext attacks. The implied encryption scheme is very efficient: compared to the previously most efficient scheme by Kurosawa and Desmedt [Crypto 2004] it has 128 bits shorter ciphertexts, between 25-50% shorter public/secret keys, and it is slightly more efficient in terms of encryption/decryption speed. Furthermore, our scheme enjoys (the option of) public verifiability of the ciphertexts and it inherits all practical advantages of secure hybrid encryption.

**Keywords:** Chosen-ciphertext security, Public-key encryption, key encapsulation.

# Contents

# 1 Introduction

One of the main fields of interest in cryptography is the design and the analysis of the security of encryption schemes in the public-key setting (PKE schemes). In this work our goal is to provide schemes for which we can provide theoretical proofs of security (without relying on heuristics such as the random oracle), but which are also efficient and practical.

Key Encapsulation. Instead of providing the full functionality of a public-key encryption scheme, in many applications it is sufficient to let sender and receiver agree on a common random session key. This can be accomplished with a *key encapsulation mechanism* (KEM) as formalized by Cramer and Shoup [43, 17]. In this protocol a sender (knowing the receivers public key) runs an encapsulation algorithm to produce a random session key together with a corresponding ciphertext. This ciphertext is sent (over a potentially insecure channel) to the receiver, who (using his secret key) can uniquely reconstruct the session key using a decapsulation algorithm. In the end both parties share a common random session key. An appropriate notion of security (against *chosen-ciphertext attacks*) says that, roughly, not even an active eavesdropper (interacting with a decapsulation oracle that allows him to obtain session keys corresponding to ciphertexts of his choosing) can learn any information about the random session key just given the ciphertext. Chosen-ciphertext security [41] is a much stronger security requirement than *chosen-plaintext security* [4], where in the latter an attacker is not given access to the decryption oracle. After the execution of the protocol the random session key may now be used for arbitrary symmetric-key operations such as a symmetric encryption scheme. If both, the KEM and the symmetric primitive, are secure against chosen-ciphertext attacks then composition theorems are used to obtain the same security guarantees for the hybrid protocol.

In this work we are interested in designing key encapsulation mechanisms that are both efficient and provably secure with respect to a reasonable intractability assumption. To motivate our approach we start with some history on key encapsulation.

Diffie-Hellman Key Encapsulation. In the Diffie-Hellman key encapsulation mechanism [18] the receiver's public key consists of the group element $g^x$ (we assume a commutative cyclic group of prime order and generator $g$ to be given), the secret key of the random index $x$. Key encapsulation is done by computing the ciphertext as $g^y$ for random $y$; the corresponding session key is the group element $g^{xy} = (g^x)^y$ (and therefore called *Diffie-Hellman Key*). This key is recovered from the ciphertext by the possessor of the secret key $x$ by computing $g^{xy}$ as $(g^y)^x$. This simple KEM can be proved secure against chosen-plaintext attacks under the Decisional Diffie-Hellman (DDH) assumption. The DDH assumption states, roughly, that the two distributions $(g^x, g^y, g^{xy})$ and $(g^x, g^y, g^z)$ for random indices $x, y, z$ are computationally indistinguishable. From an application point of view there is one major weakness in the original Diffie-Hellman key encapsulation. Namely, the session key output by the scheme is a group element, whereas in practice we rather want the session key to be a bit string. A straightforward way to overcome this shortcoming is to feed the Diffie-Hellman key $g^{xy}$ to a hash function $\mathsf{H}$ with binary image to obtain the session key $\mathsf{H}(g^{xy})$. Now the session key is a bit-string and may be used as a key to perform some symmetric operation. This key encapsulation scheme can be proved secure against chosen-plaintext attacks under the *Hashed Diffie-Hellman* (HDH) assumption, as formalized in [1]. The HDH assumption (relative to a hash function $\mathsf{H}$) states, roughly, that the two distributions $(g^x, g^y, \mathsf{H}(g^{xy}))$ and $(g^x, g^y, R)$ for random indices $x, y$ and a random bit-string $R$ (of appropriate length) are computational indistinguishable. Under the HDH assumption, Hashed Diffie-Hellman can be proven secure against chosen-plaintext attacks (IND-CPA). For various reasons, the stronger notion of chosen-ciphertext (IND-CCA) security [41] has emerged as the "right" notion of security for key encapsulation and encryption. Hashed Diffie-Hellman will be our starting point and the goal will be to modify the scheme in order to obtain security against chosen-ciphertext attacks

under a reasonable intractability assumption.

Our Construction. We modify the Hashed Diffie-Hellman key encapsulation in order to obtain a KEM which is provably secure against chosen-ciphertext attacks under the *Gap Hashed Diffie-Hellman assumption* (to be introduced later). Our main idea is to add some redundant information to the ciphertext of the Hashed Diffie-Hellman key encapsulation. This information is used to check if a given ciphertext was properly generated by the encapsulation algorithm (and hence is "consistent"); if the ciphertext is consistent then decapsulation returns the session key, otherwise it simply rejects. Our scheme's security relies on the Gap Hashed Diffie-Hellman (GHDH) assumption which states that, roughly, the two distributions $(g^x, g^y, \mathsf{H}(g^{xy}))$ and $(g^x, g^y, R)$ are hard to distinguish even relative to a "Diffie-Hellman oracle" that distinguishes $(g^x, g^y, g^{xy})$ from $(g^x, g^y, g^z)$. Here the term "gap" stems from the fact that there is a gap between the Decisional and the Computational version of the Diffie-Hellman problem: the computational problem is hard to solve even though the corresponding decisional problem is easy.

Main Results. Our main result shows that our key encapsulation mechanism (given in Section 4) is secure against chosen-ciphertext attacks assuming the GHDH assumption holds. The scheme has very short ciphertexts (2 groups elements or approximately 512 bits for 128 bits security) and its security reduction is *tight*. Furthermore, when our scheme gets instantiated in gap-groups a given ciphertext can get checked for consistency solely based on the knowledge of the public key. This feature (sometimes called "public verifiability of the ciphertext") has proved very useful, e.g. for building a chosen-ciphertext secure threshold encapsulation scheme [14]. Furthermore, we show that our framework extends to KEMs based on the Gap Hashed Multi Diffie-Hellman (GHMDH) assumption, a natural generalization of GHDH with potentially stronger security properties. The GHMDH assumption states that given many independent Diffie-Hellman instances $(g_i, h_i, g_i^{r_i})_{1 \le i \le \ell}$, a certain fixed predicate $\mathsf{H} : \mathbb{G}^{\ell^2} \to \mathbb{G}$ evaluated on the $\ell^2$ possible (hidden) Diffie-Hellman keys $(h_i)^{r_j}$ $(1 \le i, j \le \ell)$ is indistinguishable from a random element, even relative to a DDH oracle. The GHMDH assumption in particular includes (a paring-free variant of) the Gap Linear Diffie-Hellman (GLDH) assumption [12].

The GHDH assumption is not a "random-oracle assumption". We emphasize that this scheme can be proved secure under reasonable intractability assumptions, without resorting to either the random oracle heuristic, and without using "interactive intractability assumptions" like ODH as in done in [1]. At first glance one may argue that assuming the hashed key $\mathsf{H}(g^{xy})$ to be indistinguishable from a random string even though we can efficiently distinguish $g^{xy}$ from a random group element sounds quite unreasonable and that, in a sense, hardness falls back on "random-oracle-like" properties of the hash function, similar to the full-domain-hash (FDH) principle. However, this intuition is not true as we will further elaborate in Section 3. To put more confidence in the GHDH assumption we will show that it (provably) holds in generic groups assuming the hash function $\mathsf{H}$ is "weakly one-way". The latter result basically means that the GHDH assumption depends on the hardness of computing the Diffie-Hellman key plus the fact that given only $\mathsf{H}(g^{xy})$ it is hard to recover sufficient information on the Diffie-Hellman key $g^{xy}$. We will argue that the well known and often employed Bilinear Diffie-Hellman (BDH) assumption [13] can in fact be seen as a special (algebraic) instantiation of the GHDH assumption. More precisely, using the specific algebraic hash function $\mathsf{H}(X) := \hat{e}_Z(X)$, where $\hat{e}_Z(X) := \hat{e}(X, Z)$ is a bilinear mapping for fixed $Z = g^z$ (but chosen uniformly at setup), we get $\mathsf{H}(g^{xy}) = \hat{e}(g^{xy}, g^z) = \hat{e}(g, g)^{xyz}$ and GHDH actually gets BDH (here the output of $\mathsf{H}$ is a group element, not a binary string). In this context, GHDH instantiated with a cryptographic hash function like SHA1 (we will propose further candidates for $\mathsf{H}$ leading to different flavors of GHDH) appears not to be a less reasonable assumption than the "standard" BDH assumption.

Related Work. Historically, Cramer and Shoup [16] presented the first really practical public-

key encryption scheme in the standard model which was later generalized to a KEM [43, 17]. More recently, Kurosawa and Desmedt came up with a direct hybrid encryption scheme [32] improving the performance of the original CS scheme both in computational efficiency and in ciphertext length. In their hybrid construction the symmetric scheme has to be secure in the sense of *authenticated encryption* [5] which is a strictly stronger security requirement than in the standard KEM/DEM hybrid paradigm [43], and in particular it necessarily adds 128 bits of redundancy to the symmetric ciphertext. The KD-KEM (i.e. the KEM part of the Kurosawa Desmedt hybrid encryption scheme) is similar to our KEM construction. In fact, the KD-KEM can be obtained from our KEM by (roughly) switching the symmetric key with one element from the ciphertext. Our scheme can be proved chosen-ciphertext secure whereas there exists a simple chosen-ciphertext attack against the KD-KEM [26]. We think that this is really a surprising fact since a small difference in the constellation of the ciphertexts seems to turn the scale when it comes to security of the two schemes.

An alternative group of schemes ("IBE-based schemes") is based on recent results [15, 30] observing that identity-based encryption (IBE) implies chosen-ciphertext secure encryption. The recent approach taken by Boyen, Mei, and Waters [14] was to improve efficiency of one particular instantiation [10] (based on the BDH assumption) obtained by the above IBE transformation. Similar results (with improved key-decapsulation and based on a different assumption) were also obtained independently by Kiltz [30]. All the encryption schemes constructed this way, however, so far remained less efficient than the reference scheme from Kurosawa-Desmedt. Our KEM constructions are related (and generalize) the KEMs obtained in [14, 30] and therefore fits best into the latter class of IBE-based [15, 30] schemes (even though they are not derived from any IBE scheme).

Discussion and Comparison. Our implied hybrid PKE scheme is more efficient than the "reference scheme" by Kurosawa and Desmedt [32]: it has "one MAC" shorter ciphertexts (by combining it with redundancy-free symmetric encryption [39, 24, 25]), between 25-50% shorter public/secret keys, and it is slightly more efficient in terms of encryption/decryption. However, an arguable disadvantage of our scheme is that security can only be proven on the new GHDH assumption, whereas security of the KD scheme provably relies on the well-established and purely algebraic DDH assumption. An extensive comparison with all known KEM/PKE schemes in the standard model is done in Table 1 in Section 7.

Recent Results. Recently, building on this work, Hofheinz and Kiltz [27] combined a variation of our scheme with symmetric authenticated encryption (and hence adding 128 bits redundancy to the ciphertexts) to obtain public-key encryption secure under the DDH assumption. Their technique also extends to the more general class of (Hashed) Multi Diffie-Hellman assumptions which can be seen as the "DDH-oracle free" variant of GHMDH.

# 2 Preliminaries

## 2.1 Notation

If $x$ is a string, then $|x|$ denotes its length, while if $S$ is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then $1^k$ denotes the string of $k$ ones. If $S$ is a set then $s \xleftarrow{\$} S$ denotes the operation of picking an element $s$ of $S$ uniformly at random. We write $\mathcal{A}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and letting $z$ be the output. We write $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs $(x, y, \ldots)$ and access to oracles $\mathcal{O}_1, \mathcal{O}_2, \ldots$, and letting $z$ be the output.

## 2.2 Public Key Encapsulation Mechanisms

A *public-key encapsulation* (KEM) scheme $\mathcal{KEM} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with key-space $\mathsf{KeySp}(k)$ consists of three polynomial-time algorithms. Via $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$ the randomized key-generation algorithm produces keys for security parameter $k \in \mathbb{N}$; via $(K, C) \xleftarrow{\$} \mathsf{Enc}(1^k, pk)$ a key $K \in \mathsf{KeySp}(k)$ together with a ciphertext $C$ is created; via $K \leftarrow \mathsf{Dec}(sk, C)$ the possessor of secret key $sk$ decrypts ciphertext $C$ to get back a key. For consistency, we require that for all $k \in \mathbb{N}$, and all $(K, C) \xleftarrow{\$} \mathsf{Enc}(1^k, pk)$ we have $\Pr[\,\mathsf{Dec}(C) = K\,] = 1$, where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$, and the coins of all the algorithms in the expression above.

**Definition 2.1** Formally, we associate to an adversary $\mathcal{A}$ the following experiment:

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\mathit{kem}\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k) \\
\quad (pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k) \\
\quad K_0^* \xleftarrow{\$} \mathsf{KeySp}(k)\ ;\ (K_1^*, C^*) \xleftarrow{\$} \mathsf{Enc}(pk) \\
\quad \delta \xleftarrow{\$} \{0,1\} \\
\quad \delta' \xleftarrow{\$} \mathcal{A}^{\mathsf{DecO}(\cdot)}(pk, K_\delta^*, C^*) \\
\quad \text{If } \delta \neq \delta' \text{ then return 0 else return 1}
\end{array}
$$

where on input $C$ the oracle $\mathsf{DecO}(sk, \cdot)$ returns $K \leftarrow \mathsf{Dec}(sk, C)$ with the restriction that $\mathcal{A}$ is not allowed to query $\mathsf{DecO}(sk, \cdot)$ on the target ciphertext $C^*$. We define the advantage of $\mathcal{A}$ in the left experiment as

$$
\mathbf{Adv}^{\mathit{kem}\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k) \;=\; \left| \Pr\left[\, \mathbf{Exp}^{\mathit{kem}\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k) = 1 \,\right] - \frac{1}{2} \right| .
$$

A key encapsulation mechanism $\mathcal{KEM}$ is said to be *indistinguishable against chosen-ciphertext attacks* (IND-CCA) if the advantage function $\mathbf{Adv}^{\mathit{kem}\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k)$ is a negligible function in $k$ for all polynomial-time adversaries $\mathcal{A}$.

Note that in contrast to the original definition given by Cramer and Shoup [43, 17] we consider a simplified security experiment without a "find-stage". In Appendix C we show that (in case of CCA attacks) the two definitions are equivalent up to a negligible additive factor.

**Lemma 2.2** Assume a KEM is IND-CCA secure in the sense of Definition 2.1. Then it is also IND-CCA in the sense of [43, 17].

We stress that an analog of the above theorem neither holds for weaker types of attacks (such as CCA1 or CPA) nor for CCA security of PKE schemes.

KEM vs. PKE. Why to prefer a KEM over a PKE scheme? In any practical applications a random shared session key needs to be generated that is fed into a highly efficient symmetric encryption schemes. Therefore the biggest advantage of a KEM is its flexibility, i.e. it completely decouples the key encapsulation from the asymmetric part. So one is free to pick whatever security parameter necessary without changing the size of the message space. Due to his simplicity and flexibility this modular approach is incorporated in many standards (see, e.g., [44, 3, 28]). We think that the above practical advantages of a KEM are often overlooked in theoretical approaches. Ultimately, in practice one is concerned with the efficiency of the KEM and not the PKE scheme taken as a whole.

### 2.3 Target Collision Resistant Hash Functions

Let $\mathsf{TCR}_s : \mathbb{G} \to \mathbb{Z}_p$ of keyed hash functions for each $k$-bit key $s$, where $\mathbb{G}$ is a cyclic group of prime order $p$. It is assumed target collission resistant (TCR) [7, 17] captured by defining the tcr-advantage of an adversary $\mathcal{H}$ as

$$\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k) = \Pr[\mathsf{TCR}_s(c^*) = \mathsf{TCR}_s(c) \wedge c \neq c^* \ : \ s \xleftarrow{\$} \{0,1\}^k \ ; \ c^* \xleftarrow{\$} \mathbb{G} \ ; \ t \xleftarrow{\$} \mathcal{H}(s, c^*)]$$

Note $\mathsf{TCR}$ is a weaker requirement than collision-reistance, so that, in particular, any practical collision-resistant function can be used. Also note that our notion of $\mathsf{TCR}$ is related to the stronger notion of universal one-way hashing [33], where in the security experiment of the latter the target value $a^*$ is chosen by the adversary (but before seeing the hash key $s$).

Since $|\mathbb{G}| = |\mathbb{Z}_p| = p$ we can alternatively also use a fixed (non-keyed) bijective encoding function $\mathsf{INJ} : \mathbb{G} \to \mathbb{Z}_p$. We use $\mathsf{INJ}$ to refer to such a bijective encoding. In that case we have a perfect collission resistent hash function, i.e. $\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{INJ},\mathcal{H}}(k) = 0$. In particular, in the important case where $\mathbb{G}$ is an elliptic curve over a finite field $\mathbb{F}_q$ of prime order $q$ we can easily obtain such a bijective encoding. We refer to [17, 14, 20] for more examples of groups $\mathbb{G}$ that provide such efficiently computable bijective maps $\mathsf{INJ} : \mathbb{G} \to \mathbb{Z}_p$.

## 3 The Gap Hashed Diffie-Hellman Assumption

### 3.1 Standard Diffie-Hellman assumptions

We first start with the following well known standard assumptions which we review for completeness.

Computational Diffie-Hellman assumption. The *Computational Diffie-Hellman assumption* (CDH) states, that given the input $(g, g^x, g^y)$ where $x, y$ are drawn at random from $\mathbb{Z}_p$ ($g$ is a generator of a group $\mathbb{G}$ of prime order $p$), it should be computationally infeasible to compute $g^{xy}$. However, under the CDH assumption it might be as well possible to efficiently compute some information about $g^{xy}$, say a single bit of the binary representation or even all but super-logarithmically many bits.

Decisional Diffie-Hellman assumption. A stronger assumption that has been gaining popularity is the *Decisional Diffie-Hellman assumption* (DDH). It states, roughly, that the distributions $(g, g^x, g^y, g^{xy})$ and $(g, g^x, g^y, g^z)$ are computationally indistinguishable when $x, y, z$ are drawn at random from $\mathbb{Z}_p$.

Gap Diffie-Hellman assumption. Another variant of the Diffie-Hellman assumption is the *Gap Diffie-Hellman assumption* (GDH) [37]. It states that the CDH assumption is still hard even though an adversary has additional access to an oracle that solves the DDH problem.

### 3.2 The Gap Hashed Diffie-Hellman assumption

As indicated above, semantic security of Hashed Diffie-Hellman based schemes requires that we will be able to get some number of hard-core bits from the Diffie-Hellman key (i.e. bits that cannot be distinguished from random bits). We will be using a gap-assumption relative to a DDH oracle so we are not allowed to take the whole Diffie-Hellman key. Our assumption is that applying a suitable hash function $\mathsf{H}$ to $g^{xy}$ will yield such bits. The assumption we make, called the Gap Hashed Diffie-Hellman assumption (GHDH) is a "composite one"; it concerns the interaction between a hash function $\mathsf{H}$ and the group $\mathbb{G}$. The GHDH is an extension of the HDH assumption formalized by Abdalla, Bellare, Rogaway [1].

Our scheme will be parameterized by a *parameter generator*. This is a polynomial-time algorithm $\mathsf{Gen}$ that on input $1^k$ returns the description of a multiplicative cyclic group $\mathbb{G}$ of prime order $p$, where $2^k < p < 2^{k+1}$, and a random generator $g$ of $\mathbb{G}$. $\mathsf{Gen}$ furthermore outputs the description of a random hash function $\mathsf{H} : \mathbb{G} \to \{0,1\}^{l(k)}$ (possibly keyed) that outputs $l(k)$ bits for a fixed polynomial $l(\cdot)$. Throughout the paper we use $\mathcal{HG} = (\mathbb{G}, g, p, \mathsf{H})$ as shorthand for the description of the hash group obtained by running $\mathsf{Gen}$.

The GHDH assumption states, roughly, that the two distributions $(g^x, g^y, \mathsf{H}(g^{xy}))$ and $(g^x, g^y, R)$ are computationally indistinguishable when $x, y$ are drawn at random from $\mathbb{Z}_p$ and $R$ is drawn at random from $\{0,1\}^{l(k)}$. This assumption should hold relative to an oracle that efficiently solves the DDH problem.

More formally let $\mathsf{Gen}$ be a parameter generation algorithm. To an adversary $\mathcal{B}$ we associate the following experiment.

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathsf{H},\mathcal{B}}(1^k) \\
\quad \mathcal{HG} = (\mathbb{G}, g, p, \mathsf{H}) \xleftarrow{\$} \mathsf{Gen}(1^k) \\
\quad x, y \xleftarrow{\$} \mathbb{Z}_p^* \,;\; W_0 \xleftarrow{\$} \{0,1\}^{l(k)} \,;\; W_1 \leftarrow \mathsf{H}(g^{xy}) \\
\quad \gamma \xleftarrow{\$} \{0,1\} \\
\quad \gamma' \xleftarrow{\$} \mathcal{B}^{\mathsf{DDHsolve}_{\mathbb{G}}(\cdot,\cdot,\cdot,\cdot)}(1^k, \mathcal{HG}, g^x, g^y, W_\gamma) \\
\quad \text{If } \gamma \neq \gamma' \text{ then return 0 else return 1}
\end{array}
$$

Here the oracle $\mathsf{DDHsolve}_{\mathbb{G}}(g, g^a, g^b, g^c)$ returns 1 iff $ab = c \bmod p$. We define the advantage of $\mathcal{B}$ in the above experiment as

$$
\mathbf{Adv}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}(k) \;=\; \left| \Pr\left[ \mathbf{Exp}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}(1^k) = 1 \right] - \frac{1}{2} \right| .
$$

We say that the *Gap Hashed Diffie-Hellman (GHDH) assumption relative to group generator* $\mathsf{Gen}$ holds if $\mathbf{Adv}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}$ is a negligible function in $k$ for all polynomial-time adversaries $\mathcal{B}$.

We remark that in so called gap-groups [37], i.e. in groups where the Decisional Diffie-Hellman (DDH) problem is easy on every input while the computational Diffie-Hellman (CDH) problem CDH problem is hard, the GHDH assumption is equivalent to the HDH assumption. A possible implementation of gap-groups is given by the Weil/Tate bilinear pairing allowing to efficiently compute a bilinear pairing which can be used to solve DDH [13].

In Section 5 we will propose promising candidates for $\mathsf{Gen}$ (i.e., for the prime-order group $\mathbb{G}$ and the hash function $\mathsf{H}$) and provide a detailed security analysis of the GHDH assumption.

# 4 Key Encapsulation based on Gap Hashed Diffie-Hellman

In this section we present our new GHDH-based key encapsulation mechanism. The starting point is Hashed Diffie-Hellman (cf. Section 1) which itself is only secure against chosen-plaintext attacks under the HDH assumption.

## 4.1 The Key Encapsulation Mechanism

Let $\mathcal{HG} = (\mathbb{G}, g, p, \mathsf{H})$ be random parameters obtained by running the parameter algorithm $\mathsf{Gen}(1^k)$, where $\mathsf{H} : \mathbb{G} \to \{0,1\}^{l(k)}$ is a random instance of a hash function such that the GHDH assumptions holds relative to $\mathsf{Gen}$. Let $\mathsf{TCR}_k = \mathsf{TCR} : \mathbb{G} \to \mathbb{Z}_p$ be a family of target collision-resistant hash functions. We build a key encapsulation mechanism $\mathcal{KEM} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ as follows.

| $\mathsf{Kg}(1^k)$ | $\mathsf{Enc}(pk)$ | $\mathsf{Dec}(sk, C)$ |
|---|---|---|
| $x, y \xleftarrow{\$} \mathbb{Z}_p^*$ | $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; $c \leftarrow g^r$ | Parse $C$ as $(c, \pi)$ |
| $u \leftarrow g^x$ ; $v \leftarrow g^y$ | $t \leftarrow \mathsf{TCR}(c)$ ; $\pi \leftarrow (u^t v)^r$ | $t \leftarrow \mathsf{TCR}(c)$ |
| $pk \leftarrow (u, v)$ | $K \leftarrow \mathsf{H}(u^r) \in \{0,1\}^{l(k)}$ | If $c^{xt+y} \neq \pi$ then `reject` |
| $sk \leftarrow (x, y)$ | $C \leftarrow (c, \pi) \in \mathbb{G}^2$ | Else $K \leftarrow \mathsf{H}(c^x)$ |
| Return $(pk, sk)$ | Return $(C, K)$ | Return $K$ |

For completeness the public-key $pk$ should also contain a random seed for the target collision resistant hash function $\mathsf{TCR}$ and, if necessary, also for the hash function $\mathsf{H}$. Decapsulation has to perform one subgroup-membership test, i.e. it checks if $c \in \mathbb{G}$ (and rejects otherwise). Note that $c^{xt+y} = \pi$ then automatically also implies $\pi \in \mathbb{G}$.[1]

Efficiency. The public key contains two group elements, the secret key of two elements from $\mathbb{Z}_p$. A ciphertext $C$ consists of two group elements, the key $K$ is a binary string of length $l$ (that may be extended to poly($l(k)$) bits using a key-derivation function). Ignoring all "symmetric operations" (i.e., evaluating $\mathsf{TCR}$ and $\mathsf{H}$), encapsulation needs three regular exponentiations, whereas decapsulation can be carried out in two exponentiation.

Using the concept sequential/multi-exponentiations[2] (see, e.g., [40, 9]) a considerable (and practical) speed-up can be obtained: encapsulation needs two regular exponentiations (to compute $c$ and $K$) plus one multi-exponentiation (to compute $\pi = u^{tr} v^r$), whereas decapsulation can be carried out in one single sequential exponentiation (to compute $c^{xt+y}$ and $c^x$).

Correctness. Fix a pair of keys $(pk, sk)$. We call a ciphertext $C = (c, \pi) \in \mathbb{G}^2$ *consistent* if $c^{xt+y} = \pi$ for $t = \mathsf{TCR}(c)$. For a correctly generated ciphertext $C = (c, \pi) = (g^r, u^{tr} v^r)$ we have $c^{xt+y} = (g^{xt+y})^r = (u^t v)^r = \pi$ and hence $C$ is consistent. In that case decapsulation reconstructs the session key as $K = \mathsf{H}(c^x) = \mathsf{H}((g^r)^x) = \mathsf{H}(u^r)$, as the key in encapsulation. This shows correctness of the scheme.

Public Verifiability in Gap-Groups. Let $C = (c, \pi) \in \mathbb{G}^2$ be a ciphertext with $c = g^r$ for some value $r \in \mathbb{Z}_p$. Then $(g, u^t v = g^{xt+y}, c = g^r, \pi)$ is a Diffie-Hellman-tuple if and only if $g^{(xt+y)\cdot r} = \pi$ what is equivalent to $c^{xt+y} = \pi$. Therefore in gap-groups consistency of a ciphertext can be publicly checked using one call to the Diffie-Hellman oracle, i.e. by verifying if $\mathsf{DDHsolve}(g, u^t v = g^{xt+y}, c = g^r, \pi)$ returns true. This property is denoted as *public verifiability of the ciphertext* and it give rise to a public-key threshold KEM [14].

Alternative Decapsulation. We further remark that equivalently decapsulation can be done by first pick a random $s \in \mathbb{Z}_p^*$ and then reconstructing the session key as $K = \mathsf{H}(c^{x-s(xt+y)} \cdot \pi^s)$. Its equivalent computation can be demonstrated similarly to [30]. This alternative decapsulation algorithm seems usefull to protect against side-channel attacks since it does not (even internally) leak information about consistency of a queried ciphertext.

### 4.2 Security

Our main theorem can be stated as follows:

[1] Elements in $\mathbb{G}$ are usually represented as elements in $\hat{\mathbb{G}}$, where $\mathbb{G} \subseteq \hat{\mathbb{G}}$ is a prime-order subgroup of $\hat{\mathbb{G}}$. It is crucial that the two ciphertext elements are actually contained in $\mathbb{G}$ since otherwise there may be an attack on the KEM. See [17] for further discussion on the importance of such subgroup membership tests.

[2] One multi-exponentiaion computes the group element $g^a h^b$ and one sequential exponentiation computes the two group elements $g^a$ and $g^b$ in one single step (for the same fixed base $g$). Both concepts are related and (using Pippenger's algorithm [40]) can be carried out in about $(1 + 2/\log\log p)\log p$ multiplications over $\mathbb{G}$ [9] which we will count as $\approx 1.2$ exponentiations.

**Theorem 4.1** Assume $\mathsf{TCR}$ is a target collision resistant hash function. Under the Gap Hashed Diffie-Hellman assumption relative to generator $\mathsf{Gen}$, the key encapsulation mechanism from Section 4.1 is secure against chosen-ciphertext attacks. In particular, for any adversary $\mathcal{A}$ against the KEM running for time $\mathbf{Time}_{\mathcal{A}}(k)$, there exists adversaries $\mathcal{B}$, $\mathcal{H}$ with

$$\mathbf{Adv}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}(k) \geq \mathbf{Adv}^{\mathit{kem}\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k) - \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)$$

and $\mathbf{Time}_{\mathcal{B}}(k) = \mathbf{Time}_{\mathcal{H}}(k) = \mathbf{Time}_{\mathcal{A}}(k) + \mathcal{O}(q \cdot \mathbf{Time}_{\mathbb{G}}(k))$, where $q$ is an upper bound on the number of decapsulation queries made by adversary $\mathcal{A}$ and $\mathbf{Time}_{\mathbb{G}}(k)$ is the time for a standard operation in $\mathbb{G}$.

We want to stress that the key encapsulation mechanism does not make use of the Decision Diffie-Hellman oracle $\mathsf{DDHsolve}$. Its existence is part of the assumption and solely needed for the proof of security.

The proof is quite simple. An intuitive way to understand it is as follows: first consider a modified KEM that is obtained by abandoning the hash function $\mathsf{H}$ from the construction in Section 4.1, i.e. the symmetric key is now computed as $K = u^r$. What we can prove is that this modified KEM is *one-way chosen-ciphertext secure* under the gap Diffie-Hellman (GDH) assumption. In the security reduction the DDH oracle provided by the GDH assumption is used to reject (as in the original scheme) every invalid ciphertext submitted by the adversary to the decryption oracle. The key idea of the reduction is based on an algebraic technique from [10] that was also used in [14, 30] in the context of KEMs. An attacker $\mathcal{B}$ against the GDH problem can setup the public-key for the adversary attacking the security of the KEM in a way that (i) adversary $\mathcal{B}$ (without knowing the secret key) can decapsulate every ciphertexts except the challenge ciphertext; (ii) decapsulating the challenge ciphertext is equivalent to solving GDH. If the adversary against the KEM is successfull (i.e. it decapsulates the challenge ciphertext) so this adversary can be used to break the GDH problem using the above simulation.

More details. Adversary $\mathcal{B}$ inputs a GDH instance $(g, u, g^a)$ and it's goal is to compute $T = u^a$ (recall that we are attacking one-way chosen-ciphertext security). He picks a random value $d$ and defines the (thereby correctly distributed) public key as $pk = (u, v = u^{-t^*} g^d)$, where $t^* = \mathsf{TCR}(g^a)$. Note that this way a consistent ciphertext $(c, \pi)$ properly created by the encapsulation algorithm has the form

$$c = g^r, \quad \pi = (u^t v)^r = (u^r)^{t-t^*} c^d \; . \tag{1}$$

Hence, in order to decapsulate the challenge ciphertext $C^* = (c^*, \pi^*)$ defined as $c^* := g^a$, $\pi^* := (g^a)^d = (u^a)^{t^*-t^*} c^d$ (i.e., a ciphertext computed with unknown randomness $a$ from the GDH instance, where $t^* = \mathsf{TCR}(c^*)$), adversary $\mathcal{A}$ (which is run on $pk$ and $C^*$) has to compute the target key $K^* = u^a$ what is equivalent to breaking GDH. On the other hand, for a decapsulation query for ciphertext $(c, \pi)$, $\mathcal{B}$ first checks for consistency using the DDH oracle $\mathsf{DDHsolve}$ provided by the GDH assumption. If the ciphertext is inconsistent it gets rejected. Otherwise, the security properties of $\mathsf{TCR}$ imply $t = \mathsf{TCR}(c) \neq \mathsf{TCR}(c^*) = t^*$ and the correct key $K = u^r$ can be reconstructed by Eqn. (1) as $K = (\pi / c^d)^{1/(t-t^*)}$.

The step to full security (i.e., indistinguishability compared to one-wayness) now can be intuitively understood by the fact that (in terms of the assumption) we move from GDH to GHDH, i.e. under GHDH the hash function $\mathsf{H}$ hides all information about the Diffie-Hellman key $u^r$. A more formal proof is given in Appendix A.

**Remark 4.2** For the security proof of Theorem 4.1 a weaker version of the GHDH assumption is actually sufficient. The weak GHDH security experiment is the same as the one from Section 5 with the difference that the adversary only gets access to a *restricted DDH oracle* $\mathsf{DDHsolve}_{g,g^x}(\cdot,\cdot)$,

where, for fixed $g$ and $g^x$, $\mathsf{DDHsolve}_{g,g^x}(g^b, g^c)$ answers 1 if $bx = c$ and 0, otherwise. See the proof in Appendix A for more details.

### 4.3 KEM/DEM: from KEM to full encryption

Given a KEM and a symmetric encryption scheme (aka DEM), a hybrid public-key encryption scheme can be obtained by using the KEM to securely transport a random session key that is fed into the DEM to encrypt the plaintext message. It is well known that if both the KEM and the DEM are chosen-ciphertext secure, then the resulting hybrid encryption is also chosen-ciphertext secure [17, Sec. 7]. The security reduction is tight.

A DEM secure against chosen-ciphertext attacks can be built from relatively weak primitives, i.e. from any one-time symmetric encryption scheme by essentially adding a MAC. Phan and Pointcheval [39] showed that strong pseudorandomn permutations directly imply redundancy-free chosen-ciphertext secure DEMs that avoid the usual overhead due to the MAC. It seems reasonable to believe that known block-ciphers (auch as AES) are strong PRPs. In practice, the modes of operation CMC [24], EME [25], and EME* [23] (provided that the underlying block-cipher is a strong PRP) can be used to encrypt large messages. We note that for the natural task of securely generating a joint random session key, a KEM is sufficient and a fully-fledged public-key encryption scheme is not needed.

### 4.4 Direct public-key encryption

Building on a technique due to Waters [45], it is possible to build a direct public-key encryption scheme (or a tag-KEM [2]) that can be instantiated with any one-time secure symmetric encryption scheme (rather than a chosen-ciphertext secure one). A similar construction based on the BDH assumption (hence using pairings) was already given in [14]. More details of this construction are given in Appendix B. Implementing this scheme in a gap group (i.e. a group that supports an efficient implementiation of a DDH oracle) we readily get a non-interactive chosen-ciphertext secure threshold encryption scheme [14].

### 4.5 Relation to other encryption schemes

The KEMs based on "identity-based techniques" [14, 30, 31] are very similar to our construction. In fact, (a slight variation of) the KEM from [14] (which itself is based on the first IBE scheme Boneh and Boyen [10]) can be obtained from our KEM by instantiating the hash function $\mathsf{H}$ with a bilinear map, i.e. by defining $\mathsf{H}(X) = \hat{e}(g^z, X)$ (further simplifications in the decapsulation algorithm must be applied). As we will further explain in Section 5.4, security of the KEM then can be proved relative to the BDH assumption (just as in [14]). However, since it involves computing bilinear maps, the BWM-KEM is considerably less efficient than our proposal when $\mathsf{H}$ is a cryptographic hash function.

Surprisingly, the KEM part (KD-KEM) of the Kurosawa-Desmedt public-key encryption scheme [32] looks quite similar to our construction. Indeed, the KD-KEM encapsulates by computing the ciphertext as $(c_1, c_2) = (g^r, \hat{g}^r)$ and the corresponding symmetric key is defined as $K = (u^t v)^r$, where $g, \hat{g}, u = g^x \hat{g}^{\hat{x}}, v = g^y \hat{g}^{\hat{y}}$ are elements from the public key and $t$ is computed as $t = \mathsf{TCR}(c_1, c_2)$. In comparison (and ignoring the hash function) our scheme basically *swaps* the elements $c_2$ and $K$, i.e. the ciphertexts of our scheme are given by $(g^r, (u^t v)^r)$, where the corresponding key is $\mathsf{H}(u^r)$.

In contrast our scheme is provably secure under a well-defined number-theoretic assumption whereas the KD-KEM was recently shown to be not even one-way chosen-ciphertext secure [26].[3] One could

[3] In fact that the KD-KEM is not even non-malleable against chosen-plaintext attacks. We stress again that it was never claimed in [32] that the KD-KEM is chosen-ciphertext secure, nor does [26] imply that the KD-scheme, as a PKE

possible remark that the stronger security properties of our KEM inherently rely on the stronger assumption, i.e., the hash function $\mathsf{H}$ and the DDH oracle in the GHDH assumption (the gap-property). However, this is not true as we will explain now; security rather seems to depend on the particular constellation of the ciphertexts of our KEM. First, the attack from [26] aganst the KD-KEM is still valid if the two elements in the KD-KEM ciphertext get checked for consistency before decapsulating the key, i.e. the attack does not rely on "inconsistent ciphertext queries". In other words it is not the "gap"-property of the GHDH assumptions that makes the difference in the (in-)security of the two KEMs. Second, chosen-ciphertext security of our KEM does also not depend on the hash function $\mathsf{H}$ since without $\mathsf{H}$ our KEM is still *one-way* chosen-ciphertext secure under the gap computational Diffie-Hellman assumption. As pointed out earlier the hash function $\mathsf{H}$ is only responsible to provide indistinguishability (rather than one-wayness).

# 5 The Gap Hashed Diffie Hellman Assumption

In this section we discuss the security properties of the GHDH assumption introduced in Section 3.2 and propose promising candidate instantiations for the hash function $\mathsf{H}$.

## 5.1 A necessary condition on the hash function for GHDH

Clearly, "one-wayness" is a necessary condition to $\mathsf{H}$ since otherwise the Hashed Gap Diffie-Hellman assumption can be trivially broken using the DDH solver. We remark that in contrast to the usual application of one-way functions (requiring *length-preserving one-way functions* [21]) using a one-way function that hides a specific pre-image (instead of any pre-image) is sufficient in our context. A little bit more formal we say that $\mathsf{H} : \mathbb{G} \to \{0,1\}^{l(k)}$ is a *specific pre-image resistant one-way function* if $\Pr[\,\mathcal{A}(\mathsf{H}, \mathsf{H}(x)) = x\,]$, for uniform $x \in \mathbb{G}$, is a negligible function in $k$ for any polynomial-time adversary $\mathcal{A}$. With other words, $\mathsf{H}$ may get its one-wayness from a computational assumption (such as discrete log), or even from a statistical property, i.e. when $\mathsf{H}(x)$ simply does not contain sufficient information to recover $x$. (For concreteness consider a $\mathsf{H}$ that shrinks its input by at least $\omega(\log k)$ bits.) The latter case also means that $\mathsf{H}$ is many-to-one and hence inverting $\mathsf{H}$ (in a conventional sense) may be easy – but of course to break GHDH it is not sufficient to have a "random" pre-image. Instead, for an attack we would have to hit the precise value $g^{xy}$ which is information-theoretically hidden in $\mathsf{H}(g^{xy})$.

## 5.2 Generic Security of GHDH

Generic Group model. We now argue why for all "reasonable choices" of $\mathsf{H}$ the GHDH assumption holds in the generic group model [34, 42]. Let $\pi : \mathbb{G} \to \{0,1\}^{n(k)}$ be a specific pre-image resistant one-way function. In the generic group model all group elements are represented as random strings and the adversary is given access to the group operation through an oracle. Furthermore the adversary is given access to a DDH solver and an oracle that evaluates the one-way function $\pi : \mathbb{G} \to \{0,1\}^{n(k)}$. Consider the two distributions $(g^x, g^y, \pi(g^{xy}))$ and $(g^x, g^y, \pi(g^z))$ when $x, y, z$ are drawn at random from $\mathbb{Z}_p$. In the generic group model it is easy to show that the two distributions are statistically indistinguishable, i.e. that given $g$, $g^x$, $g^y$, the random variable $X_{n(k)} = \pi(g^{xy})$ has (with high probabilty) almost $n(k) \approx k$ bits of min-entropy. This has the interpretation that in the generic group model the above necessary condition on $\mathsf{H}$ actually becomes a sufficient one. The proof is similar to the proof of the BDH assumption ([11, App. A]) and is therefore omitted here.

scheme, does not meet the claimed security properties in [32].

Since $\pi(g^{xy})$ contains sufficient min-entropy we can extract a quasi uniformly distributed bit-string from it. This is done by applying a strong extractor ("entropy-smooting" function) $\sigma : \{0,1\}^{n(k)} \rightarrow \{0,1\}^{l(k)}$ and setting $\mathsf{H} = \sigma \circ \pi$ such that $\mathsf{H}(x) = \sigma(\pi(x))$. We now move on and discuss a possible choice of the function $\sigma$. Now we split $\sigma = f \circ h$ and use a variant of the Left-over Hash Lemma from [19] as follows. If $h : \{0,1\}^{n(k)} \rightarrow \{0,1\}^{N(k)}$ is pairwise independent, and $f : \{0,1\}^{N(k)} \rightarrow \{0,1\}^{l(k)}$ is an arbitrary function whose output is slightly less than the min-entropy of $X$, then the two distributions $(h, f(h(X_{n(k)}))$ and $(h, f(U_{n(k)}))$ are computational indistinguishable, where $U_{n(k)}$ denotes the uniform distribution over $\{0,1\}^{n(k)}$. To be more precise the condition we have to make on $l(k)$ is $l(k) \leq n(k) - 2\log(1/\varepsilon) - 1$, where $\varepsilon$ resembles the statistical distance between the above two distributions which should be negligible in the security parameter $k$.

Random Oracle Model. We show that in the random oracle model [6] (i.e., in an idealized world where all parties magically get black-box access to a truly random function) the GHDH assumption becomes equivalent to the GDH assumption. The novelty is that using the DDH solver we can get a tight reduction. Its simple proof is postponed to Appendix D. We note that the proof only needs a weak variant of random oracles, so called *non-programmable* or *passive* random oracles [35].

**Lemma 5.1** If $\mathsf{H}$ is modeled as a (non-programmable) random oracle then the GHDH assumption is tightly equivalent to the GDH assumption.

### 5.3 Cryptographic Hash Functions

Our suggested choice is to appropriately derive $\mathsf{H}$ from some cryptographic hash function like SHA1, MD5, or RIPEMD. We call the corresponding assumptions SHA1-HDH, MD5-HDH, and RIPEMD-HDH, respectively. The primary reason we prefer a cryptographic function is that one-wayness of $\mathsf{H}$ appears important to the Hashed Diffie-Hellman assumption: it should be hard to recover $g^{xy}$ from $\mathsf{H}(g^{xy})$, since otherwise the GHDH assumption can be trivially broken using the DDH solver. This cryptographic hash function should be chosen *independent* of the underlying group $\mathbb{G}$.

### 5.4 A Hash function based on bilinear maps

A possible point of criticism against the GHDH assumption is that it may be considered *non-standard* in some sense since it is relative to some "non-algebraic" hash function. These concerns are certainly valid. In this paragraph we show that the GHDH assumption instantiated in a *specific group* and with a *specific (algebraic) hash function* actually becomes the well-known Bilinear Diffie-Hellman (BDH) assumption which recently found numerous application and may be considered as standard. To this end we generalize the notion of the hash funtion to $\mathsf{H} : \mathbb{G} \rightarrow \mathsf{KeySp}$, where $\mathsf{KeySp}$ may be a binary string $\{0,1\}^{l(k)}$ (as before) but also an algebraic object such a cyclic group $\hat{\mathbb{G}}$.

We briefly review the necessary facts about bilinear maps and bilinear groups. Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative groups of prime order $p$ and let $g$ be a generator of $\mathbb{G}$. Assume there is a mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ that is bilinear (i.e., for all $g_1, g_2 \in \mathbb{G}, x_1, x_2 \in \mathbb{Z}$, $\hat{e}(g_1^{x_1}, g_2^{x_2}) = \hat{e}(g_1, g_2)^{x_1 x_2}$) and non-degenerate (i.e., $\hat{e}(g,g) \neq 1_{\mathbb{G}_T}$). We say that $\mathbb{G}$ is a *bilinear group* if there exist a group $\mathbb{G}_T$ and a bilinear map $\hat{e}$ satisfying the conditions above; moreover, the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ and $\hat{e}$ must be efficiently computable. Note that any bilinear group where CDH holds is a gap-group since the DDH solver $\mathsf{DDHsolve}(g, g^x, g^y, g^z)$ can be implemented by checking if $\hat{e}(g, g^z) = \hat{e}(g^x, g^y)$.

The BDH assumption [13] states that the two distributions $(g^x, g^y, g^z, \hat{e}(g,g)^{xyz})$ and $(g^x, g^y, g^z, \hat{e}(g,g)^r)$ are computationally indistinguishable when $x, y, z, r$ are drawn at random from $\mathbb{Z}_p$. To some fixed public element $Y \in \mathbb{G}$ (which is included in the description of $\mathsf{H}$) we associate the mapping $\hat{e}_Y(X) : \mathbb{G} \rightarrow \mathbb{G}_T$ with $\hat{e}_Y(X) \mapsto \hat{e}(X, Y)$. Then defining the hash function $\mathsf{H}$ as $\mathsf{H}(X) = \mathsf{H}_Y(X) = \hat{e}_Y(X)$ (such that

$\mathsf{H}_{g^z}(g^{xy}) = \hat{e}(g,g)^{xyz}$), the GHDH assumption becomes the BDH assumption. Note that the BDH assumption implicitly contains the assumption that the function $\hat{e}_Y$ is one-way, sine otherwise the BDH assumption could be broken using the DDH solver (implemented using the bilinear mapping $\hat{e}$). See [29] and [13, Sec. 3] for more discussion on that issue.

We stress that, however, we do not recommend using the bilinear map as hash function since evaluating it would dominate the running time of all our application protocols.

### 5.5 Discussion

We argued that the GHDH assumption relative to particular hash functions $\mathsf{H}$ and groups $\mathbb{G}$ seems to be a reasonable intractability assumption which may deserves further attention. Depending on specific groups (e.g., on elliptic curves) we propose to study further "algebraic" candidates for $\mathsf{H}$, and to discuss their efficiency and security in the GHDH context.

We also remark that for specific choices of the hash function $\mathsf{H}$, GHDH can be true in the scenario when DDH turns out to be wrong in all practical groups considerable for cryptography. Note that in contrast, BDH inherently relies on DDH in the target group so once DDH is wrong in general, BDH falls as well.

## 6 Key-encapsulation based on GHMDH

In this section we further generalize our KEM construction to build schemes based on the general class of GHMDH assumptions which we now introduce.

### 6.1 Gap Hashed Multi Diffie-Hellman Assumptions

For an integer $\ell \geq 1$ let $\mathbf{D} \in \mathbb{G}^{\ell\times\ell}$ be a matrix with entries $D_{i,j} \in \mathbb{G}$. Let $\mathsf{H} : \mathbb{G}^{\ell\times\ell} \to \mathcal{K}$ be a hash-function that maps $\ell^2$ group elements into the key-space $\mathcal{K}$. Informally, the Gap Hashed Multi Diffie-Hellman (GHMDH) assumption (realtive to hash function $\mathsf{H}$ and group $\mathbb{G}$) states that, given $g_1,\ldots,g_\ell,h_1,\ldots,h_\ell,g_1^{r_1},\ldots,g_\ell^{r_\ell}$ and access to a DDH oracle, it is computationally infeasible to distinguish $\mathsf{H}(\mathbf{D})$ from a random element in $\mathcal{K}$, where the (hidden) entries of matrix $\mathbf{D}$ contain all possible combinations of Diffie-Hellman keys, i.e. $D_{i,j} = h_i^{r_j}$. Intuitively, the hash function $\mathsf{H}$ can be viewed as a hard predicate of the $\ell^2$ different Diffie-Hellman keys. Clearly, for $\ell = 1$ and $\mathcal{K} = \{0,1\}^{l(k)}$ this simplifies to the GHDH assumption but in this section we focus mostly on algebraic candidates of the form $\mathsf{H} : \mathbb{G}^{\ell\times\ell} \to \mathbb{G}$.

We let a parameter generator $\mathsf{GenM}(1^k)$ output a random group $\mathbb{G}$ or primeorder $p$, a generator $g$ of $|G$, and a random instance of the hash function $\mathsf{H} : \mathbb{G}^{\ell\times\ell} \to \mathcal{K}$. More formally, to an adversary $\mathcal{B}$ we associate the following experiment.

**Experiment** $\mathbf{Exp}^{\text{ghmdh}}_{\mathsf{Gen},\mathsf{H},\mathcal{B}}(1^k)$
$(\mathbb{G},g,p,\mathsf{H}) \xleftarrow{\$} \mathsf{GenM}(1^k)$
For $i \in \{1,\ldots,\ell\}$ do $g_i,h_i \xleftarrow{\$} \mathbb{G}$ ; $r_i \xleftarrow{\$} \mathbb{Z}_p^*$
for $1 \leq i,j \leq \ell$ do $D_{i,j} \leftarrow h_i^{r_j}$
$W_0 \xleftarrow{\$} \mathcal{K}$ ; $W_1 \leftarrow \mathsf{H}(\mathbf{D})$ ; $c \xleftarrow{\$} \{0,1\}$
$c' \xleftarrow{\$} \mathcal{B}^{\mathsf{DDHsolve}(\cdot,\cdot,\cdot,\cdot)}(1^k,\mathcal{HG},g_1,\ldots,g_\ell,g_1^{r_1},\ldots,g_\ell^{r_\ell},W_c)$
If $c \neq c'$ then return 0 else return 1

We define the advantage of $\mathcal{B}$ in the above experiment as

$$\mathbf{Adv}^{\text{ghmdh}}_{\mathsf{GenM},\mathcal{B}}(k) \;=\; \left|\Pr\left[\,\mathbf{Exp}^{\text{ghmdh}}_{\mathsf{GenM},\mathcal{B}}(1^k) = 1\,\right] - \frac{1}{2}\right| .$$

We say that the *Hashed Gap Multi Diffie-Hellman (GHMDH) assumption relative to group generator* $\mathsf{GenM}$ holds if $\mathbf{Adv}^{\text{ghmdh}}_{\mathsf{GenM},\mathcal{B}}$ is a negligible function in $k$ for all polynomial-time adversaries $\mathcal{B}$.

### 6.2 The class of $\ell$-GLDH assumptions

Using the general class of GHMDH assumptions the Gap Decision Linear Diffie-Hellman (GLDH) assumption [12] is obtained by setting $\ell = 2$ and defining $\mathsf{H} : \mathbb{G}^{2\times 2} \to \mathbb{G}$ as $\mathsf{H}(\mathbf{D}) = D_{1,1} \cdot D_{1,2}$. More precisely, the GLDH assumption states that, given $g_1, g_2, g_1^{r_1}, g_2^{r_2}, h_1, W$, destinguishing $W = h_1^{r_1+r_2}$ from a uniform $W \in \mathbb{G}$ is computational infeasible, even relative to a DDH oracle. Originally the GLDH assumption was defined over bilinear maps [12] (called Decision Linear Diffie-Hellman assumption), whereas here we only require the assumption to hold relative to a DDH oracle. This, in particilar, makes it possible to define (and apply) it relative to any cyclic group [30]. We remark that the GLDH assumption holds in the generic group model, even relative to a pairing oracle [12].

More generally, for any polynomial $\ell = \ell(k) \geq 2$, one can also define the class of $\ell$-GLDH assumptions for arbitrary $\ell = \ell(k) = \text{poly}(k)$ by defining $\mathsf{H} : \mathbb{G}^{\ell\times\ell} \to \mathbb{G}$ as $\mathsf{H}(\mathbf{D}) = \prod_{i=1}^{\ell} D_{1,i}$. (Note that the 1-GLDH assumption states that DDH is hard relative to a DDH oracle which is clearly insecure without applying any further hash function to the Diffie-Hellman key.) The $\ell$-GLDH assumptions form a strict hierarchy of security assumptions with 2-GLDH = GLDH and, the larger the $\ell$, the weaker the $\ell$-GLDH assumption. More precisely, for any $\ell \geq 2$ we have that $\ell$-GLDH implies $\ell+1$-GLDH. On the other hand (extending [12]) we can show that in the generic group model [42], the $\ell+1$-GLDH assumption holds, even relative to an $\ell$-GLDH oracle.

### 6.3 Key Encapsulation based on GHMDH

Let $\mathcal{HG} = (\mathbb{G}, g, p, \mathsf{H})$ be random parameters obtained by running the parameter algorithm $\mathsf{GenM}(1^k)$, where $\mathsf{H} : \mathbb{G}^{\ell\times\ell} \to \mathcal{K}$ is a random instance of a hash function such that the GMHDH assumptions holds relative to $\mathsf{GenM}$. We define $S_{\mathsf{H}}$ as the subset of indices $(i,j) \in \{1,\ldots,\ell\}^2$ such that the hash function $\mathsf{H}(\mathbf{D})$ depends on entry $D_{i,j}$. (For example, for $\ell$-GLDH we have $S_{\mathsf{H}} = \{(1,1),\ldots,(1,\ell)\}$.) Let $\mathsf{TCR}_k = \mathsf{TCR} : \mathbb{G}^{\ell} \to \mathbb{Z}_p$ be a family of target collision-resistant hash functions.

We build a key encapsulation mechanism $\mathcal{KEM} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ as follows. Key generation generates random group elements $g_1,\ldots,g_\ell, h_1,\ldots,h_\ell$ and indices $x_{i,j}$ ($(i,j) \in S_{\mathsf{H}}$) such that $h_i = g_j^{x_{i,j}}$. Furthermore it defines $u_{i,j} = g_j^{y_{i,j}}$, for random $y_{i,j}$ ($(i,j) \in S_{\mathsf{H}}$). The public key contains the elements $(g_i)_{1\leq i\leq \ell}, (h_i)_{1\leq i\leq \ell}$, and $(u_{i,j})_{(i,j)\in S_{\mathsf{H}}}$, and the secret key contains all corresponding indices.

| $\mathsf{Enc}(pk)$ | $\mathsf{Dec}(sk, C)$ |
|---|---|
| For each $j \in \{1,\ldots,\ell\}$: $r_j \xleftarrow{\$} \mathbb{Z}_p^*$ ; $c_j \leftarrow g_j^{r_j}$ | $t \leftarrow \mathsf{TCR}(c_1,\ldots,c_\ell)$ |
| $t \leftarrow \mathsf{TCR}(c_1,\ldots,c_\ell)$ | For each $(i,j) \in S_{\mathsf{H}}$: |
| For each $(i,j) \in S_{\mathsf{H}}$: $\pi_{i,j} \leftarrow (h_i^t u_{i,j})^{r_j}$ ; $K_{i,j} \leftarrow h_i^{r_j}$ | if $c_j^{x_{i,j}t+y_{i,j}} \neq \pi_{i,j}$ `reject` |
| $K \leftarrow \mathsf{H}(\mathbf{K})$ ; $C \leftarrow (c_1,\ldots,c_\ell,(\pi_{i,j})_{(i,j)\in S_{\mathsf{H}}})$ | $K_{i,j} \leftarrow c_j^{x_{i,j}}$ |
| Return $(C, K)$ | Return $K \leftarrow \mathsf{H}(\mathbf{K})$ |

Correctness of the scheme is easy to verify. The ciphertexts of the KEM contain $\ell + |S_{\mathsf{H}}|$ group elements, public/secret keys $2\ell+|S_{\mathsf{H}}|$ elements. The scheme instantiated with the 2-GLDH assumption reproduces the KEM from [30] which, for any $\ell \geq 1$, generalizes to the class of $\ell$-GLDH schemes.

**Theorem 6.1** Assume $\mathsf{TCR}$ is a target collision resistant hash function. Under the Gap Hashed Multi Diffie-Hellman assumption relative to generator $\mathsf{GenM}$, the key encapsulation mechanism is secure against chosen-ciphertext attacks.

The proof is similar to the one of Theorem 4.1 and omitted here. We give the intuition for the OW-CCA security of the scheme based on the computational GHMDH assumption. The simulator $\mathcal{B}$ is given the values $(g_1, \ldots, g_\ell, h_1, \ldots, h_\ell, g_1^{a_1}, \ldots, g_\ell^{a_\ell})$ and wants to compute $\mathsf{H}(\mathbf{D})$, where $D_{i,j} = h_i^{a_j}$. Adversary $\mathcal{B}$ picks up random $d_{i,j}$ and sets up the elements $u_{i,j}$ of the public key as $u_{i,j} = h_i^{-t^*} g_j^{d_{i,j}}$, where $t^* = \mathsf{TCR}(g_1^{a_1}, \ldots, g_\ell^{a_\ell})$. This has the consequence that for consistent ciphertext pairs $(c_j, \pi_{i,j})$, the elements $\pi_{i,j} = (h_i^t u_{i,j})^{r_j} = ((h_i^{t-t^*} g_j^{d_{i,j}})^{r_j} = K_{i,j}^{t-t^*} c_j^{d_{i,j}}$ can be used to reconstruct all Diffie-Hellman keys $K_{i,j} = h_i^{r_j}$ as $K_{i,j} = (\pi_{i,j}/c_j^d)^{1/(t-t^*)}$, for all $\mathsf{TCR}(c_1, \ldots, c_\ell) = t \neq t^*$. The above assumes that the ciphertexts are consistent, i.e. that $c_i^{x_{i,j}t+y_{i,j}} = \pi_{i,j}$ for all $(i,j) \in S_\mathsf{H}$ which can be checked using the DDH oracle. Hence $\mathcal{B}$ can answer all decapsulation queries made by $\mathcal{A}$ (attacking the OW-CCA security of the KEM), as long as $t \neq t^*$.

For the target ciphertext $\mathcal{B}$ sets $(c_j^*, \pi_{i,j}^*) = (g_j^{a_j}, (g_j^{a_j})^{d_{i,j}})$, which is a correct ciphertext since $t^* = \mathsf{TCR}(g_1^{a_1}, \ldots, g_\ell^{a_\ell}) = \mathsf{TCR}(c_1^*, \ldots, c_\ell^*)$. To win it security game, $\mathcal{B}$ has to compute the target key $K^* = \mathsf{H}(\mathbf{K}^*) = \mathsf{H}(\mathbf{D})$ since $K_{i,j}^* = h_i^{a_j} = D_{i,j}$.

# 7 Implementation and Comparison

## 7.1 Implementation details

Implementations in prime order groups. An implementation is given by using any prime order group, for instance a prime-order subgroup of $\mathbb{Z}_q$, where $q$ is a prime of $\approx 3072$ bits for a security parameter of $k = 128$ bits. For a more efficient implementation we recommend using elliptic curve groups that allow for a considerable speed-up and small representations of one elements in $\mathbb{G}$. More precisely, NIST [36] recommends curves such that for 128 bits of security" elements in $\mathbb{G}$ need 256 bits, For concreteness we mention that for 128 bits security our schemes allows for an efficient implementation with 512 bits ciphertext size.

Implementations using bilinear groups. If one wants provable security under HDH or if the public verifiability property is needed (e.g., to make the scheme threshold [14]) one has to implement our KEM in gap-groups. Currently the only candidate instantiations of gap-groups arise from bilinear groups where the DDH solver is implemented using a bilinear pairing. It is possible that other constructions of gap-groups exist. The definition of bilinear groups from Section 5.4 assumed (for simplicity) a symmetric pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. In practice one would also like to use asymmetric pairings of the form $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_2 \neq \mathbb{G}_1$ is a group of prime order $p$. Such asymmetric bilinear groups have the advantage of being less special than symmetric ones — and consequently offer better security properties since their generality makes it harder to design tailor-made attacks. Furthermore, as we will sketch below, they lead to considerably shorter ciphertexts than symmetric pairings.

In asymmetric groups elements in $\mathbb{G}_2$ can take much more space to represent than in $\mathbb{G}_1$. Furthermore exponentiations in $\mathbb{G}_2$ are much more expensive than in $\mathbb{G}_1$ [38, 22]. We remark that our scheme can be adapted to the asymmetric case such that all ciphertext elements lie in the group $\mathbb{G}_1$ and that moreover all exponentiations can be carried out in $\mathbb{G}_1$. For a concrete implementation we mention the non-supersingular curves with security parameters between 80 and 256 bits considered in [38]. In that particular case the representation of one element in $\mathbb{G}_1$ has about the same size as in standard (non pairing-based) elliptic-curve schemes with key-sizes recommended by NIST [36].

| Scheme | Security Assmptn | Ciphertext Overhead | | Encryption #pairings + | Decryption #[multi,regular]-exp | Keygen | Keysize (pk/sk) | Publ Vfy? | Any group? |
|---|---|---|---|---|---|---|---|---|---|
| KD | DDH | $2\|p\|$ | 640 | $0+[1,2]$ | $0+[1,0]$ | $0+[2,0]$ | 4/4 | — | $\surd$ |
| CS | DDH | $3\|p\|$ | 768 | $0+[1,3]$ | $0+[1,1]$ | $0+[2,0]$ | 5/5 | — | $\surd$ |
| BMW | BDH | $2\|p\|$ | 512 | $0+[1,2]$ | $1+[0,1]$ | $1+[0,2]$ | 4/3 | $\surd$ | — |
| **Ours §4** | GHDH | $2\|p\|$ | 512 | $0+[1,2]$ | $0+[1,0]$ | $0+[0,2]$ | 3/2 | $\surd^*$ | $\surd$ |
| **Ours §6** | $\ell$-GLDH | $2\ell\|p\|$ | $512\ell$ | $0+[\ell,2\ell]$ | $0+[\ell,0]$ | $0+[0,2\ell]$ | $2\ell+1/2\ell$ | $\surd^*$ | $\surd$ |

*in gap and pairing groups only

Table 1: Efficiency comparison for chosen-ciphertext secure hybrid encryption schemes. Some figures are borrowed from [14, 30]. For efficiency we count the number of pairings + [multi exponentiations, regular exponentiations] used for encryption, decryption, and key generation. All "symmetric" operations (such as hash function/MAC/KDF) are ignored. Ciphertext overhead represents the difference (in bits) between ciphertext and plaintext length, and $|p|$ is the length of the bit-representation of a group element in $\mathbb{G}$. For concreteness the expected ciphertext overhead for an 128-bit implementation is also given. The keysize is measured in two parameters: the size of the system parameters (which are fixed for every public-key) plus the size of the public key $pk$, and the size of the secret key $sk$. Here we only take into account the number of group elements for params plus $pk$, and the number of elements in $\mathbb{Z}_p^*$ for $sk$. A "$\surd$" in the "Publ. Vfy" column means that the scheme supports public verifiability. A "$\surd$" in the "Any group?" column means that the scheme can be implemented in any prime-order group, whereas a "—" means that the scheme has to be implemented in pairing groups. For comparison we mention that relative timings for the various operations are as follows: bilinear pairing $\approx 3-5$ [38], multi(=sequential)-exponentiation $\approx 1.2$ [9], and regular exponentiation $= 1$.

## 7.2 Efficiency considerations

The usual efficiency comparison with all previously known chosen-ciphertext secure KEMs/encryption schemes in the standard model is assembled in Table 1. Here KD is the hybrid encryption scheme from Kurosawa and Desmedt [32] and CS refers to the Cramer-Shoup encryption scheme [16] which we compare in its hybrid variant from [43, 17]. BMW is the KEM from Boyen, Mei, and Waters [14]. Our first scheme is the GHDH-based KEM from Section 4 instantiated with an efficient cryptographic hash function $\mathsf{H} : \mathbb{G} \to \{0,1\}^{l(k)}$. Our second scheme refers to the $\ell$-GLDH-based scheme from Section 6 which, for the case $\ell = 2$, simplifies to the GLDH-based KEM from [30]. All KEMs are assumed to be instatiated using a redundancy-free chosen-ciphertext secure symmetric scheme to obtain a full hybrid PKE scheme. The KD encryption scheme can only be proved secure in combination with an authenticated symmetric encryption scheme which inherently adds "one MAC" overhead to the ciphertext size.

Even though our scheme shares the same number of exponentiations for encryption/decryption with the KD scheme, it has some practical advantages which makes a more efficient implementation possible. First, it is possible to use a bijective encoding $\mathsf{TCR}_0 : \mathbb{G} \to \mathbb{Z}_p$ and does not have to rely on expensive number-theoretic constructions of provably secure TCR functions. Second, one only needs one subgroup membership test for decryption, whereas the KD-scheme needs two. Depending on the underlying group such subgroup membership tests may be as expensive as one exponentiation. Third, the class of symmetric encryption schemes our KEM can be securely instantiated with is larger since we do not require authenticated encryption. This in particular makes it possible to rely on free redundancy-free "one-pass" symmetric techniques (which process the message to be encrypted only once). For authenticated encryption there are only less efficient two-pass schemes freely available since all one-pass techniques are covered by patents [8].

Acknowledgments. We thank Michel Abdalla, Mihir Bellare, Yevgeniy Dodis, Martijn Stam, and Moti Yung for comments and suggestions. This research was partially supported by the research program Sentinels (`http://www.sentinels.nl`). Sentinels is being financed by Technology Founda-

tion STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. Parts of this paper were written while the author was a visitor at University of California San Diego supported by a DAAD postdoc fellowship.

## References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer-Verlag, Berlin, Germany, Apr. 2001. 1, 2, 5

[2] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 128–146. Springer-Verlag, Berlin, Germany, May 2005. 9, 21

[3] American National Standards Institute (ANSI) X9.F1 subcommittee. ANSI X9.63 Public key cryptography for the Financial Services Industry: Elliptic curve key agreement and key transport schemes, July 5, 1998. Working draft version 2.0. 4

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, Berlin, Germany, Aug. 1998. 1

[5] M. Bellare, T. Kohno, and V. Shoup. Stateful public-key cryptosystems: How to encrypt with one 160-bit exponentiation. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 380–389. ACM Press, Oct. / Nov. 2006. 3

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 11

[7] M. Bellare and P. Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In B. S. K. Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 470–484. Springer-Verlag, Berlin, Germany, Aug. 1997. 5

[8] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation. In B. K. Roy and W. Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 389–407. Springer-Verlag, Berlin, Germany, Feb. 2004. 15

[9] D. J. Bernstein. Pippenger's exponentiation algorithm. Available from `http://cr.yp.to/papers.html`, 2001. 7, 15

[10] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, Berlin, Germany, May 2004. 3, 8, 9

[11] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer-Verlag, Berlin, Germany, May 2005. 10

[12] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, Berlin, Germany, Aug. 2004. 2, 13

[13] D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. 2, 6, 11, 12

[14] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM CCS 05*, pages 320–329. ACM Press, Nov. 2005. 2, 3, 5, 7, 8, 9, 14, 15

[15] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer-Verlag, Berlin, Germany, May 2004. 3

[16] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, Berlin, Germany, Aug. 1998. 2, 15

[17] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. 1, 3, 4, 5, 7, 9, 15

[18] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 1

[19] Y. Dodis and A. Smith. Entropic security and the encryption of high entropy messages. In J. kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 556–577. Springer-Verlag, Berlin, Germany, Feb. 2005. 11

[20] R. R. Farashahi, B. Schoenmakers, and A. Sidorenko. Efficient pseudorandom generators based on the DDH assumption. In *PKC 2007*, volume ???? of *LNCS*, pages ???–??? Springer-Verlag, 2007. 5

[21] O. Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. 10

[22] R. Granger, D. Page, and N. P. Smart. High security pairing-based cryptography revisited. In *Proceedings ANTS-7*, volume 4096, pages 480–494. Springer-Verlag, Berlin, Germany, 2006. 14

[23] S. Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In A. Canteaut and K. Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 315–327. Springer-Verlag, Berlin, Germany, Dec. 2004. 9

[24] S. Halevi and P. Rogaway. A tweakable enciphering mode. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 482–499. Springer-Verlag, Berlin, Germany, Aug. 2003. 3, 9

[25] S. Halevi and P. Rogaway. A parallelizable enciphering mode. In T. Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 292–304. Springer-Verlag, Berlin, Germany, Feb. 2004. 3, 9

[26] D. Hofheinz, J. Herranz, and E. Kiltz. The Kurosawa-Desmedt key encapsulation is not chosen-ciphertext secure. Cryptology ePrint Archive, Report 2006/207, 2006. `http://eprint.iacr.org/`. 3, 9, 10

[27] D. Hofheinz and E. Kiltz. Concise Hybrid Encryption. Manuscript, 2006. 3

[28] IEEE P1363a Committee. IEEE P1363a / D9 — standard specifications for public key cryptography: Additional techniques. `http://grouper.ieee.org/groups/1363/index.html/`, June 2001. Draft Version 9. 4

[29] A. Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, Sept. 2004. 12

[30] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer-Verlag, Berlin, Germany, Mar. 2006. 3, 7, 8, 9, 13, 15

[31] E. Kiltz. On the limitations of the spread of an IBE-to-PKE transformation. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 274–289. Springer-Verlag, Berlin, Germany, Apr. 2006. 9

[32] K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442. Springer-Verlag, Berlin, Germany, Aug. 2004. 3, 9, 10, 15

[33] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989. 5

[34] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994. 10

[35] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126. Springer-Verlag, Berlin, Germany, Aug. 2002. 11, 22

[36] NIST. Recommendation for Key Management Part 1 general — special publication 800-57. `http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf`, Aug. 2005. 14

[37] T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In K. Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, Berlin, Germany, Feb. 2001. 5, 6

[38] D. Page, N. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. *Applicable Algebra in Engineering, Communication and Computing*, 17:379–392, 2006. 14, 15

[39] D. H. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In H. Handschuh and A. Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 182–197. Springer-Verlag, Berlin, Germany, Aug. 2004. 3, 9

[40] N. Pippenger. On the evaluation of powers and related problems. In *Proceedings of FOCS 1976*, pages 258–263, 1976. 7

[41] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, Berlin, Germany, Aug. 1992. 1

[42] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer-Verlag, Berlin, Germany, May 1997. 10, 13

[43] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 275–288. Springer-Verlag, Berlin, Germany, May 2000. 1, 3, 4, 15, 21

[44] V. Shoup. ISO 18033-2: An emerging standard for public-key encryption. `http://shoup.net/iso/std6.pdf`, Dec. 2004. Final Committee Draft. 4

[45] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer-Verlag, Berlin, Germany, May 2005. 9, 21

# A Proof of Theorem 4.1

Suppose there exists a polynomial time adversary $\mathcal{A}$ that breaks the chosen-ciphertext security of the encapsulation scheme with (non-negligible) advantage $\mathbf{Adv}^{kem\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k)$ and makes at most $q$ decapsulation queries.

We show that there exists an adversary $\mathcal{B}$ that runs in time $\mathbf{Time}_{\mathcal{B}}(k) = \mathbf{Time}_{\mathcal{A}}(k) + O(q \cdot \mathbf{Time}_{\mathbb{G}}(k))$, (where $\mathbf{Time}_{\mathbb{G}}(k)$ is the time to perform a basic operation in $\mathbb{G}$) and runs adversary $\mathcal{A}$ as a subroutine to solve a random instance of the GHDH problem with advantage

$$\mathbf{Adv}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{B}}(k) \geq \mathbf{Adv}^{kem\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{A}}(k) - \mathbf{Adv}^{\mathsf{hash}\text{-}\mathrm{tcr}}_{\mathsf{TCR},\mathcal{H}}(k) \; . \tag{2}$$

Now Eqn. (2) proves the Theorem.

We now give the description of adversary $\mathcal{B}$. Adversary $\mathcal{B}$ inputs an instance of the GHDH problem, i.e. $\mathcal{B}$ inputs the values $(1^k, \mathcal{HG}, \mathsf{H}, g, u = g^a, g^b, W)$. $\mathcal{B}$'s goal is to determine whether $W = \mathsf{H}(u^b)$ or $W \in \{0,1\}^l$ is a random bit string. Adversary $\mathcal{B}$ runs adversary $\mathcal{A}$ simulating its view as in the original KEM security experiment as follows:

**Key Generation & Challenge.** Initially adversary $\mathcal{B}$ picks a random value $d \in \mathbb{Z}_p^*$ and defines the target ciphertext

$$C^* = (c^*, \pi^*) \leftarrow (g^b, (g^b)^d) \; . \tag{3}$$

and the challenge key as $K^* = W$. We denote $t^* = \mathsf{TCR}(c^*)$ as the target tag (associated with the target ciphertext). The value $v$ from the public key $pk = (u, v)$ is defined as

$$v \leftarrow u^{-t^*} \cdot g^d \; . \tag{4}$$

Note that the public key is identically distributed as in the original KEM.

With each ciphertext $C = (c, \pi)$ we associate a tag $t = \mathsf{TCR}(c)$. Recall that we call a ciphertext consistent if $\pi = (u^t v)^r$, where $r = \log_g c$. Note that the way the keys are setup for a consistent ciphertext we have

$$\pi = (u^t v)^r = (u^t u^{-t^*} g^d)^r = (u^r)^{t-t^*} \cdot c^d \; . \tag{5}$$

Given a consistent ciphertext $C = (c, \pi)$ with associated tag $t \neq t^*$ the session key $K = \mathsf{H}(c^x)$ can alternatively be computed by Eqn. (5) as

$$K = \mathsf{H}((\pi / c^d)^{(t-t^*)^{-1}}) \; . \tag{6}$$

By Eqn. (5) and since $t^* = \mathsf{TCR}(c^*)$ the challenge ciphertext $C^* = (c^*, \pi^*) = (g^b, (g^b)^d) = (c^*, (c^*)^d)$ is a correctly generated ciphertext for randomness $b$. If $W = \mathsf{H}(u^b)$ then it follows by

Eqn. (4) that $C^* = (g^b, (g^b)^d)$ is a correct ciphertext of key $K^* = W = \mathsf{H}(u^b)$, distributed as in the original experiment. On the other hand, when $W$ is uniform and independent in $\{0,1\}^l$ then $C^*$ is independent of $K^* = W$ in the adversary's view.

Adversary $\mathcal{B}$ runs $\mathcal{A}$ on input $(pk, K^*, C^*)$ answering to its queries as follows:

**Decryption queries.** The KEM decapsulation queries are simulated by $\mathcal{B}$ as follows: Let $C = (c, \pi)$ be an arbitrary ciphertext submitted to the decapsulation oracle $\mathsf{DecO}(\cdot)$. First $\mathcal{B}$ performs a consistency check of the ciphertext, i.e. it checks (using the Diffie-Hellman oracle $\mathsf{DDHsolve}(\cdot,\cdot,\cdot,\cdot)$) if $(g, u^t v, c, \pi)$ is a valid Diffie-Hellman tuple.[4]

We remark that this is the only case where the simulation depends on the existence of the DDH oracle $\mathsf{DDHsolve}$. If $C$ is not consistent then $\mathcal{B}$ returns `reject`. Otherwise, if the ciphertext is consistent $\mathcal{B}$ computes $t = \mathsf{TCR}(c)$ and distinguishes the following three cases:

**Case 1:** $t = t^*$ and $c = c^*$: adversary $\mathcal{B}$ rejects the query. In this case consistency (c.f. Eqn. (5)) implies $\pi = c^d = (c^*)^d = \pi^*$ and hence $C = C^*$ and the query made by $\mathcal{A}$ is illegal. Therefore it may be rejected by $\mathcal{B}$.

**Case 2:** $t = t^*$ and $c \neq c^*$: adversary $\mathcal{B}$ found a collision $c \neq c^*$ in $\mathsf{TCR}$ with $\mathsf{TCR}(c) = \mathsf{TCR}(c^*)$. In that case $\mathcal{B}$ returns the collision and aborts.

**Case 3:** $t \neq t^*$: adversary $\mathcal{B}$ computes the correct session key by Eqn. (6) as $K \leftarrow \mathsf{H}((\pi/c^d)^{(t-t^*)^{-1}})$.

This completes the description of the decapsulation oracle.

We have shown that unless $\mathcal{B}$ finds a collision in $\mathsf{TCR}$ (Case 2) the simulation of the decapsulation oracle is always perfect, i.e. the output of the simulated oracle $\mathsf{DecO}(sk, \cdot)$ is identically distributed as the output of $\mathsf{Dec}(sk, \cdot)$.

**Guess.** Eventually, $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$ where $\delta' = 1$ means that $K^*$ is the correct key. Algorithm $\mathcal{B}$ concludes its own game by outputting $\gamma' = \delta'$ where $\gamma' = 1$ means that $W = \mathsf{H}(g^{ab})$ and $\gamma' = 0$ means that $W$ is random.

This completes the description of adversary $\mathcal{B}$.

Analysis. We have shown that as long as there is no hash collision in $\mathsf{TCR}$ found by $\mathcal{B}$, adversary $\mathcal{A}$'s view in the simulation is identically distributed to its view in the real attack game.

Note that $c^*$ is a random element from $\mathbb{G}_1$ (provided from outside of $\mathcal{B}$'s view), therefore finding a value $c \neq c^*$ with $\mathsf{TCR}(c) = \mathsf{TCR}(c^*)$ really contradicts to the security property of the *target* collision resistant hash function. The probability that $\mathcal{B}$ finds a collision in the hash function $\mathsf{TCR}$ is bounded by $\mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)$, where $\mathcal{H}$ is an adversary against the target collision resistance of $\mathsf{TCR}$, running in about the same time as $\mathcal{B}$.

Define "F" to be the event that $\mathcal{B}$ wins its GHDH game, i.e. it outputs $\delta' = 1$ if $W = \mathsf{H}(g^{ab})$ and $\delta' = 0$ if $W$ is random. Assume there was no hash collision found by $\mathcal{B}$. On the one hand, if $W$ is uniform and independent in $\{0,1\}^l$ then the challenge ciphertext $C^*$ is independent of $K^* = W$ in the adversary's view. In that case we have $\Pr[\,\mathrm{F}\,] = \Pr[\,\delta' = 0\,] = \frac{1}{2}$. On the other hand, when $W = \mathsf{H}(g^{ab})$ then $C^*$ is a correct ciphertext of the challenge key $K^*$, distributed as in the original experiment. Then, by our assumption, $\mathcal{A}$ must make a correct guess $\delta' = 1$ with advantage at least $\mathbf{Adv}^{kem\text{-cca}}_{\mathcal{KEM},\mathcal{A}}(k)$ and we have $|\Pr[\,\mathrm{F}\,] - \frac{1}{2}| = |\Pr[\,\delta' = 1\,] - \frac{1}{2}| \geq \mathbf{Adv}^{kem\text{-cca}}_{\mathcal{KEM},\mathcal{A}}(k)$.

[4] At this point the existence of a weak DDH oracle $\mathsf{DDHsolve}_{g,u}(\cdot,\cdot)$ for fixed $u$ is sufficient. This is since $(g, u^t v, c, \pi)$ is a valid Diffie-Hellman tuple iff $(g, u, c, (\pi/c^d)^{(t-t^*)^{-1}})$ is a valid Diffie-Hellman tuple. So to verify consistency of the KEM ciphertext, $\mathcal{B}$ equivalently queries $\mathsf{DDHsolve}_{g,u}(c, (\pi/c^d)^{(t-t^*)^{-1}})$. Also cf. Remark 4.2.

Therefore, adversary $\mathcal{B}$'s advantage in the GHDH game is $\mathbf{Adv}^{\text{ghdh}}_{\mathsf{Gen},\mathcal{B}}(k) \geq \mathbf{Adv}^{kem\text{-cca}}_{\mathcal{KEM},\mathcal{A}}(k) - \mathbf{Adv}^{\text{hash-tcr}}_{\mathsf{TCR},\mathcal{H}}(k)$ which proves Eqn. (2) and completes the proof of the theorem.

# B Public-key encryption based on Gap Hashed Diffie-Hellman

We give more details of the direct PKE scheme based on GHDH that applies a hash-technique due to Waters [45] to our KEM scheme from Section 4. Let $\mathcal{HG} = (\mathbb{G}, g, p, \mathsf{H})$ be random group parameters obtained by running the group parameter algorithm $\mathsf{Gen}(1^k)$. Let $\mathsf{CR}_k = \mathsf{CR} : \mathbb{G} \times \{0,1\}^{l(k)} \to \{0,1\}^{n(k)}$ be a family of collision-resistant hash functions. We build a public-key encryption scheme $\mathcal{PKE}$ as follows.

| $\mathsf{PKEkg}(1^k)$ | $\mathsf{PKEenc}(pk, M)$ | $\mathsf{PKEdec}(sk, C)$ |
|---|---|---|
| $x_0, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_p^*$ | $r \xleftarrow{\$} \mathbb{Z}_p^*$ ; $c_1 \leftarrow g^r$ | Parse $C$ as $(c_1, c_2, e)$ |
| $h_0 \leftarrow g^{x_0}, \ldots, h_n \leftarrow g^{x_n}$ | $e \leftarrow M \oplus \mathsf{H}(h_0^r)$ ; $t \leftarrow \mathsf{CR}(c_1 \| e)$ | $t \leftarrow \mathsf{CR}(c_1 \| e)$ |
| $pk \leftarrow (h_0, \ldots, h_n)$ | $c_2 \leftarrow (h_0 \prod_{i=1}^n h_i^{t_i})^r$ | If $c_1^{x_0 + \sum_{i=1}^n x_i t_i} \neq c_2$ |
| $sk \leftarrow (x_0, \ldots, x_n)$ | $C \leftarrow (c_1, c_2, e)$ | then `reject` |
| Return $(pk, sk)$ | Return $C \in \mathbb{G}^2 \times \{0,1\}^{l(k)}$ | Else return $M \leftarrow e \oplus \mathsf{H}(c_1^{x_0})$ |

**Theorem B.1** Assume $\mathsf{CR}$ is a collision resistant hash function which outputs strings of length $n(k)$. Under the Gap Hashed Diffie-Hellman assumption relative to generator $\mathsf{Gen}$ and hash function $\mathsf{H}$, the above public-key encryption scheme is secure against chosen-ciphertext attacks. In particular, for any adversary $\mathcal{A}$ against the PKE scheme running for time $\mathbf{Time}_{\mathcal{A}}(k)$, there exists an adversary $\mathcal{B}$ against GHDH with $\mathbf{Adv}^{\text{ghdh}}_{\mathsf{Gen},\mathcal{B}}(k) := \varepsilon$, where

$$\varepsilon = \Omega(\mathbf{Adv}^{pke\text{-cca}}_{\mathcal{PKE},\mathcal{A}}(k)/(nq) - \mathbf{Adv}^{\text{hash-cr}}_{\mathsf{CR},\mathcal{A}'}(k) - q/p)$$

and $\mathbf{Time}_{\mathcal{B}}(k) = \mathbf{Time}_{\mathcal{A}}(k) + \mathcal{O}((\varepsilon^{-2}\ln(\varepsilon^{-1}) + q) \cdot \mathbf{Time}_{\mathbb{G}}(k))$, where $q$ is an upper bound on the number of decapsulation queries made by adversary $\mathcal{A}$ and $\mathbf{Time}_{\mathbb{G}}(k)$ is the time for a standard operation in $\mathbb{G}$.

The (quite technical) proof of Theorem B.1 combines the proof of Theorem 4.1 with Waters' hash [45] and is omitted. We remark that the above scheme can also be viewed as a tag-KEM [2].

# C Equivalence of the two security notions for KEMs

In this section we provide the original two-phase definition of IND-CCA security for KEMs [43] and prove it equivalent to our simplified definition.

**Definition C.1** Formally, we associate to an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ the following experiment:

**Experiment** $\mathbf{Exp}^{kem\text{-ccaf}}_{\mathcal{KEM},\mathcal{B}}(k)$
$(pk, sk) \xleftarrow{\$} \mathsf{Kg}(1^k)$
$state \xleftarrow{\$} \mathcal{B}_1^{\mathsf{DecO}(\cdot)}(pk)$
$K_0^* \xleftarrow{\$} \mathsf{KeySp}(k)$ ; $(K_1^*, C^*) \xleftarrow{\$} \mathsf{Enc}(pk)$
$\delta \xleftarrow{\$} \{0,1\}$
$\delta' \xleftarrow{\$} \mathcal{B}_2^{\mathsf{DecO}(\cdot)}(state, K_\delta^*, C^*)$
If $\delta \neq \delta'$ then return 0 else return 1

where the oracle $\mathsf{DecO}(C)$ returns $K \xleftarrow{\$} \mathsf{Dec}(sk, C)$ with the restriction that $\mathcal{B}$ is not allowed to query $\mathsf{DecO}(\cdot)$ on the target ciphertext $C^*$. We define the advantage of $\mathcal{B}$ in the left experiment as

$$\mathbf{Adv}^{kem\text{-}\mathrm{ccaf}}_{\mathcal{KEM},\mathcal{B}}(k) \;=\; \left|\Pr\left[\,\mathbf{Exp}^{kem\text{-}\mathrm{ccaf}}_{\mathcal{KEM},\mathcal{B}}(k) = 1\,\right] - \frac{1}{2}\right| .$$

A key encapsulation mechanism $\mathcal{KEM}$ is said to be *indistinguishable against chosen-ciphertext attacks* if the advantage function $\mathbf{Adv}^{kem\text{-}\mathrm{ccaf}}_{\mathcal{KEM},\mathcal{B}}(k)$ is a negligible function in $k$ for all polynomial-time adversaries $\mathcal{B}$.

To prove Theorem 2.2 we show the following:

**Lemma C.2** Suppose there exists an attacker $\mathcal{A}$ that attacks the IND-CCA security as defined in Definition 2.1 with advantage $\mathbf{Adv}^{kem\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{B}}(k)$. Then there exists an attacker $\mathcal{B}$ that attacks the IND-CCA security as defined in Definition C.1 with advantage $\mathbf{Adv}^{kem\text{-}\mathrm{ccaf}}_{\mathcal{KEM},\mathcal{B}}(k) \geq \mathbf{Adv}^{kem\text{-}\mathrm{cca}}_{\mathcal{KEM},\mathcal{B}}(k) - q_1/|\mathsf{KeySp}(k)|$, where $q_1$ is an upper bound on the number of decapsulation queries $\mathcal{B}_1$ makes (in the find phase).

**Proof:** The proof is simple (sketch). $\mathcal{A}$ recieves $(pk, K^*, C^*)$ from the experiment. $\mathcal{A}$ runs $\mathcal{B}_1$ on $pk$ and answers $\mathcal{B}_1$'s decapsulation queries using its own decapsulation oracle. This simulation is perfect unless $\mathcal{B}_1$ queries $C^*$. Since $C^*$ is information theoretically hidden from $\mathcal{B}_1$'s view this happens with probability at least $q_1/|\mathsf{KeySp}(k)|$. Eventually $\mathcal{B}_1$ terminates and outputs a state *state*. Then $\mathcal{B}_2$ is run on $(state, K^*, C^*)$ and again $\mathcal{B}_2$'s decapsulation queries are answered by $\mathcal{A}$. Eventually $\mathcal{B}_2$ terminates and $\mathcal{A}$ outputs whatever $\mathcal{B}_2$ outputs. ∎

# D Proof of Lemma 5.1

**Proof:** We first show the tight equivalence of GDH and GHDH in the (non-programmable) random oracle model, i.e. the non-trivial direction that GHDH implies GDH. Adversary $\mathcal{B}$ inputs an instance of the GDH problem, i.e. $\mathcal{B}$ inputs the values $(1^k, \mathbb{G}, g, h, g^r)$. $\mathcal{B}$'s goal is to compute $h^r$.

Now suppose there exists an adversary $\mathcal{A}$ that breaks the GHDH assumption with (non-negligible) advantage $\mathbf{Adv}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathsf{H},\mathcal{A}}(k)$. We show that adversary $\mathcal{B}$ can run adversary $\mathcal{A}$ to solve its instance of the GDH problem with the same advantage.

$\mathcal{B}$ picks a random $Y \in \{0,1\}^l$ and runs $\mathcal{A}$ on input $(1^k, \mathbb{G}, g, h, g^r, Y)$. The hash queries $\mathsf{H}(X)$ made by $\mathcal{A}$ are answered as follows. $\mathcal{B}$ maintains a table $(X, K(X))$ of all oracle queries. If $X$ was already queried the corresponding value $K(X)$ from the table is returned. If $X$ was not queried yet $\mathcal{B}$ tests if the tuple $(g, h, g^r, X)$ is a Diffie-Hellman tuple using the provided DDH oracle $\mathsf{DDHsolve}$. If the answer is yes then $\mathcal{B}$ stops the simulation and returns $X$. Otherwise $\mathcal{B}$ picks a new random value $K(X) \xleftarrow{\$} \{0,1\}^l$. Finally the values $(X, K(X))$ are stored in a table and $K(X)$ is returned.

Eventually $\mathcal{A}$ returns a bit $\gamma'$ and terminates. In that case $\mathcal{B}$ ignores the returned bit, returns a random value $X$ and terminates. This completes the description of $\mathcal{B}$. Note that the simulation is always perfect, i.e. the view of $\mathcal{A}$ is as in the GDH security experiment.

Now the standard argument is that when $\mathcal{A}$ has a chance in deciding between a random $Y$ and $Y = \mathsf{H}(h^r)$ then it must have asked the random oracle $\mathsf{H}$ for the value $h^r$. This query will be detected by $\mathcal{B}$ using the DDH oracle. That shows $\mathbf{Adv}^{\mathrm{gdh}}_{\mathsf{Gen},\mathcal{B}}(k) \geq \mathbf{Adv}^{\mathrm{ghdh}}_{\mathsf{Gen},\mathcal{A}}(k)$. Note that in the proof $\mathcal{B}$ does not "programm" the random oracle, it only observes queries to it made by $\mathcal{A}$. That is, in the reduction the hash function $\mathsf{H}$ is modeled as a non-programmable random oracle [35]. ∎