Earlier versions of of this work appear in [9] and [1]. This is the full version.

DHIES: An encryption scheme based on the Diffie-Hellman Problem

Michel Abdalla* Mihir Bellare† Phillip Rogaway‡

September 18, 2001

Abstract

This paper describes a Diffie-Hellman based encryption scheme, DHIES (formerly named DHES and DHAES), which is now in several (draft) standards. The scheme is as efficient as ElGamal encryption, but has stronger security properties. Furthermore, these security properties are proven to hold under appropriate assumptions on the underlying primitive. DHIES is a Diffie-Hellman based scheme that combines a symmetric encryption method, a message authentication code, and a hash function, in addition to number-theoretic operations, in a way which is intended to provide security against chosen-ciphertext attacks. The proofs of security are based on the assumption that the underlying symmetric primitives are secure and on appropriate assumptions about the Diffie-Hellman problem. The latter are interesting variants of the customary assumptions on the Diffie-Hellman problem, and we investigate relationships among them, and provide security lower bounds. Our proofs are in the standard model; no random-oracle assumption is required.

Keywords: Cryptographic standards, Diffie-Hellman key exchange, ElGamal encryption, elliptic curve cryptosystems, generic model, provable security.

---

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093. E-Mail: `mabdalla@cs.ucsd.edu`. URL: `http://www.michelabdalla./net`. Supported by CAPES under Grant BEX3019/95-2.

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093. E-Mail: `mihir@cs.ucsd.edu`. URL: `http://www-cse.ucsd.edu/users/mihir`. Supported by NSF CAREER Award CCR-9624439 and a 1996 Packard Foundation Fellowship in Science and Engineering.

‡Dept. of Computer Science, University of California at Davis, Davis, CA 95616. E-mail: `rogaway@cs.ucdavis.edu`. URL: `http://www.cs.ucdavis.edu/~rogaway`. Supported by NSF CAREER Award CCR-9624560 and funding provided by Certicom Corporation under MICRO Grant 97-150.

Contents

1 Introduction **1**

2 Definitions **3**
- 2.1 Represented groups 3
- 2.2 Message Authentication Codes 3
- 2.3 Symmetric Encryption 4
- 2.4 Asymmetric Encryption 5

3 The Scheme DHIES **6**

4 Attributes and Advantages of DHIES **7**
- 4.1 Encrypting with Diffie-Hellman: The ElGamal Scheme 8
- 4.2 Deficiencies of ElGamal Encryption 8
- 4.3 Overcoming Deficiencies in ElGamal Encryption: DHIES 9
- 4.4 More on Provable Security 10
- 4.5 Concrete Security 11

5 Diffie-Hellman Assumptions **11**

6 Security against Chosen-Plaintext Attack **15**

7 Security against Chosen-Ciphertext Attack **17**

8 ODH and SDH **21**

A Attacks on the ElGamal Scheme **27**

1 Introduction

This paper describes a method for encrypting strings using the Diffie-Hellman assumption. We are concerned with the "details" of Diffie-Hellman based encryption — how a message should be "packaged" in order to best exploit the group operations (e.g., modular exponentiation) which are at the core of a Diffie-Hellman based encryption.

The method we suggest is called DHIES, standing for "Diffie-Hellman Integrated Encryption Scheme". It is a simple extension of the ElGamal encryption scheme and is now in the draft standards of ANSI X9.63 and IEEE P1363a [2, 23] and in the corporate standard SECG [31]. The scheme was formerly known as DHES and as DHAES. It is all the same scheme.

DHIES uses symmetric encryption, message authentication, and hashing. This may seem like a lot of cryptography beyond the group operation, but it is exactly this additional cryptography which ensures, by and large, that we get our security guarantees.

The security analysis of DHIES requires some interesting new variants of the Diffie-Hellman assumption. We look at relationships among these notions, and we prove a complexity lower bound, in the generic model, about one of them.

BACKGROUND. DHIES is designed to be a natural extension of the ElGamal scheme, suitable in a variety of groups, and which enhanced ElGamal in a couple of ways important to cryptographic practice. First, the scheme needs to provide the capability of encrypting arbitrary bit strings (ElGamal requires that message be a group element). And second, the scheme should be secure against chosen-ciphertext attack (ElGamal is not). The above two goals have to be realized without increasing the number of group operations for encryption and decryption, and without increasing key sizes relative to ElGamal. Within these constraints, we want to provide the best possible provable-security analysis. But efficiency and practicality of the scheme should not be sacrificed in order to reduce assumptions.

The approach above is somewhat in contrast to related schemes in the literature. More typical is to fix an assumption and then strive to find the lowest cost scheme which can be proven secure under that assumption. Examples of work in this style are that of Cramer and Shoup [14] and that of Shoup [35], who start from the decisional Diffie-Hellman assumption, and then try to find the best scheme they can that will resist chosen-ciphertext attack under this assumption. In fact, the latter can also be proved secure in the random oracle model based on the weaker computational Diffie-Hellman assumption. These schemes are remarkable, but their costs are about double that of ElGamal, which is already enough to dampen some practical interest. A somewhat different approach was taken by Fujisaki and Okamoto [19], starting from weaker asymmetric and symmetric schemes to construct a stronger hybrid asymmetric scheme. Their scheme can be quite practical, but the proof of security relies heavily on the use of random oracles.

The DHIES scheme uses a hash function. In [9], a claim is made that DHIES should achieve plaintext awareness if this hash function is modeled as a public random oracle and one assumes the computational Diffie-Hellman assumption. In fact, technical problems would seem to thwart any possibility of pushing through such a result.

OUR APPROACH. DHIES is a very "natural" scheme. (See Section 3 for its definition.) The method follows standard ideas and practice. Intuitively, it is secure. Yet it seems difficult to prove security under existing assumptions about the Diffie-Hellman problem.

This situation seems to arise frequently. It seems often to be the case that we think certain methods are good, but we don't know how to prove that they are good starting from "standard" assumptions. We suggest that what we are seeing with DHIES is a manifestation of hardness properties of Diffie-Hellman problems which just haven't been made explicit so far.

In this paper we capture some of these hardness properties as formal assumptions. We will then show how DHIES can then be proven secure under these assumptions. Then we further explore these assumptions by studying their complexity in the generic model [34], and by studying how the assumptions relate to one other.

RELATED WORK. As we have indicated, the DHIES scheme first appears in [9]. No proof appears in that work. It was suggested that a proof of plaintext awareness [7, 5] could be achieved under the random-oracle model. However, no such proof has appeared, and technical difficulties would seem to bar it.

DHIES is now embodied in three (draft) standards [2, 23, 31]. All of these assume an elliptic curve group of prime order. To harmonize this paper with those standards, and to simplify complexity assumptions, we shall assume the the underlying group in which we work has prime order. When working with a group whose order is not prime a minor change can be made to the protocol so that it will still be correct. Namely, the value g^u should be fed into the hash function H.

Zheng and Seberry [37] have proposed an ElGamal-based scheme that uses universal one-way hash functions. Security of their scheme is not supported by proofs in the reductionist sense of modern cryptography. Lim and Lee [25] have pointed out that in some of the cryptosystems proposed in [37], the method of adding authentication capability may fail just under known plaintext attacks. A submission to IEEE P1363a based on [37] has been made by Zheng [36].

Another contemporaneous suggestion was put forward by Johnson, Matyas and Peyravian [24]. Assume that the message M already contains some redundancy (e.g., some number of fixed bits) and unpredictability (e.g., random bits have been embedded in M). Then to asymmetrically encrypt M, [24] suggest to subject it to 4 rounds of a Feistel network based on a function H, thereby obtaining a new string M'. Encrypt, using an arbitrary encryption primitive, an arbitrary piece of M'. It is plausible that if H is modeled as a random function then the above approach can be proven sound.

Cramer and Shoup describe an encryption scheme based on the decisional Diffie-Hellman problem which achieves provable security against adaptive chosen-ciphertext attack [14]. They prove their scheme secure under the decisional Diffie-Hellman assumption (and a collision-intractable hash function), or, in the random-oracle model, under the ordinary Diffie-Hellman assumption [33]. Their scheme is more costly than ours in terms of key sizes, encryption time, and decryption time (in particular, encryption takes five exponentiations), but the scheme is still practical.

The notions of indistinguishability and semantic security, and their equivalence under chosen-plaintext attack is due to [21]. The notion of chosen-ciphertext security that we use is due to [30]. Equivalences are further investigated by [5]. Note that the form of chosen-ciphertext security we use is the "strong" form, called CCA2 in [5].

OUTLINE. To specify our scheme in a compact and precise way, we first specify in Section 2 the "syntax" of an asymmetric encryption scheme and what it means for it to be secure. We also specify in Section 2 the syntax of the types of primitives which our asymmetric encryption scheme employs along with their security definitions. The specification of DHIES is then given in Section 3 and its attributes and advantages are discussed in Section 4.

The security of DHIES relies on variants of the Diffie-Hellman problem, which we introduce in Section 5. More specifically, we formalize three new Diffie-Hellman assumptions (though one of them, the hash Diffie-Hellman assumption, is essentially folklore). The assumptions are the *hash* Diffie-Hellman assumption (HDH), the *oracle* Diffie-Hellman assumption (ODH), and the *strong* Diffie-Hellman assumption (SDH). The HDH and ODH assumptions measure the sense in which a hash function H is "independent" of the underlying Diffie-Hellman problem. One often hears intuition asserting that two primitives are independent. Here is one way to define this. The SDH assumption formalizes, in a simple manner, that the "only" way to compute a value g^{uv} from g^v is to choose a value u and compute $(g^v)^u$. The definitions for both ODH and SDH have oracles which play a central role.

Section 6 shows that DHIES is secure against chosen-plaintext attacks. The HDH assumption is what is required to show this. In Section 7, we show that DHIES is secure against chosen-ciphertext attacks. The

ODH assumption is what is required to show this. Of course this means that DHIES is also secure against chosen-plaintext attacks [5] based on the ODH assumption, but in fact we can prove the latter using the HDH assumption (although we do not show it here), a much weaker one.

These two results make additional cryptographic assumptions: in the case of chosen-plaintext attacks, the security of the symmetric encryption scheme; in the case of chosen-ciphertext attacks, the security of the symmetric encryption scheme and the security of the message authentication code. But the particular assumptions made about these primitives are extremely weak.

The ODH assumption is somewhat technical; SDH is rather simpler. In Section 8, we show that, in the random-oracle model, the SDH assumption implies the ODH assumption. A lower bound for the difficulty of the SDH assumption in the generic model of Shoup [34] is also given in Section 8. This rules out a large class of efficient attacks.

Following works such as [7, 8], we take a concrete, quantitative approach for all of the results above.

2 Definitions

2.1 Represented groups

DHIES makes use of a finite cyclic group $G = \langle g \rangle$. (This notation indicates that G is generated by the group element g.) We will use multiplicative notation for the group operation. So, for $u \in \mathsf{N}$, g^u denotes the group element of G that results from multiplying u copies of g. Naturally, g^0 names the identity element of G. Note that, if $u \in \mathsf{N}$, then, by Lagrange's theorem, $g^u = g^{u \bmod |G|}$.

Algorithms which operate on G will be given string representations of elements in G. We thus require an injective map $\_ : G \to \{0,1\}^{gLen}$ associated to G, where $gLen$ is some number (the length of the representation of group elements). Similarly, when a number $i \in \mathsf{N}$ is an input to, or output of, an algorithm, it must be appropriately encoded, say in binary. We assume all necessary encoding methods are fixed, and do not normally write the $\_$ operators.

Any "reasonable" group supports a variety of computationally feasible group operations. Of particular interest is there being an algorithm $\uparrow$ which takes (the representations of) a group element x and a number i and computes (the representation of) x^i. For clarity, we write this operator in infix, so that $(x) \uparrow (i)$ returns x^i. We will call the tuple $\mathcal{G} = (G, g, \_, \uparrow)$ a *represented group*.

2.2 Message Authentication Codes

Let $\mathsf{Message} = \{0,1\}^*$ and let $\mathsf{mKey} = \{0,1\}^{mLen}$ for some number $mLen$. Let $\mathsf{Tag} = \{0,1\}^{tLen}$ for some number $tLen$ (a superset of the possible tags). A *message authentication code* is a pair of algorithms $\mathrm{MAC} = (\mathcal{T}, \mathcal{V})$. Algorithm $\mathcal{T}$ (the *MAC generation algorithm*) takes a key $k \in \mathsf{mKey}$ and a message $x \in \mathsf{Message}$ and returns a string $\mathcal{T}(k, x)$. This string is called the *tag*. Algorithm $\mathcal{V}$ (the *MAC verification algorithm*) takes a key $k \in \mathsf{mKey}$, a message $x \in \mathsf{Message}$, and a purported tag $\tau \in \mathsf{Tag}$. It returns a bit $\mathcal{V}(k, x, \tau) \in \{0,1\}$, with 0 indicating that the message was rejected (deemed unauthentic) and 1 indicating that the message was accepted (deemed authentic). We require that for all $k \in \mathsf{mKey}$ and $x \in \mathsf{Message}$, $\mathcal{V}(k, x, \mathcal{T}(k, x)) = 1$. The first argument of either algorithm may be written as a subscript.

SECURITY. The security of a MAC is defined by an experiment in which we first choose a random key $k \in \mathsf{mKey}$ and then give an adversary F a $\mathcal{T}_k(\cdot)$ oracle, we say that F's output (x^*, τ^*) is *unasked* if τ^* is not the response of the $\mathcal{T}_k(\cdot)$ oracle to an earlier query of x^*. Our definition of MAC security follows.

Definition 1 Let $\mathrm{MAC} = (\mathcal{T}, \mathcal{V})$ be a message authentication scheme and let F be an adversary. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}^{\text{suf-cma}}_{\text{MAC},F} \\
\quad k \stackrel{R}{\leftarrow} \mathsf{mKey} \\
\quad (x^*, \tau^*) \leftarrow F^{\mathcal{T}_k(\cdot), \mathcal{V}_k(\cdot,\cdot)} \\
\quad \texttt{if } \mathcal{V}_k\,(x^*, \tau^*) = 1 \texttt{ and } \tau^* \text{ was never returned by } \mathcal{T}_k(\cdot) \text{ in response to query } x^* \\
\quad \texttt{then return } 1 \texttt{ else return } 0
\end{array}
$$

Now define the *suf-cma-advantage* of F as follows:

$$\mathbf{Adv}^{\text{suf-cma}}_{\text{MAC},F} \;=\; \Pr[\,\mathbf{Exp}^{\text{suf-cma}}_{\text{MAC},F} = 1\,]\;.$$

For any t, q_t, μ_t, q_v, and μ_t, we define the *suf-cma-advantage* of MAC as

$$\mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t, q_t, \mu_t, q_v, \mu_v) \;=\; \max_F\{\,\mathbf{Adv}^{\text{suf-cma}}_{\text{MAC},F}\,\}$$

where the maximum is over all F with time-complexity t, making to the tag oracle at most q_t queries the sum of whose lengths is at most μ_t bits and making to the verification oracle at most q_v queries the sum of whose lengths is at most μ_v bits. $\diamondsuit$

We say adversary F has *forged* when, in the experiment above, it outputs a pair (x^*, τ^*) such that $\mathcal{V}_k\,(x^*, \tau^*) = 1$ and (x^*, τ^*) was not previously obtained via a query to the tag oracle.

This definition is stronger than the usual one as given in [6]. There, one asks that the adversary not be able to produce MACs of new messages. Here we require additionally that the adversary not be able to generate new MACs of old messages. However, if the MAC generation function is deterministic and verification is done by simply re-computing the MAC (this is typically true) then there is no difference.

CANDIDATES. Candidate algorithms include HMAC [3] or the CBC MAC (but only a version that is correct across messages of arbitrary length).

2.3 Symmetric Encryption

Let $\mathsf{Message}$ be as before, and let $\mathsf{eKey} = \{0,1\}^{eLen}$, for some number $eLen$. Let $\mathsf{Ciphertext} = \{0,1\}^*$ (a superset of all possible ciphertexts). Let Coins be a synonym for $\{0,1\}^\infty$ (the set of infinite strings). A *s*ymmetric encryption scheme is a pair of algorithms $\text{SYM} = (\mathcal{E}, \mathcal{D})$. Algorithm $\mathcal{E}$ (the *e*ncryption algorithm) takes a key $k \in \mathsf{eKey}$, a plaintext $x \in \mathsf{Message}$, and coins $r \in \mathsf{Coins}$, and returns ciphertext $\mathcal{E}(k, x, r)$. Algorithm $\mathcal{D}$ (the *decryption algorithm*) takes a key $k \in \mathsf{eKey}$ and a purported ciphertext $y \in \mathsf{Ciphertext}$, and returns a value $\mathcal{D}(k, y) \in \mathsf{Message} \cup \{\text{BAD}\}$. We require that for all $x \in \mathsf{Message}$, $k \in \mathsf{Key}$, and $r \in \mathsf{Coins}$, $\mathcal{D}(k, \mathcal{E}(k, x, r)) = x$. Usually we omit mentioning the coins of $\mathcal{E}$, thinking of $\mathcal{E}$ as a probabilistic algorithm, or thinking of $\mathcal{E}(k, x)$ as the induced probability space. A return value of BAD from $\mathcal{D}$ is intended to indicate that the ciphertext was regarded as "invalid" (it is not the encryption of any plaintext). The first argument of either algorithm may be written as a subscript.

SECURITY. Security of a symmetric encryption scheme is defined as in [4], in turn an adaptation of the notion of polynomial security as given in [21, 26]. We imagine an adversary A that runs in two stages. During either stage the adversary may query an encryption oracle $\mathcal{E}(k, \cdot)$ which, on input x, returns $\mathcal{E}(k, x, r)$ for a randomly chosen r. In the adversary's find stage it endeavors to come up with a pair of equal-length messages, x_0 and x_1, whose encryptions it wants to try to tell apart. It also retains some state information s. In the adversary's guess stage it is given a random ciphertext y for one of the plaintexts x_0, x_1, together with the saved state s. The adversary "wins" if it correctly identifies which plaintext goes with y. The encryption scheme is "good" if "reasonable" adversaries can't win significantly more than half the time.

Definition 2 [4] Let $\text{SYM} = (\mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let A be an adversary. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}^{\text{ind-cpa-fg}}_{\text{SYM},A} \\
\quad k \xleftarrow{R} \textsf{eKey} \\
\quad (x_0, x_1, s) \leftarrow A^{\mathcal{E}(k,\cdot)}(\textsf{find}) \\
\quad b \xleftarrow{R} \{0,1\} \\
\quad y \leftarrow \mathcal{E}(k, x_b) \\
\quad \widetilde{b} \leftarrow A^{\mathcal{E}(k,\cdot)}(\textsf{guess}, y, s) \\
\quad \texttt{if } \widetilde{b} = b \texttt{ then return } 1 \texttt{ else return } 0
\end{array}
$$

Now define the *ind-cpa-advantage* of A in the find-and-guess notion as follows:

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},A} \quad = \quad 2 \cdot \Pr[\,\mathbf{Exp}^{\text{ind-cpa-fg}}_{\text{SYM},A} = 1\,] - 1$$

if A is legitimate, and 0 otherwise. For any t, q, and μ, we define the *ind-cpa-advantage* of SYM as

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t, q, \mu) \quad = \quad \max_{A}\{\,\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},A}\,\}$$

where the maximum is over all A with time-complexity t, making to the encryption oracle at most q queries the sum of whose lengths is at most μ bits. $\diamondsuit$

It is understood that, above, A must output x_0 and x_1 with $|x_0| = |x_1|$. The multiplication by 2 and subtraction by 1 are just scaling factors, to make a numeric value of 0 correspond to no advantage and a numeric value of 1 correspond to perfect advantage. As a reminder, "time-complexity" is the maximum execution time of the experiment $\mathbf{Exp}^{\text{ind-cpa-fg}}_{\text{SYM},A}$ plus the size of the code for A, all in some fixed RAM model of computation.

CANDIDATES. One candidate algorithms for the symmetric encryption are CBC encryption and Vernam cipher encryption.

2.4 Asymmetric Encryption

Let Coins, Message, Ciphertext be as before and let $\textsf{PK} \subseteq \{0,1\}^*$ and $\textsf{SK} \subseteq \{0,1\}^*$ be sets of strings. An *asymmetric encryption scheme* is a three-tuple of algorithms $\text{ASYM} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$. The *encryption algorithm* $\overline{\mathcal{E}}$ takes a public key $pk \in \textsf{PK}$, a plaintext $x \in \textsf{Message}$, and coins $r \in \textsf{Coins}$, and returns a ciphertext $y = \overline{\mathcal{E}}(k, x, r)$. The decryption algorithm $\overline{\mathcal{D}}$ takes a secret key $sk \in \textsf{SK}$ and a ciphertext $y \in \textsf{Ciphertext}$, and returns a plaintext $\overline{\mathcal{D}}(sk, y) \in \textsf{Message} \cup \{\text{BAD}\}$. The key generation algorithm $\overline{\mathcal{K}}$ takes coins $r \in \textsf{Coins}$ and returns a pair $(pk, sk) \in \textsf{PK} \times \textsf{SK}$. We require that for all (pk, sk) which can be output by $\overline{\mathcal{K}}$, for all $x \in \textsf{Message}$ and $r \in \textsf{Coins}$, we have that $\overline{\mathcal{D}}(sk, \overline{\mathcal{E}}(pk, x, r)) = x$. The first argument to $\overline{\mathcal{E}}$ and $\overline{\mathcal{D}}$ may be written as a subscript.

PRIVACY AGAINST CHOSEN-PLAINTEXT ATTACK. Our treatment mimics the find-then-guess notion of [4] and follows [21, 26, 20]. The definition is similar to Definition 2, so we state it without further discussion.

Definition 3 Let $\text{ASYM} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ be an asymmetric encryption scheme and let A an adversary. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}^{\text{ind-cpa-fg}}_{\text{ASYM},A} \\
\quad (sk, pk) \leftarrow \overline{\mathcal{K}} \\
\quad (x_0, x_1, s) \leftarrow A(\textsf{find}, pk) \\
\quad b \xleftarrow{R} \{0,1\} \\
\quad y \leftarrow \overline{\mathcal{E}}_{pk}(x_b) \\
\quad \widetilde{b} \leftarrow A(\textsf{guess}, pk, y, s) \\
\quad \texttt{if } \widetilde{b} = b \texttt{ then return } 1 \texttt{ else return } 0
\end{array}
$$

Now define the *ind-cpa-advantage* of A in the find-and-guess notion as follows:

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{ASYM},A} = 2 \cdot \Pr[\,\mathbf{Exp}^{\text{ind-cpa-fg}}_{\text{ASYM},A} = 1\,] - 1$$

if A is legitimate, and 0 otherwise. For any t, we define the *ind-cpa-advantage* of ASYM as

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{ASYM}}(t, c) = \max_{A}\{\,\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{ASYM},A}\,\}$$

where the maximum is over all A with time-complexity t and whose challenge has length at most c bits. $\diamondsuit$

PRIVACY AGAINST ADAPTIVE CHOSEN-CIPHERTEXT ATTACK. The definition of chosen-ciphertext security of an asymmetric encryption scheme is very similar to that given in Definition 3. The difference is that here the adversary is given access to a decryption oracle in both stages. So we state it without further discussion.

Definition 4 Let $\text{ASYM} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$ be an asymmetric encryption scheme and let A an adversary for its chosen-ciphertext security. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}^{\text{ind-cca-fg}}_{\text{ASYM},A} \\
\quad (sk, pk) \leftarrow \overline{\mathcal{K}} \\
\quad (x_0, x_1, s) \leftarrow A^{\overline{\mathcal{D}}_{sk}}(\mathsf{find}, pk) \\
\quad b \xleftarrow{R} \{0,1\} \\
\quad y \leftarrow \overline{\mathcal{E}}_{pk}(x_b) \\
\quad \tilde{b} \leftarrow A^{\overline{\mathcal{D}}_{sk}}(\mathsf{guess}, pk, y, s) \\
\quad \texttt{if } \tilde{b} = b \texttt{ then return } 1 \texttt{ else return } 0
\end{array}
$$

Now define the *ind-cca-advantage* of A in the find-and-guess notion as follows:

$$\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{ASYM},A} = 2 \cdot \Pr[\,\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{ASYM},A} = 1\,] - 1$$

if A is legitimate, and 0 otherwise. For any t, we define the *ind-cpa-advantage* of ASYM as

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{ASYM}}(t, c) = \max_{A}\{\,\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{ASYM},A}\,\}$$

where the maximum is over all A with time-complexity t, making to the decryption oracle at most q queries the sum of whose lengths is at most μ bits. $\diamondsuit$

3 The Scheme DHIES

This section recalls the DHIES scheme. Refer to Figure 1 for a pictorial representation of encryption under DHIES, and Figure 2 for the formal definition of the algorithm. Let us explain the scheme in reference to those descriptions.

Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, where group elements are represented by strings of $gLen$ bits. Let $\text{SYM} = (\mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with key length $eLen$, and let $\text{MAC} = (\mathcal{T}, \mathcal{V})$ be a message authentication code with key length $mLen$ and tag length $tLen$. Let $H : \{0,1\}^{gLen} \to \{0,1\}^{mLen+eLen}$ be a function. From these primitives we define the asymmetric encryption scheme $\text{DHIES} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$. If we want to explicitly indicate the dependency of DHIES on its associated primitives, then we will write DHIES $[\![\mathcal{G}, \text{SYM}, \text{MAC}, \text{H}]\!]$. The component algorithms of DHIES are defined in Figure 2.

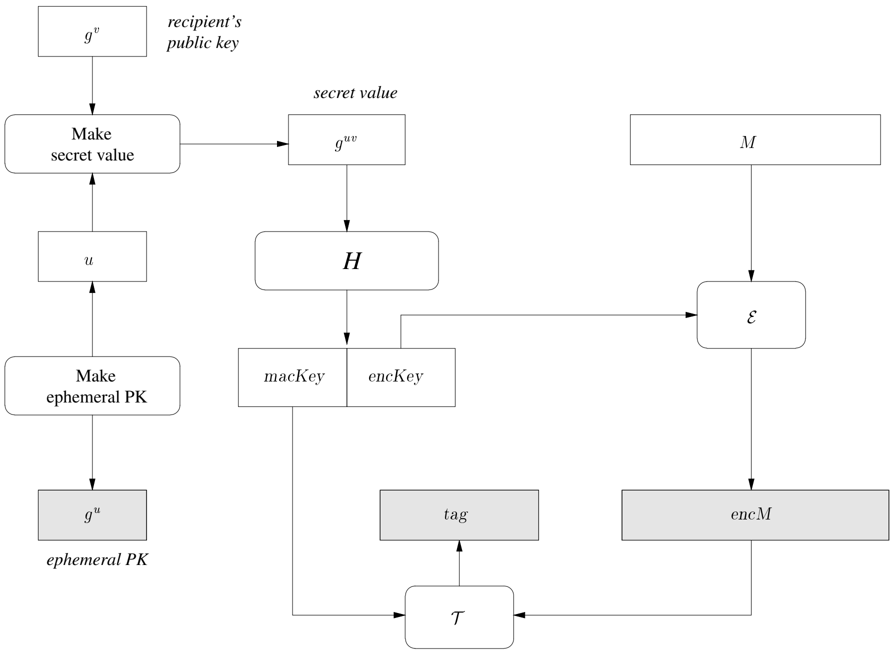

Figure 1: *Encrypting with the scheme DHIES. We use the symmetric encryption algorithm, $\mathcal{E}$, of* SYM*; the MAC generation algorithm, $\mathcal{T}$, of* MAC*; and a hash function, H. The shaded rectangles comprise the ciphertext.*

Each user's public key and secret key is exactly the same as with the ElGamal scheme: g^v and v, respectively, for a randomly chosen v. (Here we will not bother to distinguish group elements and their bit-string representations.) To send a user an encrypted message we choose a random u and compute an "ephemeral public key," g^u. Including g^u in the ciphertext provides an "implicit" Diffie-Hellman key exchange: the sender and receiver will both be able to compute the "secret value" g^{uv}. We pass g^{uv} to the hash function H and parse the result into two pieces: a MAC key, $macKey$, and an encryption key, $encKey$. We symmetrically encrypt the message we wish to send with the encryption key, and we MAC the resulting ciphertext using the MAC key. The ciphertext consists of the ephemeral public key, the symmetrically encrypted plaintext, and the authentication tag generated by the MAC.

THE GROUP G IS OF PRIME ORDER. We henceforth assume that $|G|$ is prime. This is extremely important to ensure the security of DHIES or otherwise the scheme could be malleable. The reason stems from the fact that in groups where $|G|$ is not a prime (e.g., $\mathbb{Z}_p^*$), g^{uv} and g^v together might not uniquely determine g^u. That is, there may exist two values u and u' such that $u \neq u'$ but $g^{uv} = g^{u'v}$. As a result, both u and u' would produce two different valid ciphertexts for the same plaintext. Therefore, if one can compute $g^{u'}$, given g^u and g^v, such that $g^{uv} = g^{u'v}$ holds with high probability, then we would break the scheme in the malleability sense. To prevent such attacks in groups not of prime order (e.g., $\mathbb{Z}_p^*$), one should feed g^u to H.

4 Attributes and Advantages of DHIES

To explain the problem which DHIES solves, and the sense in which it solves this problem, let us back up and provide a bit of background.

`algorithm` $\overline{\mathcal{E}}(pk, M)$
`begin`
$u \leftarrow \{1, \ldots, \|G\|\}$
$X \leftarrow pk \uparrow u$
$U \leftarrow g \uparrow u$
$hash \leftarrow H(X)$
$macKey \leftarrow hash[1 \,..\, mLen]$
$encKey \leftarrow hash[mLen + 1 \,..\, mLen + eLen]$
$encM \leftarrow \mathcal{E}(encKey, M)$
$tag \leftarrow \mathcal{T}(macKey, M)$
$EM \leftarrow U \parallel encM \parallel tag$
`return` EM
`end`
	`algorithm` $\overline{\mathcal{D}}(sk, EM)$
	`begin`
	$U \parallel encM \parallel tag \leftarrow EM$
	$X \leftarrow U \uparrow sk$
	$hash \leftarrow H(X)$
	$macKey \leftarrow hash[1 \,..\, mLen]$
	$encKey \leftarrow hash[mLen + 1 \,..\, mLen + eLen]$
	`if` $\mathcal{V}(macKey, encM, tag) = 0$
	`then return` BAD
	$M \leftarrow \mathcal{D}(encKey, encM)$
	`return` M
	`end`
		`algorithm` $\overline{\mathcal{K}}$
		`begin`
		$v \leftarrow \{1, \ldots, \|G\|\}$
		$pk \leftarrow g \uparrow v$
		$sk \leftarrow v$
		`return` (pk, sk)
		`end`

Figure 2: *The scheme* $\mathrm{DHIES} = (\overline{\mathcal{E}}, \overline{\mathcal{D}}, \overline{\mathcal{K}})$, *where:* $\mathrm{SYM} = (\mathcal{E}, \mathcal{D})$ *is a symmetric encryption scheme using keys of length* $eLen$*;* $\mathrm{MAC} = (\mathcal{T}, \mathcal{V})$ *is a message authentication code with keys of length* $mLen$ *and tags of length* $tLen$*;* $\mathcal{G} = (G, g, \_, \uparrow)$ *is a represented group whose group elements encoded by strings of length* $gLen$*; and* $H : \{0,1\}^{gLen} \to \{0,1\}^{eLen+mLen}$.

4.1 Encrypting with Diffie-Hellman: The ElGamal Scheme

Let G be a finite cyclic group, say $G = Z_p^*$, the multiplicative group of integers modulo a (large) prime p. We'll denote the group operation of G multiplicatively, so that repeated multiplication is represented by exponentiation. Let g be a generator for G; that is, the elements of G are $\{g^1, g^2, \ldots, g^{|G|}\}$. Fix such a group G and its generator g. All multiplications (or exponentiations, which is just shorthand for repeated multiplication) will be performed in G.

Diffie and Hellman suggested that two parties communicating over a channel subject to (passive) eavesdropping could come to share a secret key as follows [16]. The first party chooses a random number $u \in \{1, \ldots, |G|\}$ and sends g^u to the second party. The second party chooses a random number $v \in \{1, \ldots, |G|\}$ and sends g^v to the first party. The shared key is declared to be g^{uv}, which the first party can calculate as $(g^v)^u$ and the second party can calculate at $(g^u)^v$.

Roughly said, the *Diffie-Hellman assumption* for G asserts that an adversary who sees g^u and g^v (for a random u, v) cannot compute g^{uv}.

ElGamal [18] explained how to adapt the above to give a public key encryption method. The intended receiver of an encrypted message has a public key which specifies g^v (where v was chosen randomly from $\{1, \ldots, |G|\}$). The sender wants to send to that receiver a ciphertext C which is the encryption of a message M. We assume $M \in G$. The sender computes C by choosing a random u (again in $\{1, \ldots, |G|\}$) and transmitting $C = (g^u, M \cdot g^{uv})$. Knowing v, the receiver can compute $g^{uv} = (g^u)^v$ from C and then multiply $M \cdot g^{uv}$ by the inverse of g^{uv} to recover M.

4.2 Deficiencies of ElGamal Encryption

We highlight a number of issues arising from the encryption method we have just described.

1. *Limited message space.* First there was the assumption that $M \in G$. Messages are naturally regarded as bit strings, not group elements. Often there will be a natural embedding of *some* bit strings into group elements, but that may fall short of all potential messages.

2. *May not provide good privacy.* As Goldwasser and Micali explain and formalize in [21], a good encryption scheme should do more than make it infeasible for an adversary to decrypt: the scheme should conceal from an adversary mounting a passive attack "any" information about the plaintext. For example, it should not be possible to determine even one bit of the plaintext given the ciphertext. This property has been defined in several ways which have been shown to be equivalent [21], including a definitions known as "indistinguishability" and one known as "semantic security."

Even in groups for which one anticipates using ElGamal encryption, the ElGamal encryption does not achieve semantic security. For example, when the scheme is implemented in the group $G = \mathbb{Z}_p^*$, there are attacks showing that some information about the plaintext can be determined from the ciphertext. See Appendix A for a description of such an attack.

It is possible to guarantee the semantic security of ElGamal encryption if it is done in special groups, and if we make a stronger assumption about the Diffie-Hellman problem. Specifically, the order of the group should be prime (note the order of $\mathbb{Z}_p^*$ is $p-1$ which is not prime) and we make the *decisional Diffie-Hellman* assumption, which says that it is infeasible to distinguish the following two distributions: (g^u, g^v, g^{uv}), for a random u and v, and (g^u, g^v, g^z), for a random u,v, and z. This is a very strong assumption.

It would be preferable to have a scheme which worked in any group where the Diffie-Hellman problem is hard, and one which was guaranteed to achieve semantic security under a weaker number-theoretic assumption.

3. *We want more than basic privacy.* For an encryption scheme to be a maximally useful tool in the design of higher-level protocols it should actually do *m*ore than shield information about the plaintext in the presence of a passive attack. Stronger goals include non-malleability [15] and chosen-ciphertext security [28, 30]. Informally, non-malleability means that an adversary cannot mutate one ciphertext into a related one. Chosen-ciphertext security means that an adversary cannot break an encryption scheme even if it can cause some ciphertexts to be decrypted. ElGamal encryption achieves neither of these "beyond semantic security" goals: it is easy to see that the scheme is malleable and also insecure under a chosen-ciphertext attack. (See Appendix A).

We are finding that uses of encryption in cryptographic practice relies more and more on the scheme meeting these "beyond semantic security" goals. For example, the designers of SET (Secure Electronic Transactions) mandated the use of an encryption scheme which achieves more than semantic security. This was necessary, in the sense that the SET protocols would be *w*rong if instantiated by a primitive which achieves *o*nly semantic security, and to design SET-like protocols using a primitive which achieves only semantic security would seem to yield more complicated protocols. As a second example, Bleichenbacher [10] has shown that encryption under RSA PKCS #1 v1.5 [29] is vulnerable to chosen-ciphertext attack, and he goes on to demonstrate how this leads to an attack on SSL 3.0. Because schemes which achieve "only" semantic security are so easily misused by protocol designers, we believe it is highly desirable that standardized schemes achieve "beyond semantic security" goals, particularly non-malleability and chosen-ciphertext security.

4.3 Overcoming Deficiencies in ElGamal Encryption: DHIES

The scheme we have presented, DHIES, does Diffie-Hellman based encryption in a way which overcomes the limitations enumerated above, but without significant increase in cost compared to ElGamal. Key characteristics and advantages of DHIES include the following.

1. *Basic privacy — Proven in the sense of provable security.* Roughly said, to achieve semantic security we assume the existence of a function $H : G \to \{0,1\}^*$ such that $\langle g^u, g^v, H(g^{uv})\rangle$ looks like a pair of random group elements together with a random string. For non-trivial functions H this assumption —that H is hard-core for the Diffie-Hellman problem on G— would seem to be weaker than decisional Diffie-Hellman.

We prove that under this assumption, our scheme achieves semantic security. For reasonable choices of H, this assumption would seem to hold for any group one would imagine using, not just particular groups.

2. *Beyond basic privacy: non-malleability and chosen-ciphertext security — proven in the sense of provable security.* We prove that our scheme is secure against adaptive chosen-ciphertext attacks. This is proved under an assumption called the Oracle Diffie-Hellman assumption, and assuming the underlying MAC and encryption schemes are secure. It is shown in [5, 17] that security under adaptive chosen-ciphertext attack implies non-malleability, so that property is achieved automatically.

3. *No random oracles.* The proofs here do not appeal to the random oracle model. They are all in the standard model. This addresses concerns that have been raised about this model [13].

4. *Efficiency.* The efficiency of ElGamal encryption is preserved: the cost of encryption is essentially the same as with ElGamal encryption: two exponentiations to encrypt, one to decrypt. For encryption, both of these exponentiations can be *o*ff-line, meaning that they can be done even before the message M is known. The length of ciphertexts and the public key is the same as in ElGamal.

5. *Versatile instantiation — The group.* We allow considerable versatility in instantiating DHIES. First, the group G in which we perform our operations can be essentially any group in which our version of the Diffie-Hellman assumption is reasonable. It could be Z_p^*, or a subgroup of Z_n^*, or an elliptic curve group (in which case the group operation is usually written additively, so what we have been denoting g^u would be written multiplicatively, as ug). Our proofs assume no algebraic structure for G beyond its being a finite cyclic group.

6. *Versatile instantiation — Ancillary primitives.* Cryptography beyond the group operations is performed using generic primitives. We employ primitives for symmetric encryption, message authentication, and hashing. For achieving semantic security, the underlying symmetric encryption and hashing schemes must meet weak, formalized assumptions. For achieving non-malleability and chosen-ciphertext security the encryption scheme and message authentication code must meet weak, formalized assumptions, while the hash function is modeled by a public random oracle.

7. *Arbitrary message space.* Finally, messages to be encrypted are arbitrary bit strings; messages are not restricted in length or content.

4.4 More on Provable Security

It is easy to come up with a Diffie-Hellman-based encryption scheme which *might* work well when its primitives (cryptographic hash function, universal hash families, etc.) are concretely instantiated, in the sense that no attacks seem discernible. What we do here is provide a greater assurance of security, by proving that the scheme meets formally defined objectives under given model and complexity-theoretic assumptions.

Let us explain. A cryptographic scheme S based on a primitive P is said to be *provably secure* if the security of P has been demonstrated to imply the security of S. More precisely, we use this phrase when someone has formally defined the goals G_P and G_S for some primitive P and scheme S, respectively; and then has proven that the existence of an adversary A_S who breaks scheme S, in the sense of violating G_S, implies the existence of an adversary A_P who breaks primitive P, in the sense of violating G_P.

What provable security means is that as long as we are ready to believe that P is secure, then there are no attacks on S. This obviates the need to consider any specific cryptanalytic attacks on S.

4.5 Concrete Security

Following works such as [7, 8], we take a concrete, quantitative approach to proving security. Let S be an encryption scheme which makes use of a primitive P, and let A_S be an adversary which attacks S. To show the security of S one converts A_S into an adversary A_P which attacks P. Ideally, A_P should use the same computational resources as A_S and, with this investment in resources, A_P should be just as successful in attacking P as A_S was successful in attacking S. This way "practical" attacks on P imply practical attacks on S, and so the assumed *absence* of practical attacks on P implies the absence of practical attacks on S.

To quantify how close to this ideal we come we define the success probability of A_P attacking P, we define the success probability of A_S attacking S, and then we give concrete formulas to show how A_P's computational resources and success probability depend on A_S's computational resources and success probability. These formulas measure the demonstrated security. By giving explicit formulas we make statements which are more precise than those that are given in doing asymptotic analyses of reductions.

5 Diffie-Hellman Assumptions

This section specifies five versions of the Diffie-Hellman assumption. The first two are standard (included here only for completeness); the next one is straightforward/folklore; and the last assumptions are new.

COMPUTATIONAL DIFFIE-HELLMAN ASSUMPTION: CDH. We refer to the "standard" Diffie-Hellman assumption as the *computational Diffie-Hellman* assumption, CDH. It states that given g^u and g^v, where u, v were drawn at random from $\{1, \ldots, |G|\}$, it is hard to compute g^{uv}. Under the computational Diffie-Hellman assumption it might well be possible for the adversary to compute something interesting about g^{uv} given g^u and g^v; for example, the adversary might be able to compute the most significant bit, or even half of the bits. This makes the assumption too weak to directly use in typical applications. For example, the ElGamal scheme is not semantically secure given only this assumption.

Definition 5 [Computational Diffie-Hellman: CDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let A be an adversary. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},A} \\
\quad u \stackrel{R}{\leftarrow} \{1, \ldots, |G|\};\ U \leftarrow g^u \\
\quad v \stackrel{R}{\leftarrow} \{1, \ldots, |G|\};\ V \leftarrow g^v \\
\quad Z \leftarrow A(U, V) \\
\quad \texttt{if } Z = g^{uv} \texttt{ then } b \leftarrow 1 \texttt{ else } b \leftarrow 0 \\
\quad \texttt{return } b
\end{array}
$$

Now define the *advantage* of A in violating the computational Diffie-Hellman assumption as

$$\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G},A} = \Pr[\,\mathbf{Exp}^{\mathrm{cdh}}_{\mathcal{G},A} = 1\,]\,. \qquad \diamondsuit$$

DECISIONAL DIFFIE-HELLMAN ASSUMPTION: DDH. A stronger assumption that has been gaining popularity is the *decisional Diffie-Hellman* assumption, DDH. (For a nice discussion, see Boneh's survey [11].) It states, roughly, that the distributions (g^u, g^v, g^{uv}) and (g^u, g^v, g^w) are computationally indistinguishable when u, v, w are drawn at random from $\{1, \ldots, |G|\}$. This assumption can only hold in a group G whose order does not contain small prime factors (e.g., subgroup of order q of $\mathbb{Z}_p^*$ for large primes p and q). In such groups the assumption suffices to prove the semantic security of the ElGamal scheme.

Definition 6 [Decisional Diffie-Hellman: DDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let A be an adversary. Consider the experiments

$$
\begin{array}{l|l}
\texttt{experiment } \mathbf{Exp}^{\text{ddh-real}}_{\mathcal{G},A} & \texttt{experiment } \mathbf{Exp}^{\text{ddh-rand}}_{\mathcal{G},A} \\
\quad u \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ U \leftarrow g^u & \quad u \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ U \leftarrow g^u \\
\quad v \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ V \leftarrow g^v & \quad v \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ V \leftarrow g^v \\
\quad Z \leftarrow g^{uv} & \quad z \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ Z \leftarrow g^z \\
\quad b \leftarrow A(U,V,Z) & \quad b \leftarrow A(U,V,Z) \\
\quad \texttt{return } b & \quad \texttt{return } b
\end{array}
$$

Now define the *advantage* of A in violating the decisional Diffie-Hellman assumption as

$$\mathbf{Adv}^{\text{ddh}}_{\mathcal{G},A} \quad = \quad \Pr[\,\mathbf{Exp}^{\text{ddh-real}}_{\mathcal{G},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ddh-rand}}_{\mathcal{G},A} = 1\,]\,. \qquad \diamondsuit$$

The assumption we make to prove security for DHIES under chosen-plaintext attack is weaker than DDH but stronger than CDH. It is called the *hash Diffie-Hellman* assumption, HDH. To prove the security of DHIES under chosen-ciphertext attacks, we will make stronger versions of the Hash Diffie-Hellman assumptions which say the assumption is true even when the adversary has additional power in the form of oracles giving certain kinds of information about other, independent Diffie-Hellman keys. The precise formulation of all three of our assumptions is below, and they are followed by a discussion on the choice of hash functions suitable for these assumptions.

HASH DIFFIE-HELLMAN ASSUMPTION: HDH. As indicated above, semantic security of a Diffie-Hellman-based scheme requires that we be able to get some number of "hard-core" bits from the Diffie-Hellman key, namely key derived bits that cannot be distinguished from random bits. Our assumption is that applying a suitable hash function H to g^{uv} will yield such bits. The assumption we make, called the *Hash Diffie-Hellman* assumption, HDH, is a "composite" one—it concerns the interaction between a hash function H and the group operations in G. Here is the definition.

Definition 7 [Hash Diffie-Hellman: HDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let $hLen$ be a number, let $H : \{0,1\}^* \to \{0,1\}^{hLen}$, and let A be an adversary. Consider the experiments

$$
\begin{array}{l|l}
\texttt{experiment } \mathbf{Exp}^{\text{hdh-real}}_{\mathcal{G},\mathrm{H},A} & \texttt{experiment } \mathbf{Exp}^{\text{hdh-rand}}_{\mathcal{G},\mathrm{H},A} \\
\quad u \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ U \leftarrow g^u & \quad u \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ U \leftarrow g^u \\
\quad v \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ V \leftarrow g^v & \quad v \stackrel{R}{\leftarrow} \{1,\ldots,|G|\};\ V \leftarrow g^v \\
\quad Z \leftarrow H(g^{uv}) & \quad Z \stackrel{R}{\leftarrow} \{0,1\}^{hLen} \\
\quad b \leftarrow A(U,V,Z) & \quad b \leftarrow A(U,V,Z) \\
\quad \texttt{return } b & \quad \texttt{return } b
\end{array}
$$

Now define the *advantage* of A in violating the hash Diffie-Hellman assumption as

$$\mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\mathrm{H},A} \quad = \quad \Pr[\,\mathbf{Exp}^{\text{hdh-real}}_{\mathcal{G},\mathrm{H},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{hdh-rand}}_{\mathcal{G},\mathrm{H},A} = 1\,]\,. \qquad \diamondsuit$$

The decisional Diffie-Hellman assumption says that g^{uv} looks like a random group element, even if you know g^u and g^v. The hash Diffie-Hellman assumption says that $H(g^{uv})$ looks like a random string, even if you know g^u and g^v. So if you set H to be the identity function you almost recover the decisional Diffie-Hellman assumption (the difference being that in one case you get a random group element and in the other you get a random string). When H is a cryptographic hash function, like SHA-1 [32], the hash Diffie-Hellman assumption would seem to be a much weaker assumption than the decisional Diffie-Hellman assumption.

We now move on to some more novel assumptions.

ORACLE DIFFIE-HELLMAN ASSUMPTION: ODH. Suppose we provide an adversary A with g^v and an oracle $\mathcal{H}_v$, which computes the function $\mathcal{H}_v(X) = X^v$. Think of $v \in \{1, \ldots, |G|\}$ as having been chosen at random. Now if we give the adversary g^u (where $u \in \{1, \ldots, |G|\}$ is chosen at random) then the oracle will certainly enable the adversary to compute g^{uv}: the adversary need only ask the query g^u and she gets back $\mathcal{H}_v(g^u) = g^{uv}$. Even if we *f*orbid the adversary from asking g^u, still she can exploit the self-reducibility of the discrete log to find the value of g^{uv}. For example, the adversary could compute $\mathcal{H}_v(gg^u) = g^{uv}g^v$ and divide this by $\mathcal{H}_v(1) = g^v$.

But what if instead we give the adversary an oracle $\mathcal{H}_v$ which computes $\mathcal{H}_v(X) = H(X^v)$, for H a cryptographic hash function such as SHA-1? Suppose the adversary's goal is to compute $H(g^{uv})$, where g^u and g^v are provided to the adversary. Now, as long as the oracle $\mathcal{H}_v$ can not be queried at g^u, the oracle would seem to be useless. We formalize this as follows.

Definition 8 [Oracle Diffie-Hellman: ODH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let $hLen$ be a number, let $H : \{0,1\}^* \to \{0,1\}^{hLen}$, and let A be an adversary. Consider the experiments

experiment $\mathbf{Exp}^{\text{odh-real}}_{\mathcal{G},\text{H},A}$	experiment $\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\text{H},A}$
$u \xleftarrow{R} \{1, \ldots, \|G\|\};\ U \leftarrow g^u$	$u \xleftarrow{R} \{1, \ldots, \|G\|\};\ U \leftarrow g^u$
$v \xleftarrow{R} \{1, \ldots, \|G\|\};\ V \leftarrow g^v$	$v \xleftarrow{R} \{1, \ldots, \|G\|\};\ V \leftarrow g^v$
$W \leftarrow H(g^{uv})$	$W \xleftarrow{R} \{0,1\}^{hLen}$
$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$	$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$
$b \leftarrow A^{\mathcal{H}_v(\cdot)}(U, V, W)$	$b \leftarrow A^{\mathcal{H}_v(\cdot)}(U, V, W)$
return b	return b

Now define the *advantage* of A in violating the oracle Diffie-Hellman assumption as

$$\mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H},A} \quad = \quad \Pr[\,\mathbf{Exp}^{\text{odh-real}}_{\mathcal{G},\text{H},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\text{H},A} = 1\,]\;.$$

Here A is not allowed to call its oracle on g^u. $\diamondsuit$

We emphasize that the adversary is allowed to make oracle queries that depend on the target g^u, with the sole restriction of not being allowed to query g^u itself.

STRONG DIFFIE-HELLMAN ASSUMPTION: SDH. Suppose A is an algorithm which, given g^v, outputs a pair of strings (g^u, g^{uv}), for some $u \in \{1, \ldots, |G|\}$. One way for A to find such a pair is to pick some value u and then compute g^u and g^{uv}. Indeed, we expect this to be the "only" way A can compute such a pair of values. We capture this idea as follows.

Given a represented group $\mathcal{G} = (G, g, \_, \uparrow)$ and a number v, let $\mathcal{O}_v$ be an oracle, called a *restricted DDH oracle*, which behaves as follows:

$$\mathcal{O}_v(U, X) \quad = \quad \begin{cases} 1 & \text{if } X = U^v \\ 0 & \text{otherwise} \end{cases}$$

That is, the oracle tells whether the second argument equals the first argument raised to v-th power. This oracle can be seen as a restricted form of a DDH oracle for which we fix one of its arguments as being g^v. Our next definition speaks to the uselessness of having a restricted DDH oracle.

Definition 9 [Strong Diffie-Hellman: SDH] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let A be an adversary. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}^{\mathrm{sdh}}_{\mathcal{G},A} \\
\quad u \stackrel{R}{\leftarrow} \{1, \ldots, |G|\};\ U \leftarrow g^u \\
\quad v \stackrel{R}{\leftarrow} \{1, \ldots, |G|\};\ V \leftarrow g^v \\
\quad \mathcal{O}_v(U, X) \stackrel{\mathrm{def}}{=} (X = U^v) \\
\quad Z \leftarrow A^{\mathcal{O}_v(\cdot,\cdot)}(U, V) \\
\quad \texttt{if } Z = g^{uv} \texttt{ then } b \leftarrow 1 \texttt{ else } b \leftarrow 0 \\
\quad \texttt{return } b
\end{array}
$$

Now define the *advantage* of A in violating the strong Diffie-Hellman assumption as

$$\mathbf{Adv}^{\mathrm{sdh}}_{\mathcal{G},A} \;=\; \Pr[\,\mathbf{Exp}^{\mathrm{sdh}}_{\mathcal{G},A} = 1\,]\,. \qquad \diamondsuit$$

The intuition is that the restricted DDH oracle is useless because the adversary already "knows" the answer to almost any query it will ask.

Similar intuition was captured in [22] by saying that for every non-uniform probabilistic polynomial-time algorithm A that, on input g^v, outputs (g^u, g^{uv}), there exists a non-uniform probabilistic polynomial-time algorithm S (the "extractor") that not only outputs (g^u, g^{uv}), but also u. Our approach avoids the complexity of a simulator-based formulation. We emphasize that our oracle does not return a value u (the discrete log of its first argument) but only a bit indicating whether a given pair has the right form.

RESOURCE MEASURES. We have defined several different senses of adversarial advantage. For each notion xxx we overload the notation and define

$$\mathbf{Adv}^{\mathrm{xxx}}_{\Pi}(R) \;=\; \max_{A}\{\,\mathbf{Adv}^{\mathrm{xxx}}_{\Pi,A}\,\}$$

where R is a resource measure and the maximum is taken over all adversaries that use resources at most R. The resources of interest in this paper are time-complexity (denoted by t) and, when appropriate, number of queries (denoted by q). Any other resources of importance will be mentioned when the corresponding notion is described. Here and throughout this paper "time-complexity" is understood to mean the maximum of the execution times of the experiments defining the advantage of adversary A plus the size of the code for A, all in some fixed RAM model of computation. (Note that the execution time refers to that of the entire experiment, not just the execution time of the adversary.)

We comment that we are considering the complexity of adversaries who try to attack a specific represented group $\mathcal{G}$. Such an adversary may depend on $\mathcal{G}$, so explicitly providing a description of $\mathcal{G}$ to A is unnecessary.

CHOICE OF HASH FUNCTION. Now that we understand how we want the hash function to interact with the group, we can consider various choices for the hash function H.

Our suggested choice is to appropriately derive H from some cryptographic hash function like SHA-1 [32]. (The precise manner in which H is derived from SHA-1 is important and should be discussed.) A primary reason we prefer a cryptographic function is that one-wayness of H appears important to the oracle Diffie-Hellman assumption: it should be hard to recover g^{uv} from $H(g^{uv})$, since otherwise the self-reducibility-based attack we discussed above can be mounted.

Let us back up a bit and try to see what requirements the different assumptions impose on the choice of H. Suppose first we are interested only in semantic security, namely we need just the HDH assumption. There is no known choice of H for which one can prove the hard-coreness under the CDH assumption. Under the DDH assumption, however, things get much easier, since this assumption already says that the Diffie-Hellman key is indistinguishable from a random group element: the only remaining problem is to go from a random group element to a random string of appropriate length. In some groups this can be done quite easily by simple truncation of the key. Alternatively, Naor and Reingold [27] show that application of a function H chosen at random from a family of universal hash functions will suffice. Zheng and Seberry [37] had earlier suggested the application of a universal hash function to the Diffie-Hellman key as a heuristic under the computational Diffie-Hellman assumption. The result of [27] says that under the stronger DDH assumption this heuristic is valid. Note this function can be chosen at random once and for all and included in the public key. In [37], the function is chosen anew for each encryption and included in the ciphertext, which increases the size of the ciphertext.

However, the use of truncation or universal hash functions appears more dangerous when we come to consider the stronger oracle Diffie-Hellman assumption above. In particular, the result of Boneh and Venkatesan [12] showing that computing the most significant bits of Diffie-Hellman keys is as hard as computing the key itself can be turned on its head to give an algorithm to attack the ODH assumption. Namely, their results show that for some simple choices of functions H, an adversary can use the HDH oracle $\mathcal{H}_v$ defined above to solve the Diffie-Hellman problem. These attacks do not appear to work when a one-way cryptographic hash function is used, which is why we recommend this choice. We do not know whether these attacks rule out all choices of universal hash families, but they do seem to rule out some particular ones.

6 Security against Chosen-Plaintext Attack

We show that $\mathrm{DHIES}\,[\![\mathcal{G}, \mathrm{SYM}, \mathrm{MAC}, \mathrm{H}]\!]$ meets the notion of indistinguishability under a chosen-plaintext attack, as defined in Definition 3.

Theorem 1 Let $\mathcal{G}$ be a represented group, let SYM be a symmetric encryption scheme, let MAC be a message authentication scheme, and let H be a function. Let DHIES be the asymmetric key encryption scheme associated to these primitives, as defined in Section 3. Then, for any numbers t and c,

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathrm{DHIES}}(t,c) \;\le\; 2\cdot\mathbf{Adv}^{\mathrm{hdh}}_{\mathcal{G},H}(t) + \mathbf{Adv}^{\text{ind-cpa-fg}}_{\mathrm{SYM}}(t,0,0)\ .$$

IDEA OF PROOF. The assumption is that the symmetric encryption scheme SYM is secure and H is hard-core for the Diffie-Hellman problem in the underlying group. (The assumption that MAC is secure is not needed to ensure semantic security.) The proof considers an adversary A who defeats the semantic security of the scheme. Let g^v be the recipient public key and let $y = U \parallel encM \parallel tag$ be the challenge ciphertext that this adversary gets in its guess stage. We consider two cases depending on whether the output of H "looks random".

- *Case 1 — The output of H looks random.* In this case, we present an adversary B that breaks the encryption scheme SYM.
- *Case 2 — The output of H does not look random.* In this case, we present an algorithm C that breaks the hard-coreness of H on $\mathcal{G}$.

The formal proof, given below, does not actually consider separate cases, but the underlying intuition is the same. Given A, we construct B and C and then relate A's advantage to that of B and C.

Proof: Let A be an adversary attacking DHIES in the sense of semantic security. Assume it has time-complexity at most t. We construct an adversary B attacking SYM and an adversary C attacking H being hard-core for $\mathcal{G}$, and then upper bound the advantage of A in terms of the advantages of these adversaries.

ALGORITHM B. Figure 3 describes algorithm B. Recall from Definition 2 that B has access to an oracle for encryption, and runs in two stages. Notice that B never invokes its encryption oracle $\mathcal{E}$. Moreover, the running time of $\mathbf{Exp}^{\text{ind-cpa-fg}}_{\mathrm{SYM},B}$ is at most t.

```
algorithm B^{E(·)}(find)
begin
    v ←R {1, ..., |G|}
    pk ← g^v
    (x_0, x_1, s) ← A(find, pk)
    s~ ← (x_0, x_1, s, pk)
    return (x_0, x_1, s~)
end
```

```
algorithm B^{E(·)}(guess, y~, s~)
begin
    parse s~ as (x_0, x_1, s, pk)
    u ←R {1, ..., |G|}; U ← g^u
    macKey ←R {0,1}^{mLen}
    tag ← T_{macKey}(y~)
    y ← U || y~ || tag
    b ← A(guess, pk, s, y)
    return b
end
```

Figure 3: *Algorithm B for attacking the security of* SYM.

```
algorithm C(U, V, W)
begin
    macKey ← W[1 ... mLen];   encKey ← W[mLen + 1 ... mLen + eLen]
    pk ← V
    (x_0, x_1, s) ← A(find, pk)
    b~ ← {0,1};   encM ← E_{encKey}(x_{b~})
    tag ← T_{macKey}(encM)
    y ← U || encM || tag
    b ← A(guess, pk, s, y)
    if b = b~ then return 1 else return 0
end
```

Figure 4: *Algorithm C for attacking the hard-coreness of H on $\mathcal{G}$.*

ALGORITHM C. Figure 4 depicts the behavior of algorithm C. C is given as input U, V, W, where $U = g^u$ and $V = g^v$ for random u, v, and W is either $H(g^{uv})$ or a random string. C outputs at the end a bit indicating its guess as to which of these cases occurs. Notice that the time-complexity of C is at most t.

ANALYSIS. When $W = H(g^{uv})$ we notice that C is running A as the latter would be run in its attack on the semantic security of DHIES. From the definition of $\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}$, we have that

$$\Pr[\,\mathbf{Exp}^{\text{hdh-real}}_{\mathcal{G},\text{H},C} = 1\,] \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}}{2}\;.$$

On the other hand, when W is a random string, we notice that C runs A in the same way as B does, and hence

$$\Pr[\,\mathbf{Exp}^{\text{hdh-rand}}_{\mathcal{G},\text{H},C} = 1\,] \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},B}}{2}\;.$$

Subtracting gives us

$$\mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\text{H},C} \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}}{2} - \frac{1}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},B}}{2} \;=\; \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A}}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},B}}{2}\;;$$

whence

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A} \;=\; 2\cdot \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\text{H},C} + \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},B} \;.$$

Since the time-complexity of C is at most t, we conclude that $\mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\text{H},C} \le \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\text{H}}(t)$. Moreover, since B makes 0 encryption queries and has time-complexity at most t, we also have $\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},B} \le \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0)$. Thus from the above we have

$$\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{DHIES},A} \;\le\; 2\cdot \mathbf{Adv}^{\text{hdh}}_{\mathcal{G},\text{H}}(t) + \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0) \;.$$

But A was an arbitrary adversary subject to the constraint that it had time-complexity at most t and the length of its challenge ciphertext is at most c. The theorem follows. ▮ ▮

7 Security against Chosen-Ciphertext Attack

We show that DHIES $[\![\mathcal{G}, \text{SYM}, \text{MAC}, \text{H}]\!]$ meets the notion of indistinguishability under an adaptive chosen-ciphertext attack, as in Definition 4.

Theorem 2 Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let SYM be a symmetric encryption scheme, and let MAC be a message authentication scheme. Let DHIES be the asymmetric encryption scheme associated to these primitives as defined in Section 3. Then for any numbers t, q, μ, and c,

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES}}(t,q,\mu,c) \quad &\le \quad \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0) + 2\cdot \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H}}(t,q) + \\ &\qquad 2\cdot \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t,1,c,q,\mu) \;.\end{aligned}$$

IDEA OF PROOF. The assumption is that both symmetric encryption scheme SYM and the message authentication scheme MAC are secure and H is a hard-core for the Diffie-Hellman problem on $\mathcal{G}$ under adaptive Diffie-Hellman attack. The proof considers an adversary A who defeats the adaptive chosen-ciphertext security of the scheme. Let g^v be the recipient public key; let $y = U \,\|\, encM \,\|\, tag$ be the challenge ciphertext that algorithm A gets in its guess stage. Let us call a Type 1 query a ciphertext of the form $U \,\|\, \widetilde{encM} \,\|\, \widetilde{tag}$. A Type 2 query have the form $\widetilde{U} \,\|\, \widetilde{encM} \,\|\, \widetilde{tag}$ with $\widetilde{U} \neq U$. We consider three cases depending on whether the output of H looks random and on whether there was a Type 1 query $\widetilde{y}$ to the decryption oracle $\overline{\mathcal{D}}_{sk}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$.

- *Case 1 — The output of H does not look random.* In this case we present an algorithm C that breaks the hard-coreness of H on $\mathcal{G}$ under adaptive Diffie-Hellman attack.
- *Case 2 — The output of H looks random and there was a* Type 1 *query $\widetilde{y}$ to $\overline{\mathcal{D}}_{sk}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq$* BAD. In this case we present an adversary F which breaks the message authentication scheme MAC.
- *Case 3 — The output of H looks random and there was not a* Type 1 *query $\widetilde{y}$ to $\overline{\mathcal{D}}_{sk}$ such that* $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$. In this case we present an adversary B which breaks the encryption scheme SYM.

Proof: Let A be an adversary attacking DHIES in the sense of adaptive chosen-ciphertext security. Assume it has running time at most t, makes at most q queries to its decryption oracle. We construct an adversary B attacking SYM, an adversary C attacking H being a hard-core for $\mathcal{G}$ under non-adaptive Diffie-Hellman attack, and an adversary F for the message authentication scheme MAC and then upper bound the advantage of A in terms of the advantages of these adversaries.

`algorithm` $B^{\mathcal{E}(\cdot)}(\mathsf{find})$
`begin`
$\quad v \stackrel{R}{\leftarrow} \{1,\ldots,\lvert G\rvert\}$
$\quad pk \leftarrow g^v$
$\quad$`run` $A(\mathsf{find}, pk)$
$\quad$– For each decryption query y_i
$\qquad$`parse` y_i `as` $U_i \parallel encM_i \parallel tag_i$
$\qquad hash_i \leftarrow H(U_i^v)$
$\qquad macKey_i \leftarrow hash_i[1..mLen]$
$\qquad encKey_i \leftarrow hash_i[mLen+1..mLen+eLen]$
$\qquad$`if` $\mathcal{V}_{macKey_i}(encM_i, tag_i) = 1$ `then`
$\qquad\quad$`return` $\mathcal{D}_{encKey_i}(encM_i)$
$\qquad$`else return` BAD
$\quad$– Let (x_0, x_1, s) be the output of A
$\quad \widetilde{s} \leftarrow (x_0, x_1, s, v, pk)$
$\quad$`return` $(x_0, x_1, \widetilde{s})$
`end`
	`algorithm` $B^{\mathcal{E}(\cdot)}(\mathsf{guess}, \widetilde{y}, \widetilde{s})$
	`begin`
	$\quad$`parse` $\widetilde{s}$ `as` (x_0, x_1, s, v, pk)
	$\quad$ASK $\leftarrow$ false
	$\quad u \stackrel{R}{\leftarrow} \{1,\ldots,\lvert G\rvert\}$
	$\quad U \leftarrow g^u$
	$\quad macKey \stackrel{R}{\leftarrow} \{0,1\}^{mLen}$
	$\quad tag \leftarrow \mathcal{T}_{macKey}(\widetilde{y})$
	$\quad y \leftarrow U \parallel \widetilde{y} \parallel tag$
	$\quad$`run` $A(\mathsf{guess}, pk, s, y)$
	$\quad$– For each decryption query y_i
	$\qquad$`parse` y_i `as` $U_i \parallel encM_i \parallel tag_i$
	$\qquad hash_i \leftarrow H(U_i^v)$
	$\qquad macKey_i \leftarrow hash_i[1..mLen]$
	$\qquad encKey_i \leftarrow hash_i[mLen+1..mLen+eLen]$
	$\qquad$`if` $\mathcal{V}_{macKey_i}(encM_i, tag_i) = 1$ `then`
	$\qquad\quad$`if` $U_i \neq U$ `then`
	$\qquad\qquad$`return` $\mathcal{D}_{encKey_i}(encM_i)$
	$\qquad\quad$`else` ASK $\leftarrow$ true;
	$\qquad$`return` BAD
	$\quad$– `if` ASK = true `then` $b \stackrel{R}{\leftarrow} \{0,1\}$
	$\qquad$`else` let b be the output of A
	$\quad$`return` b
	`end`

Figure 5: *Algorithm B for attacking the security of* SYM.

ALGORITHM B. Figure 5 describes algorithm B. Recall from Definition 2 that B has access to an oracle for encryption and runs in two stages. Since the time-complexity t of A accounts for the time taken by decryption queries as well as the time to generate pk and the challenge ciphertext y, the time-complexity of B is at most that of A (i.e., t).

ALGORITHM C. Figure 6 defines the behavior of algorithm C. C is given as input U, V, W, where $U = g^u$ and $V = g^v$ for random u and v, respectively, and W is either $H(g^{uv})$ or a random string. Recall from Definition 8 that C is also given access to a $\mathcal{H}_v$-oracle. At the end, C outputs a bit indicating its guess as to which of these cases occurs.

Notice that, since the time-complexity of A accounts for the time taken by decryption queries as well as the time to compute the challenge ciphertext, the time-complexity of C is at most t.

ALGORITHM F. Figure 7 describes algorithm F. Recall from Definition 1 that F has access to two oracles: a tag-generation oracle $\mathcal{T}$ and a verification oracle $\mathcal{V}$. It outputs a pair message-tag, a possible forgery. Notice that, since the time complexity of A accounts for the time taken by decryption queries and for the time to generate the secret-public key pair $sk, pk)$ and the challenge ciphertext y, F's time-complexity is at most t.

ANALYSIS. As in the proof of non-adaptive chosen-ciphertext security, our goal is to upper bound the success probability of adversary A in terms of the success probabilities of adversaries B for the symmetric encryption scheme, C for the hard-coreness of H for $\mathcal{G}$ under non-adaptive Diffie-Hellman attack, and F

algorithm $C^{\mathcal{H}_v(\cdot)}(U, V, W)$
begin
  $macKey \leftarrow W[1 \ldots mLen]$
  $encKey \leftarrow W[mLen + 1 \ldots mLen + eLen]$
  $pk \leftarrow V$
  run $A(\mathsf{find}, pk)$
  – For each decryption query y_i
    return **Decr-Simulator**(y_i, U, V, W)
  – Let (x_0, x_1, s) be the output of A
  $\widetilde{b} \leftarrow \{0, 1\}$
  $encM \leftarrow \mathcal{E}_{encKey}(x_{\widetilde{b}})$
  $tag \leftarrow \mathcal{T}_{macKey}(encM)$
  $y \leftarrow U \parallel encM \parallel tag$
  run $A(\mathsf{guess}, pk, s, y)$
  – For each decryption query y_i
    return **Decr-Simulator**(y_i, U, V, W)
  – Let b be the output of A
  if $b = \widetilde{b}$ then return 1 else return 0
end

subroutine **Decr-Simulator**(y_i, U, V, W)
begin
  parse y_i as $U_i \parallel encM_i \parallel tag_i$
  if $U_i = U$ then
    $macKey_i \leftarrow W[1 \ldots mLen]$
    $encKey_i \leftarrow W[mLen + 1 \ldots mLen + eLen]$
  else
    $hash_i \leftarrow \mathcal{H}_v(U_i)$
    $macKey_i \leftarrow hash_i[1..mLen]$
    $encKey_i \leftarrow hash_i[mLen + 1..mLen + eLen]$
  if $\mathcal{V}_{macKey_i}(encM_i, tag_i) = 1$ then
    return $\mathcal{D}_{encKey_i}(encM_i)$
  else return BAD
end

Figure 6: *Algorithm C for attacking the hard-coreness of H on $\mathcal{G}$ under adaptive Diffie-Hellman attack.*

algorithm $F^{\mathcal{T}(\cdot), \mathcal{V}(\cdot,\cdot)}$
begin
  $W \leftarrow (\epsilon, \epsilon)$
  $v \stackrel{R}{\leftarrow} \{1, \ldots, |G|\};\ pk \leftarrow g^v$
  $u \stackrel{R}{\leftarrow} \{1, \ldots, |G|\};\ U \leftarrow g^u$
  $encKey \stackrel{R}{\leftarrow} \{0, 1\}^{eLen}$
  run $A(\mathsf{find}, pk)$
  – For each decryption query $\widetilde{y}$
    return **Decr-Simulator**$(\widetilde{y})$
  – Let (x_0, x_1, s) be the output of A
  $\widetilde{b} \stackrel{R}{\leftarrow} \{0, 1\}$
  $encM \leftarrow \mathcal{E}_{encKey}(x_{\widetilde{b}})$
  $tag \leftarrow \mathcal{T}(encM)$
  $y \leftarrow U \parallel encM \parallel tag$
  run $A(\mathsf{guess}, pk, s, y)$
  – For each decryption query $\widetilde{y}$
    return **Decr-Simulator**$(\widetilde{y})$
  – Let b be the output of A
  return W
end

subroutine **Decr-Simulator**$(\widetilde{y})$
begin
  parse $\widetilde{y}$ as $\widetilde{U} \parallel \widetilde{encM} \parallel \widetilde{tag}$
  $\widetilde{hash} \leftarrow H(\widetilde{U}^v)$
  $\widetilde{macKey} \leftarrow \widetilde{hash}[1..mLen]$
  $\widetilde{encKey} \leftarrow \widetilde{hash}[mLen + 1..mLen + eLen]$
  if $\widetilde{U} = U$ then
    if $\mathcal{V}(\widetilde{encM}, \widetilde{tag}) = 1$ then
      $W \leftarrow (\widetilde{encM}, \widetilde{tag})$
      return $\mathcal{D}_{encKey}(\widetilde{encM})$
    else return BAD
  else
    if $\mathcal{V}_{\widetilde{macKey}}(\widetilde{encM}, \widetilde{tag}) = 1$ then
      return $\mathcal{D}_{\widetilde{encKey}}(\widetilde{encM})$
    else return BAD
end

Figure 7: *Algorithm F for attacking the security of* MAC.

for the message authentication scheme. For this purpose, let y be the challenge ciphertext in experiment

$\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{DHIES},A}$ and let SOMEVALID be the event where A makes a Type 1 query $\widetilde{y}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$ in this experiment. Let $\overline{\text{SOMEVALID}}$ denote the event where there is no Type 1 query $\widetilde{y}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$ in experiment $\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{DHIES},A}$. We make use of the following three claims.

Claim 3 $\Pr[\,\mathbf{Exp}^{\text{odh-real}}_{\mathcal{G},\text{H},C} = 1\,] \;=\; \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}}{2}$.

Proof: When the input $W = H(g^{uv})$, we notice that C is running A as the latter would be run in its attack on the adaptive chosen-ciphertext security of DHIES. Therefore, the claim follows from the definition of $\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}$. ▮

Claim 4 $\Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\text{H},C} = 1 \wedge \overline{\text{SOMEVALID}}\,] \leq \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0)}{2}$

Proof: When A does not make a Type 1 query to its decryption oracle nor makes a Type 1 query $\widetilde{y}$ such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$, C runs A in the same way B does. Hence, the probability that C outputs 1 given $\overline{\text{SOMEVALID}}$ is at most $1/2 + 1/2 \cdot \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM},B}$. Since B makes 0 encryption queries and has time-complexity at most t, the claim follows directly from the assumed security of SYM. ▮

Claim 5 $\Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\text{H},C} = 1 \wedge \text{SOMEVALID}\,] \leq \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t,1,c,q,\mu)$

Proof: When there is a Type 1 query $\widetilde{y}$ to the decryption oracle such that $\overline{\mathcal{D}}_{sk}(\widetilde{y}) \neq \text{BAD}$, let i be the number of one such query and let $y_i = U \,\|\, encM_i \,\|\, tag_i$ be its value. By assumption, we know that $\mathcal{V}_{macKey}(encM_i, tag_i) = 1$. Because $(encM_i, tag_i) \neq (encM, tag)$ (or otherwise $y_i = y$ and A would have queried its decryption oracle with the challenge ciphertext), either $encM_i$ was not queried of tag-generation oracle $\mathcal{T}(\cdot)$ or tag_i was not the response returned by tag-generation oracle $\mathcal{T}(\cdot)$ on query $encM$. In either case, $(encM_i, tag_i)$ is a valid forgery and F succeeds in breaking MAC. Therefore, $\Pr[\,\mathbf{Exp}^{\text{odh-rand}}_{\mathcal{G},\text{H},C} = 1 \wedge \text{SOMEVALID}\,] \;\leq\; \Pr[\,\text{SOMEVALID}\,] \;\leq\; \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC},F}$. Hence, since F has time-complexity at most t, makes exactly one query to its tag-generation oracle $\mathcal{T}(\cdot)$ whose length is at most c (the upper bound on the length of challenge ciphertext), and makes at most q queries to its verification oracle $\mathcal{V}(\cdot,\cdot)$ the sum of whose lengths is at most μ bits, the claim follows from the assumed security of MAC. ▮

From Definition 8 and Claims 3, 4, and 5, we have that:

$$\begin{aligned}
\mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H},C} \quad &\geq \quad \frac{1}{2} + \frac{\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}}{2} - \frac{1}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0)}{2} - \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t,1,c,q,\mu) \\
&= \quad \frac{\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A}}{2} - \frac{\mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0)}{2} - \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t,1,c,q,\mu) \ ;
\end{aligned}$$

whence

$$\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A} \leq \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0) + 2 \cdot \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H},C} + 2 \cdot \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t,1,c,q,\mu) \ .$$

We conclude that, since C has time-complexity at most t and makes at most q queries to its oracle $\mathcal{H}_v$, $\mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H},C} \leq \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H}}(t,q)$. Thus, from the above, we have

$$\mathbf{Adv}^{\text{ind-cca-fg}}_{\text{DHIES},A} \;\leq\; \mathbf{Adv}^{\text{ind-cpa-fg}}_{\text{SYM}}(t,0,0) + 2 \cdot \mathbf{Adv}^{\text{odh}}_{\mathcal{G},\text{H}}(t,q) + 2 \cdot \mathbf{Adv}^{\text{suf-cma}}_{\text{MAC}}(t,1,c,q,\mu) \ .$$

But A was an arbitrary adversary subject to the constraint that it has time-complexity at most t and makes at most q queries to its decryption oracle the sum of whose lengths is at most μ bits, and the length of the challenge ciphertext is at most c in experiment $\mathbf{Exp}^{\text{ind-cca-fg}}_{\text{DHIES},A}$. The theorem follows. ▌▌

8 ODH and SDH

In this section, we first exploit the relationship between the strong Diffie-Hellman (SDH) and the oracle Diffie-Hellman (ODH) assumptions when the hash function H is modeled as a random oracle. More specifically, we show that, in the random oracle (RO) model, the SDH assumption implies the ODH assumption. We then go on to prove a lower bound on the complexity of strong Diffie-Hellman assumption with respect to generic algorithms.

However, before proving the implication between the SDH and ODH assumption in the RO model, we need to back up a little and modify the experiment defining the security of ODH assumption to account for the presence of a random oracle H. The following is the definition of ODH assumption in the the RO model.

Definition 10 [Oracle Diffie-Hellman in the random oracle model: ODH-RO] Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let $hLen$ be a number, let Ω be the set of all functions from $\{0,1\}^*$ to $\{0,1\}^{hLen}$, and let A be an adversary. Consider the experiments

experiment $\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},\text{H},A}$	experiment $\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},\text{H},A}$
$u \stackrel{R}{\leftarrow} \{1,\ldots,\lvert G\rvert\};\; U \leftarrow g^u$	$u \stackrel{R}{\leftarrow} \{1,\ldots,\lvert G\rvert\};\; U \leftarrow g^u$
$v \stackrel{R}{\leftarrow} \{1,\ldots,\lvert G\rvert\};\; V \leftarrow g^v$	$v \stackrel{R}{\leftarrow} \{1,\ldots,\lvert G\rvert\};\; V \leftarrow g^v$
$W \leftarrow H(g^{uv})$	$W \stackrel{R}{\leftarrow} \{0,1\}^{hLen}$
$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$	$\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$
$H \stackrel{R}{\leftarrow} \Omega$	$H \stackrel{R}{\leftarrow} \Omega$
$b \leftarrow A^{\mathcal{H}_v(\cdot),H(\cdot)}(U,V,W)$	$b \leftarrow A^{\mathcal{H}_v(\cdot),H(\cdot)}(U,V,W)$
return b	return b

Now define the *advantage* of A in violating the oracle Diffie-Hellman assumption in the random oracle model and the *advantage function* of ODH assumption in the RO model, respectively, as follows:

$$\begin{aligned}\mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\text{H},A} &= \Pr[\,\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},\text{H},A} = 1\,] - \Pr[\,\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},\text{H},A} = 1\,] \\ \mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\text{H}}(t, q_h, q_o) &= \max_A\{\,\mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\text{H},A}\,\}\ ,\end{aligned}$$

where the maximum is over all A with time-complexity t, making at most q_h queries to its H-oracle and q_o queries to its $\mathcal{H}_v$-oracle. Here A is not allowed to call its oracle $\mathcal{H}_v$ on input g^u. $\diamondsuit$

The following theorem shows that, in the random oracle (RO) model, the strong Diffie-Hellman assumption implies the oracle Diffie-Hellman assumption.

Theorem 6 Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let the associated hash function H be chosen at random. Let q_h and q_o be, respectively, the total number of queries to H-oracle and to the $\mathcal{H}_v$-oracle. Then,

$$\mathbf{Adv}^{\text{odh-ro}}_{\mathcal{G},\text{H}}(t, q_h, q_o) \;\leq\; \mathbf{Adv}^{\text{sdh}}_{\mathcal{G}}(t + q_h\, q_o\, O(gLen + hLen), (q_h + q_o)^2)\ .$$

Proof: Let A be any adversary against the ODH assumption in the RO model. Let t be its time-complexity and q_h and q_o be, respectively, the number of queries it makes to H and $\mathcal{H}_v$ oracles. We can construct an adversary B for the Diffie-Hellman problem under SDH on $\mathcal{G}$, using A as a sub-routine, as follows.

```
algorithm B^{O_v(·)}(U,V)
begin
    hash_0 ←R {0,1}^{hLen};   hash_1 ←R {0,1}^{hLen}
    b ←R {0,1};   W ← hash_b
    guess ← g
    run A^{H_v(·),H}(U,V,W)
    –  For each H_v-query Ũ, return response as described in the text
    –  For each H-query X, return response as described in the text
       (updating guess if O_v(U,X) = 1)
    –  Let b̃ be the output of A
    return guess
end
```

Figure 8: *Algorithm B for attacking the hard-coreness of the Diffie-Hellman problem under SDH on $\mathcal{G}$.*

ALGORITHM B. Algorithm B is shown in Figure 8. B is given as input (U,V), where $U = g^u$ and $V = g^v$ for random u and v and outputs a value *guess*, its guess for g^{uv}. B is also given access to a restricted DDH oracle $\mathcal{O}_v$. Initially, B picks two values, $hash_0$ and $hash_1$, at random and set $hash_0$ to be the output of H on input U^v (although it still does not know the value of U^v). It also fix a default value g for *guess*. Then it runs A as a subroutine, feeding its input with either $hash_0$ or $hash_1$. Our hope is that, at some point, A is going to make a query of the form U^v to the H oracle, since otherwise A would have no advantage in distinguishing the real output $hash_0 \stackrel{\text{def}}{=} H(U^v)$ from the random string $hash_1$. To find out when A queries its H-oracle on input U^v, we query our restricted DDH oracle $\mathcal{O}_v$ on input (U,X) whenever A makes a query X to its H-oracle. If $\mathcal{O}_v(U,X)$ returns 1, then $X = U^v$ and we update the value of *guess* to X. Notice that B does not have access to either the H-oracle or the $\mathcal{H}_v$-oracle. However, it can simulate both oracles.

The oracle H is to be simulated as follows. In response to a query X, if X has been asked of H, then return the same response as given to that previous query. If X has not been asked of H, then first check whether $\mathcal{O}_v(U,X) = 1$. If so, then update the value of *guess* to X and return $hash_0$ as the response to the current H-query. If not, then check whether there was some query $\widetilde{U}$ to the $\mathcal{H}_v$-oracle such that $\mathcal{O}_v(\widetilde{U},X) = 1$. If there was such query, then let $hash$ be the same value used as response to that query. If not, then let $hash$ be a random string. Return $hash$ as the response to the current H query.

The $\mathcal{H}_v$-oracle is to be simulated as follows. When a query $\widetilde{U} \neq U$ is made, check whether $\widetilde{U}$ has already been asked of $\mathcal{H}_v$. If so, then return the same response as given to that previous query. If not, then check whether there was a previous H-query X where $\mathcal{O}_v(\widetilde{U},X) = 1$. If so, then let $hash$ be the output given to that H-query. If not, then let $hash$ be a random value. Return $hash$ as the response to the current query.

Notice that B runs in time at most $t + q_o q_h O(hLen + gLen)$ and makes at most $(q_o + q_h)^2$ queries to its $\mathcal{O}_v$-oracle.

ANALYSIS. Consider the experiments $\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A}$ and $\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A}$ and let ASKA and $\overline{\text{ASKA}}$ denote, respectively, the event in which H-oracle query U^v is made by A and its complement.

When query U^v is not made directly by A to its H-oracle, there is no way for it to tell whether its input W is equal to $H(U^v)$ or a random string of length $hLen$ since the former can take any value in $\{0,1\}^{hLen}$. Hence, the probabilities that A outputs 1 in experiments $\mathbf{Exp}^{\text{odh-rand-ro}}_{\mathcal{G},A}$ and $\mathbf{Exp}^{\text{odh-real-ro}}_{\mathcal{G},A}$, given that A

does not query its H-oracle on input U^v are exactly the same. That is, $\Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}} = 1 \wedge \overline{\textsc{AskA}}\,] = \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}} = 1 \wedge \overline{\textsc{AskA}}\,]$.

Consider now the case in which adversary A queries its H-oracle on input U^v in experiments $\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}}$ and $\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}}$. In such case, we know that adversary B, which runs A as a sub-routine, succeeds in solving the Diffie-Hellman problem under SDH on $\mathcal{G}$. This is because we know there is an index $i \in \{1,\ldots,q_h\}$ such that $h_i = X$ and $X = U^v$. Since $\mathcal{O}_v(U,X) = 1$, *guess* will take the correct value U^v and B will succeed in solving the strong Diffie-Hellman problem on $\mathcal{G}$. Hence, $\Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}} = 1 \wedge \textsc{AskA}\,] - \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}} = 1 \wedge \textsc{AskA}\,] \leq \Pr[\,\textsc{AskA}\,] \leq \mathbf{Adv}_{\mathcal{G},B}^{\text{sdh}}$. Moreover, since B makes at most $(q_o+q_h)^2$ queries to its oracle $\mathcal{O}_v$ and has time complexity at most $t + q_o q_h O(hLen + gLen)$, it is also the case that $\mathbf{Adv}_{\mathcal{G},B}^{\text{sdh}} \leq \mathbf{Adv}_{\mathcal{G}}^{\text{sdh}}(t,(q_o+q_h)^2)$.

Putting it all together, we have that

$$
\begin{aligned}
\mathbf{Adv}_{\mathcal{G},H,A}^{\text{odh}} &= \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}} = 1\,] - \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}} = 1\,] \\
&= \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}} = 1 \wedge \textsc{AskA}\,] + \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}} = 1 \wedge \overline{\textsc{AskA}}\,] \\
&\quad -\Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}} = 1 \wedge \textsc{AskA}\,] - \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}} = 1 \wedge \overline{\textsc{AskA}}\,] \\
&= \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-real-ro}} = 1 \wedge \textsc{AskA}\,] - \Pr[\,\mathbf{Exp}_{\mathcal{G},A}^{\text{odh-rand-ro}} = 1 \wedge \textsc{AskA}\,] \\
&\leq \Pr[\,\textsc{AskA}\,] \\
&\leq \mathbf{Adv}_{\mathcal{G},B}^{\text{sdh}} \\
&\leq \mathbf{Adv}_{\mathcal{G}}^{\text{sdh}}(t + q_o q_h O(hLen + gLen), (q_o+q_h)^2).
\end{aligned}
$$

The bound claimed in the theorem follows easily from the fact that A was an arbitrary adversary subject to the constraint that it had time-complexity at most t and made at most q_h queries to its H-oracle and at most q_o queries to its $\mathcal{H}_v$-oracle. ▌▌

LOWER BOUNDS WITH RESPECT TO GENERIC ALGORITHMS. Generic algorithms in groups are algorithms which do not make use of any special properties of the encoding of group elements other than assuming each element has a unique representation. This model was introduced by Shoup [34] and is very useful in proving lower bounds (with respect to such algorithms) for some problems. In fact, Shoup proved that in such a model both the discrete logarithm and the Diffie-Hellman problems are hard to solve as long as the order of the group contains at least one large prime factor. Following the same approach, we also use this model here to prove lower bounds for some new problems we introduce. Let us proceed now with the formalization of this model.

Let $Z_n = \{1,\ldots,n\}$ be the additive group of integers modulo n, the order of the group. Let S be a set of bit strings of order at least n. We call an injective map from Z_n to S an *encoding function*. One example for such a function would be the function taking $u \in Z_{|G|}$ to $g^{u \bmod |G|}$, where G is a finite cyclic group of order $|G|$ generated by the group element g.

A generic algorithm is a probabilistic algorithm A which takes as input a list

$$(\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_k)),$$

where each $x_i \in Z_n$ and σ is a random *encoding function*, and outputs a bit string. During its execution, A can make queries to an oracle Σ. Each query will result in updating the encoding list, to which A has always access. Σ gets as input two indices i and j and sign bit, and then computes $\sigma(x_i \pm x_j)$ and appends it to the list. It is worth noticing that A does not depend on σ, since it is only accessible by means of oracle queries.

We need to extend the original generic model to allow queries to the restricted DDH oracle $\mathcal{O}_v$. In this case, $\mathcal{O}_v$ gets as input two indices i and j and returns 1 if $x_j = v \cdot x_i$ and 0, otherwise. In general lines, our result shows that the restricted DDH oracle $\mathcal{O}_v$ does not help in solving the Diffie-Hellman problem whenever the group order contains a large prime factor. One should note, however, that our result has no implications on non-generic algorithms, such as index-calculus methods for multiplicative groups of integers modulo a large prime. Let us state this more formally.

Definition 11 [SDH in generic model] Let Z_n be the additive group of integers modulo n, let S be a set of strings of cardinality at least n, and let σ be a random encoding function of Z_n on S. In addition, let Ω be the set of all mappings Z_n to S. Let A be an generic algorithm. Consider the experiment

$$
\begin{array}{l}
\texttt{experiment } \mathbf{Exp}_n^{\text{g-sdh}} \\
\quad \sigma \xleftarrow{R} \Omega \\
\quad g \leftarrow \sigma(1) \\
\quad u \xleftarrow{R} \{1, \ldots, n\};\ U \leftarrow \sigma(u) \\
\quad v \xleftarrow{R} \{1, \ldots, n\};\ V \leftarrow \sigma(v) \\
\quad \Sigma(i,j,\pm) \stackrel{\text{def}}{=} x_i \pm x_j \\
\quad \mathcal{O}_v(i,j) \stackrel{\text{def}}{=} (x_j = v x_i) \\
\quad Z \leftarrow A^{\mathcal{O}_v(\cdot,\cdot),\Sigma(\cdot,\cdot,\cdot)}(g, U, V) \\
\quad \texttt{if } Z = \sigma(uv) \texttt{ then } b \leftarrow 1 \texttt{ else } b \leftarrow 0 \\
\quad \texttt{return } b
\end{array}
$$

Now define the *advantage* of A in violating the strong Diffie-Hellman assumption in the generic model and the advantage function of SDH assumption in this model, respectively, as follows:

$$
\begin{aligned}
\mathbf{Adv}_{n,A}^{\text{g-sdh}} &= \Pr[\, \mathbf{Exp}_{n,A}^{\text{g-sdh}} = 1 \,] \\
\mathbf{Adv}_{n}^{\text{g-sdh}}(q) &= \max_{A}\{\, \mathbf{Adv}_{n,A}^{\text{g-sdh}} \,\} ,
\end{aligned}
$$

where q is the total number of queries made by A to its oracles. $\diamondsuit$

Theorem 7 Let Z_n be the additive group of integers modulo n, let S be a set of strings of cardinality at least n. Then, for any number q,

$$\mathbf{Adv}_{n}^{\text{g-sdh}}(q) \leq O(q^2/p)$$

where p is the largest prime factor of n.

A corollary of Theorem 7 is that any generic algorithm solving the Diffie-Hellman problem under SDH with success probability bounded away from 0 has to perform at least $\Omega(p^{1/2})$ group operations.

Proof: Here we just present a proof sketch using a technique used by Shoup in [34]. Let $n = s\, p^t$ with $\gcd(s,p) = 1$. Since additional information only reduces the running time, we can assume that solving the Diffie-Hellman problem in the subgroup of order s is easy. Hence, let $n = p^t$ wlog.

We start by running algorithm A. In doing so, we need to simulate all its oracles. We play the following game. Let U and V be indeterminants. During the execution of the algorithm, we will maintain a list $F_1, \ldots, F_k$ of polynomials in $Z_{p^t}[U,V]$, along with a list $\sigma_1, \ldots, \sigma_k$ of distinct values in S. Initially, we have $F_1 = 1$, $F_2 = U$, and $F_3 = V$; and three distinct values σ_1, σ_2, and σ_3 chosen at random from S. When the algorithm makes a query $(i,j,\pm)$ to its Σ-oracle, we first compute $F_{k+1} = F_i \pm F_j \in Z_{p^t}[U,V]$ and check whether there is some $l \leq k$ such that $F_{k+1} = F_l$. If so, then we return σ_l to A. Else we pick choose a random but distinct σ_{k+1}, return it to A, and update both lists. When the algorithm makes a query (i, j) to its $\mathcal{O}_v$, we return 1 if $F_j = V \cdot F_i$ else 0.

We can assume that A outputs an element in the encoding list (otherwise $\mathbf{Adv}^{\mathrm{sdh}}_{n,A} \leq 1/(p-q)$), where q is the number of queries made by A. Then, let us choose u and v at random from Z_{p^t}. Notice that $\mathbf{Adv}^{\mathrm{sdh}}_{n,A}$ can be upper bounded by the probability of one of the following happening: $F_i(u,v) = F_j(u,v)$ for some F_i and F_j; or $F_i(u,v) = uv$ for some i; or $F_j \neq V_i$ but $F_j(u,v) = vF_i(u,v)$. Otherwise, the algorithm cannot learn anything about u or v except that $F_i(u,v) \neq F_j(u,v)$ for every i and j. But, using results from [34], for fixed i and j, the probability that $F_i - F_j$ vanishes is at most $1/p$; the probability that $F_i - UV$ vanishes is at most $2/p$; and the probability that $F_j - VF_i$ vanishes is at most $2/p$. It follows that the probability of one of these happening is $O(q^2/p)$. The theorem follows from the fact that A was an arbitrary adversary subject to the constraint that it makes at most q queries to its oracles. ▌ ▌

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, Berlin Germany, Apr. 2001.

[2] 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.

[3] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1996.

[4] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In IEEE, editor, *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, 1997.

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

[6] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1994.

[7] M. Bellare and P. Rogaway. Optimal asymmetric encryption: How to encrypt with RSA. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1994. `http://www-cse.ucsd.edu/users/mihir`.

[8] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and rabin. In U. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1996.

[9] M. Bellare and P. Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. In *Information and Communications Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin Germany, 1997.

[10] D. Bleichenbacher. A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1998.

[11] D. Boneh. The decision diffie-hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, 1998. Invited paper.

[12] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1996.

[13] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *30th Annual ACM Symposium on Theory of Computing*, New York, NY, May 23–26 1998. ACM Press.

[14] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1998.

[15] C. D. D. Dolev and M. Naor. Non-malleable cryptography. In ACM, editor, *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, Louisiana, May 6–8 1991. ACM Press.

[16] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1978.

[17] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography, 1998. Manuscript.

[18] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[19] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1999.

[20] O. Goldreich. A uniform complexity treatment of encryption and zero-knowledge. *IACR Journal of Cryptology*, 6(1):21–53, 1993.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[22] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1998.

[23] 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.

[24] D. Johnson and M. P. S. Matyas. Encryption of long blocks using a short-block encryption procedure, Nov. 1996. `http://stdsbbs.ieee.org/groups/1363/index.html`.

[25] C. Lim and P. Lee. Another method for attaining security against adaptively chosen ciphertext attacks. In D. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1994.

[26] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, Apr. 1988. Special issue on cryptography.

[27] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In IEEE, editor, *38th Annual Symposium on Foundations of Computer Science*, Miami Beach, FL, Oct. 19 - 22 1997. IEEE, IEEE Computer Society Press.

[28] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In ACM, editor, *22nd Annual ACM Symposium on Theory of Computing*, Baltimore, Maryland, May 14–16 1990. ACM Press.

[29] PKCS #1: RSA cryptography standard. RSA Data Security, Inc., June 1991.

[30] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, Aug. 1991.

[31] Certicom research, standards for efficient cryptography group (SECG) — sec 1: Elliptic curve cryptography. `http://www.secg.org/secg\_docs.htm`, Sept. 20 2000. Version 1.0.

[32] Secure hash standard. National Institute of Standards and Technology, NIST FIPS PUB 180-1, U.S. Department of Commerce, Apr. 1995.

[33] V. Shoup. Personal Communication.

[34] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 1997.

[35] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Germany, May 2000.

[36] Y. Zheng. Public key authenticated encryption schemes using universal hashing. Contribution to P1363. `ftp://stdsbbs.ieee.org/pub/p1363/contributions/aes-uhf.ps`.

[37] Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attack. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.

A Attacks on the ElGamal Scheme

ElGamal encryption fails to achieve strong notions of security, such as non-malleability and chosen-ciphertext security, in any represented group. In fact, it does not even achieves semantic security in some groups, such as Z_p^*. To support these claims, we here provide the reader with examples of attacks on the ElGamal scheme.

The first of these attacks against the ElGamal scheme shows that it is not semantically secure when Z_p^* is the underlying group of $\mathcal{G}$, p is a prime, and g is a generator. The attack is based on the fact that we can check whether a number $x \in \mathsf{Z}_p^*$ is a square or not in polynomial time by computing the value $x^{(p-1)/2} \bmod p$, which is 1 if x is a quadratic residue mod p and -1, otherwise. In the find stage, we choose two messages in Z_p^*, one which is a square and one which is not. In the guess stage, we first check whether g^u and g^v are square. We know that g^{uv} is a non-square if and only if both g^u and g^v are non-square. Then, knowing this, we can tell which message was encrypted by checking whether the encrypted message $M \cdot g^{uv}$ is a square or not. That is, if g^{uv} is a square, then $M \cdot g^{uv}$ is a square if and only if M is a square. If g^{uv} is a non-square, then $M \cdot g^{uv}$ is a square if and only if M is a non-square.

In order to provide a malleability attack against the ElGamal scheme, we can see that, given a ciphertext $EM = (g^u, encM)$ where $encM = M \cdot g^{uv}$, we can easily produce a valid ciphertext $\widetilde{EM}$ by just modifying the second part of EM. That is, if we multiply $encM$ by some value g^k ($k \neq 0$) to obtain $\widetilde{encM}$, then the resulting ciphertext $\widetilde{EM} = (g^u, encM)$ will be an encryption for a message $\widetilde{M} = M \cdot g^k$ because the value of g^{uv} does not change in this case. Note that this is not dependent on which group G is being used.

To provide a chosen-ciphertext attack against the ElGamal scheme, we can show that we can obtain the plaintext for any given ciphertext. Let $EM = (g^u, encM)$ be the challenge ciphertext. Let $\widetilde{encM}$ be a point in G such that $\widetilde{encM} \neq encM$ and let $\widetilde{M}$ be the decryption of $\widetilde{EM} = (g^u, \widetilde{encM})$. As we know that $\widetilde{M} = \widetilde{encM}/g^{uv}$, we can compute g^{uv} and then $encM/g^{uv}$, which is the decryption of EM.