Submission to IEEE P1363a.

DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem

Michel Abdalla* Mihir Bellare† Phillip Rogaway‡

September 1998

Abstract

This paper describes a Diffie-Hellman based encryption scheme, DHAES. 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.

We show that DHAES has not only the "basic" property of secure encryption (namely privacy under a chosen-plaintext attack) but also achieves privacy under both non-adaptive and adaptive chosen-ciphertext attacks. (And hence it also achieves non-malleability.)

DHAES is built in a generic way from lower-level primitives: a symmetric encryption scheme, a message authentication code, group operations in an arbitrary group, and a cryptographic hash function. In particular, the underlying group may be an elliptic-curve group or the multiplicative group of integers modulo a prime number.

The proofs of security are based on appropriate assumptions about the hardness of the Diffie-Hellman problem and the assumption that the underlying symmetric primitives are secure. The assumptions are all standard in the sense that no random oracles are involved.

We suggest that DHAES provides an attractive starting point for developing public-key encryption standards based on the Diffie-Hellman assumption.

Keywords: Cryptography, Diffie-Hellman key exchange, ElGamal encryption, Elliptic curve cryptosystems, 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-cse.ucsd.edu/users/mabdalla. 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 Description of the Scheme

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 (eg., modular exponentiation) which are at the core of a Diffie-Hellman based encryption.

The method we suggest is called DHAES. It is as efficient as ElGamal encryption, but has more and better security properties. The scheme is versatile, in that it can be used in any setting where the Diffie-Hellman problem is hard — so not only the group of integers modulo a prime (or subgroups thereof), but also elliptic-curve groups.

DHAES 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.

In this section we specify DHAES; the following sections provide background and comparisons, and explain the sense in which DHAES is demonstrably secure. To specify our scheme in a compact and precise way, we first specify the "syntax" of an asymmetric encryption scheme and the types of primitives which our asymmetric encryption scheme employs.

1.1 Preliminaries

Represented Groups. DHAES 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 g by itself u times. 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 explicitly invoke any encoding functions in our scheme descriptions and proofs.

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 $\mathsf{GROUP} = (G, g, \_, \uparrow)$ a *represented group.*

We will be making assumptions about the hardness of the Diffie-Hellman problem in a given represented group.

Message Authentication Codes. A message authentication scheme enables users sharing a secret key to tag data for the purpose of authenticity and integrity. DHAES will make use of such a scheme. Any scheme meeting the security requirements will be adequate.

To describe the operation of a scheme we first let $\mathsf{Message} = \{0,1\}^*$ be the space of message we can MAC. 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 $\mathsf{MAC} = (\mathsf{MAC.gen}, \mathsf{MAC.ver})$. Algorithm $\mathsf{MAC.gen}$ (the *MAC generation algorithm*) takes a key $k \in \mathsf{mKey}$ and a message $x \in \mathsf{Message}$ and returns a string $\mathsf{MAC.gen}(k, x)$. (One could also have allowed $\mathsf{MAC.gen}$ to be probabilistic.) This string is called the *tag*. Algorithm $\mathsf{MAC.ver}$ (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 $\mathsf{MAC.ver}(k, x, \tau) \in \{0,1\}$, 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 $x \in \mathsf{Message}$, $\mathsf{MAC.ver}(k, x, \mathsf{MAC.gen}(k, x)) = 1$. The first argument of either algorithm may be written as a subscript.

We assume for simplicity that the MAC generation algorithm and the verification algorithm are deterministic. This is true in most existing schemes.

Candidate algorithms for the MAC are HMAC [2, 27], the CBC MAC based on a block cipher with large key and block sizes [5], or any of a variety of MACs based on the Wegman-Carter paradigm [37].

SYMMETRIC ENCRYPTION. A symmetric encryption scheme permits users sharing a key to encrypt data to achieve privacy. DHAES can use any such scheme meeting the appropriate security requirements discussed later. To describe the components of such a scheme let 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 *symmetric encryption scheme* is a pair of algorithms $\mathsf{SYM} = (\mathsf{SYM.enc}, \mathsf{SYM.dec})$. Algorithm SYM.enc (the *encryption algorithm*) takes a key $K \in \mathsf{eKey}$, a plaintext $x \in \mathsf{Message}$, and coins $r \in \mathsf{Coins}$, and returns ciphertext $\mathsf{SYM.enc}(k, x, r)$. Algorithm SYM.dec (the *decryption algorithm*) takes a key $k \in \mathsf{eKey}$ and a purported ciphertext $y \in \mathsf{Ciphertext}$, and returns a value $\mathsf{SYM.dec}(k, y) \in \mathsf{Message} \cup \{\mathsf{BAD}\}$. We require that for all $x \in \mathsf{Message}$, $k \in \mathsf{Key}$, and $r \in \mathsf{Coins}$, $\mathsf{SYM.dec}(k, \mathsf{SYM.enc}(k, x, r)) = x$. Usually we omit mentioning the coins of SYM.enc, thinking of SYM.enc as a probabilistic algorithm, or thinking of $\mathsf{SYM.enc}(k, x)$ as the induced probability space. A return value of BAD from SYM.dec 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.

Candidate algorithms for the symmetric encryption are CBC encryption with a block cipher of large enough key and block size [3], or some other mode meeting the security requirements of [3] or Definition 4. However, the security theorems will show that the symmetric encryption scheme can be quite weak, in that it suffices to be able to securely encrypt a single message. Thus, a simple way to implement it is to use the key as a seed, apply a pseudorandom bit generator to it to get a pad, and XOR the message with this pad to get the ciphertext.

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

1.2 Definition of DHAES

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

Let $\mathsf{GROUP} = (G, g, , \uparrow)$ be a represented group, where group elements are represented by strings of *gLen* bits. Let $\mathsf{SYM} = (\mathsf{SYM.enc}, \mathsf{SYM.dec})$ be a symmetric encryption scheme with key length *eLen*, and let $\mathsf{MAC} = (\mathsf{MAC.gen}, \mathsf{MAC.ver})$ be a message authentication code with key length *mLen* and tag length *tLen*. Let $H : \{0,1\}^{2gLen} \to \{0,1\}^{mLen+eLen}$ be a function. From these primitives we define the asymmetric encryption scheme $\mathsf{DHAES} = (\mathsf{DHAES.enc}, \mathsf{DHAES.dec}, \mathsf{DHAES.key})$. If we want to explicitly indicate the dependency of DHAES on its associated primitives, then we will write $\mathsf{DHAES}[\![\mathsf{GROUP}, \mathsf{SYM}, \mathsf{MAC}, \mathsf{H}]\!]$. The component algorithms of DHAES —the encryption algorithm, decryption algorithm, and key-generation

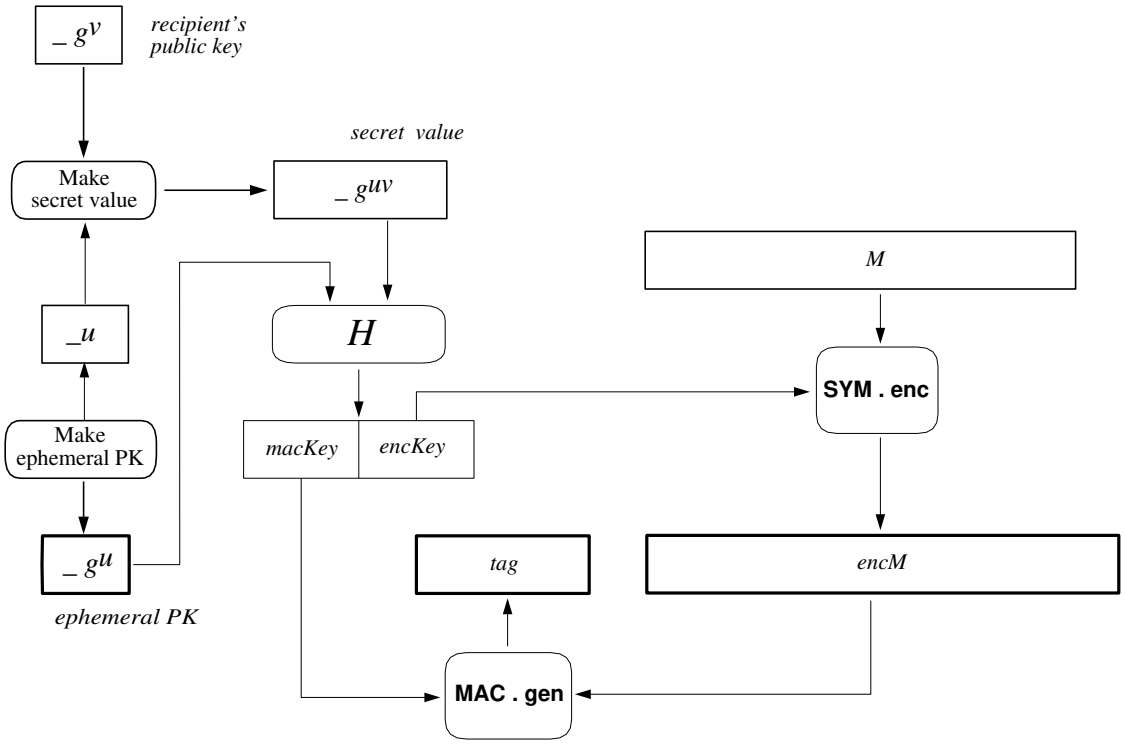

Figure 1: *Encrypting with the scheme* DHAES. *We use a symmetric encryption algorithm,* SYM.enc*; a MAC generation algorithm,* MAC.gen*; and a hash function, H. The emboldened rectangles comprise the ciphertext.*

algorithm— are defined in Figure 2.

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, along with the ephemeral public key, g^u. The result is parsed 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.

1.3 Notes

The conventions of [23] associate each scheme with exactly one "cryptographic family," where three cryptographic families are currently described: discrete logarithm over finite fields (DL); discrete logarithm over elliptic curve groups (EC); and integer factorization (IF). Since DHAES works over an arbitrary represented cyclic group, it is natural to use it to define two DHAES schemes in the sense of [23]: DLES and ECES.

Algorithm DHAES.enc(pk, M)
begin
  $u \leftarrow \{1, \ldots, |G|\}$
  $X \leftarrow pk \uparrow u$
  $U \leftarrow g \uparrow u$
  $hash \leftarrow H(U \parallel X)$
  $macKey \leftarrow hash[1 \,..\, mLen]$
  $encKey \leftarrow hash[mLen + 1 \,..\, mLen + eLen]$
  $encM \leftarrow \mathsf{SYM.enc}(encKey, M)$
  $tag \leftarrow \mathsf{MAC.gen}(macKey, M)$
  $EM \leftarrow U \parallel tag \parallel encM$
  return EM
end

Algorithm DHAES.dec(sk, EM)
begin
  $U \parallel tag \parallel encM \leftarrow EM$
  $X \leftarrow U \uparrow sk$
  $hash \leftarrow H(U, X)$
  $macKey \leftarrow hash[1 \,..\, mLen]$
  $encKey \leftarrow hash[mLen + 1 \,..\, mLen + eLen]$
  if $\mathsf{MAC.ver}(macKey, encM, tag) = 0$
    then return BAD
  $EM \leftarrow \mathsf{SYM.dec}(encKey, encM)$
  return EM
end

Algorithm DHAES.key
begin
  $v \leftarrow \{1, \ldots, |G|\}$
  $pk \leftarrow g \uparrow v$
  $sk \leftarrow v$
  return (pk, sk)
end

Figure 2: *The scheme* DHAES = (DHAES.enc, DHAES.dec, DHAES.key)*, where:* SYM *is a symmetric encryption scheme using keys of length eLen;* MAC *is a message authentication code with keys of length mLen and tags of length tLen;* GROUP $= (G, g, , \uparrow)$ *is a represented group whose group elements encoded by strings of length gLen; and* $H : \{0,1\}^{2gLen} \to \{0,1\}^{eLen+mLen}$.

2 Attributes and Advantages of the Scheme

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

2.1 Encrypting with Diffie-Hellman: The ElGamal Scheme

Let G be a finite cyclic group, say $G = \mathsf{Z}_p^*$, 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 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 [19] 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.

2.2 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 shouldn't 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 $G = \mathsf{Z}_p^*$, 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 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 *more* than shield information about the plaintext in the presence of a passive attack. Stronger goals include non-malleability [17] and chosen-ciphertext security [33, 36]. 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 she 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 *wrong* if instantiated by a primitive which achieves *only* 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 has recently shown that encryption under RSA PKCS #1 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.

2.3 Overcoming Deficiencies in ElGamal Encryption: DHES

The scheme we have presented, DHAES, 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 DHAES 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 $\langle g^u, g^v, H(g^u \parallel g^{uv})\rangle$ looks like a pair of random group elements together with a random string. For non-trivial functions H this assumption —that H is hardcore 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 both non-adaptive and adaptive chosen-ciphertext attacks. This is proved under an assumption called the Diffie-Hellman independence assumption, and assuming the underlying MAC and encryption schemes are secure.

It is shown in [3, 18] 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 [14].

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 *off-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 DHAES. 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.

2.4 More on Provable Security

It is easy to come up with a DH-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.

2.5 Concrete Security

Following works such as [8, 9], 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 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.

2.6 Related Work

The current work has grown out of interest from within the IEEE P1363 committee, which has been drafting a bit-level standard to cover a variety of cryptographic aims [23]. A version of DHAES is suggested in the draft standard [1]. The scheme described here was first proposed in an earlier version of this work [6]. The current document provides the technical support for the DHAES proposal.

We view DHAES as the natural adaptation of the ElGamal scheme to withstand active attacks and apply to arbitrary length messages. The use of symmetric primitives is based on the classical cryptographic paradigm of first encrypting a symmetric key and then using the latter to process the actual data.

Zheng and Seberry [39] have proposed an ElGamal adaptation 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 [28] have pointed out that in some of the cryptosystems proposed in [39], the method of adding authentication capability may fail just under known plaintext attacks. A submission to IEEE P1363a based on [39] has been made by Zheng [38].

Another contemporaneous suggestion was put forward by Johnson, Matyas and Peyravian [26]. 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, [26] 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, but no such proof has been given.

Recently Cramer and Shoup described an encryption scheme based on the decisional Diffie-Hellman problem which achieves security against adaptive chosen-ciphertext attack [15]. Their scheme has the advantage of provably meeting a strong notion of security under a standard assumption. Their assumption is weaker than the ones used in this paper, so that the theoretical guarantees provided by their scheme should be considered superior to ours.[1] However, this is at the cost of efficiency: their scheme is more costly than ours in terms of key sizes, encryption time, and decryption time. (In particular, encryption takes five exponentiations.)

3 Security Assessment of the Scheme

Some simple variants of DHAES do not retain its security properties. Before getting into the definitions and proofs of security for DHAES, let us demonstrate that point by looking at a simple variant of DHAES is not

[1] Strictly speaking, the assumptions are not comparable due to the presence of the hash function in our assumptions, but they do not need the kind of independence assumptions we make, and hence their assumptions are in spirit weaker.

secure.

3.1 Variants

AN INCORRECT VARIANT OF DHAES. One of the more mysterious aspects of DHAES may be the placing of $U = g^u$, the "ephemeral public key," in the scope of H. The reason for feeding g^u into H is to ensure non-malleability and chosen-ciphertext security. This stems from the fact that, in some represented groups GROUP (including 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 when g^u is not part of the input of H. Therefore, if, given g^u and g^v, we can compute $g^{u'}$ in such a way that $g^{uv} = g^{u'v}$ holds with high probability, then we would break the scheme in the malleability sense.

As a concrete example, let p be a prime number, Z_p^* the group, and g a generator of it. Here is an attack showing that scheme fails non-malleability when we do not feed g^u to H. Let $\ell = (p-1)/2$. Then, using a basic result from number theory which states that $g^\ell = -1$, we can show that, given a ciphertext $EM = g^u \parallel encM \parallel tag$, we can generate a new ciphertext $EM' = g^{u'} \parallel encM \parallel tag$ which is a encryption for the same message with very high probability. Let $g^{u'} = g^u \cdot g^\ell$. In this case, $g^{u'v} = g^{uv} \cdot g^{\ell v} = g^{uv} \cdot (g^\ell)^v = g^{uv} \cdot (-1)^v$. Thus, whenever v is even, $g^{uv} = g^{u'v}$. As the probability for this to happen is $1/2$, EM and EM' would represent the encryption for the same message half of the time if g^u is not fed into H.

Note this implies the scheme is insecure against adaptive chosen ciphertext attack, since otherwise it would be non-malleable [3, 18].

SOME CORRECT VARIANT OF DHAES. In [6], on encryption we provided MAC.gen with a further argument: a string which encodes (unspecified, publicly known) "auxiliary information." We believe that this may be useful: it has the effect of binding to the ciphertext the (non-secret) auxiliary information, so that the same auxiliary information must be presented to decrypt a ciphertext. Since we never formalized the associated security assertion we have ceased to carry this into the present exposition.

In [6] we also provided MAC.gen with a string which encodes the length of $|M|$. This had historical reasons for being there, but no formal significance. So we have again ceased to carry this forward.

3.2 Security assumptions about DHES primitives

DHAES uses three primitives as discussed in Section 1.1. We require (1) group-theoretic operations corresponding to the Diffie-Hellman assumption; (2) a symmetric encryption scheme; and (3) a message authentication code. In this section we give quantitative security definitions for these notions, measuring adversarial success in relation to adversarial resources.

Note that DHAES also uses a hash function H. This is not discussed separately, but rather as part of (1): we make assumptions about the quality of the bits that H extracts from the DH key. At the end of Section 3.2.1 we discuss choices of H in light of the DH assumptions we make there.

3.2.1 Diffie-Hellman Problem

DIFFIE-HELLMAN ASSUMPTIONS. We refer to the standard Diffie-Hellman assumption as the *computational Diffie-Hellman assumption*, CDH-A. It states that given g^u, g^v, where u, v were drawn at random from $\{1, \ldots, |G|\}$, it is hard to compute g^{uv}. However, given just CDH-A, it might still be possible for the adversary to compute something interesting about g^{uv}, such as its most significant bit. Thus this assumption is too weak to be fruitful in the design of encryption schemes. (Even the ElGamal scheme is not semantically

secure given only this assumption.) We must make some assumption about the unpredictability of bits of the DH key rather than just the ability to compute the entire key.

A stronger assumption that is gaining popularity these days is the *Decisional Diffie-Hellman assumption*, DDH-A. (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 of prime order, and in such groups suffices to prove the semantic security of the ElGamal scheme. Note that DDH-A implies CDH-A in any group in which the DDH-A holds.

The assumption we make to prove the semantic security of DHES is actually weaker than the DDH-A, but still stronger than the CDH-A. (Remember, the weaker the assumption under which security can be proven, the better!) It is called the *Hash Diffie-Hellman assumption*, HDH-A. To prove the security of DHES 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 DH 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. As indicated above, semantic security of a DH based scheme requires that we be able to get some number of "hardcore" bits from the DH key, namely key derived bits that cannot be distinguished from random bits. Our assumption is that applying a suitable hash function H to $g^u \,\|\, g^{uv}$ will yield such bits. The assumption we make, called the *Hash Diffie-Hellman assumption*, HDH-A, is a composite one, in the sense that the assumption speaks about the interaction between the hash function and the Diffie-Hellman problem.

The HDH-A states that for reasonable values of the time t invested by the adversary, the quantity $InSec^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}\,;\, t)$ defined below is small.

Definition 1 *Let* $\mathsf{GROUP} = (G, g, \_, \uparrow)$ *be a represented group, let hLen be a number, let* $H : \{0,1\}^* \to \{0,1\}^{hLen}$*, and let* A *be an adversary. The advantage of* A *in violating* H *being hardcore on* GROUP *is*

$$\begin{aligned} Adv_A^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}) \stackrel{\text{def}}{=} & \Pr\left[u, v \leftarrow \{1, \ldots, |G|\} : \; A(g^u, g^v, H(g^u \,\|\, g^{uv})) = 1\right] - \\ & \Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\; r \leftarrow \{0,1\}^{hLen} : \; A(g^u, g^v, r) = 1\right] . \end{aligned}$$

The security of H *on* GROUP *is the function*

$$InSec^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}\,;\, t) \stackrel{\text{def}}{=} \max_A \left\{Adv_A^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP})\right\},$$

where the maximum is over all adversaries A *running in time at most* t*.*

Informally, adversary A breaks (GROUP, H) if A spends "reasonable" time t in order to get "significant" advantage ϵ. Certainly A can break (GROUP, H) if it is easy to compute discrete logarithms (base g) of random group elements, or solve the computational DH problem. But to make an efficient encryption scheme under weak assumptions we make the potentially stronger assumption we have just described.

Here and throughout this paper "running time" is understood to mean the maximal number of steps that the algorithm requires (relative to some fixed model of computation) plus the size of the encoding of the algorithm (relative to some fixed convention on writing algorithms). Henceforth we continue to employ this convention that running time actually includes the space for the algorithm's description.

We are considering the complexity of adversaries who try to attack a specific represented group GROUP. Such an adversary may depend on GROUP, so explicitly providing a description of GROUP to A is unnecessary.

This should be compared with the Decisional Diffie-Hellman assumption discussed above. That assumption says that the DH key g^{uv} looks like a random group element. In groups where the DDH-A is true, one

can almost imagine that setting H to return the DH key g^{uv} will suffice to make the HDH-A true. This is almost, but not quite true, the difference being that between a random group element and random string of some length. See further discussion below.

HDH INDEPENDENCE ASSUMPTIONS: MOTIVATING DISCUSSION. We will now strengthen the HDH-A to say it also holds in the presence of oracles that give information about DH keys, as long as these keys are sufficiently "independent" of the target instance. We make two assumptions, one under which we will prove the security of DHES under non-adaptive chosen-ciphertext attack; the second, stronger one, for proving security against adaptive chosen-ciphertext attack. Before stating them let us try to provide some intuition.

The intuition is that the ability to compute g^{uv} from g^u and g^v does not seem to increase if one has access to a DH oracle solving the same problem, as long as one invokes the oracle only on instances "independent" from the target. We wish to capture this in a formal assumption. However, one needs to be careful, particularly with regard to what "independent" means.

The first setting we consider has a strong notion of "independence:" the target instance is simply not known at the time the oracle is present. More precisely consider an adversary that is given g^v and an oracle that takes $X \in G$ and returns X^v, where $v \in \{1, \ldots, |G|\}$ is unknown to the adversary. (This is a DH oracle relative to v, since if $X = g^u$ then $X^v = g^{uv}$ is the DH key corresponding to g^u and g^v.) We let the adversary play with this oracle for a while. Now take the oracle away, and then give the adversary a challenge g^u and ask it to compute g^{uv}. Our assumption is that it will fail. The independence here is in the fact that g^u (the target) is given to the adversary only after the oracle is taken away. (Else of course it could invoke the oracle on $X = g^u$ and get back $X^v = g^{uv}$ at once. But not knowing the target in advance, the oracle appears to be useless.)

In the actual assumption made below (called the non-adaptive hash Diffie-Hellman independence assumption), we consider not the ability to compute g^{uv}, but to predict $H(g^u \parallel g^{uv})$, as before; namely we want to say that the hardcore bits corresponding to the target remain hard even when the oracle is allowed in the pre-processing stage. We also *weaken* the oracle, having it return not the actual DH key X^v, but the corresponding hash $H(X \parallel X^v)$. (This is not crucial to the assumption, but weakens it, and is all we need.) This assumption will be used to prove the security of DHES under non-adaptive chosen-ciphertext attack.

To prove the security of DHES under adaptive chosen-ciphertext attack, we must strengthen the assumption to allow oracle queries that are a function of the target g^u. Of course we cannot allow g^u itself to be queried, so that particular query is disallowed.

Notice that in the adaptive case, it is crucial to consider the weakened oracle that given X returns not X^v but $H(X \parallel X^v)$. For otherwise (meaning if the oracle returned X^v) we could exploit the self-reducibility of the discrete log problem to obtain g^{uv} from the oracle without querying g^u. For example let $X = g \cdot g^u$, query it to get back $X^v = g^v \cdot g^{uv}$, and divide out by g^v (which we know) to get back g^{uv}. This kind of attack does not however appear possible when the oracle returns $H(X \parallel X^v)$ rather than X^v.

Now let us state the assumptions formally.

NON-ADAPTIVE HDH INDEPENDENCE ASSUMPTION. Given a represented group $\mathsf{GROUP} = (G, g, \_, \uparrow)$, a hash function H and a number $v \in \{1, \ldots, |G|\}$, we let HDH_v be an oracle, called the *HDH oracle*, which behaves as follows:

$$\mathrm{HDH}_v(X) \quad = \quad H(X \parallel X^v)$$

for all $X \in G$. In other words, $\mathrm{HDH}_v(g^x) = H(g^x \parallel g^{vx})$ for any $x \in \{1, \ldots, |G|\}$. Now the following definition tells us when an adversary is considered successful. The *non-adaptive hash Diffie-Hellman independence assumption*, HDHI1-A, says that the quantity $\mathit{InSec}^{\mathrm{DHI1}}(H, \mathsf{GROUP}\,;\, t, q)$ defined below is small as long as t and q are not too large.

Definition 2 *Let* $\mathsf{GROUP} = (G, g, \_, \uparrow)$ *be a represented group, let hLen be a number, let* $H : \{0,1\}^* \to \{0,1\}^{hLen}$*, and let* A *be an adversary with access to a DH oracle. Then the advantage of* A *in violating* H *being hardcore on* GROUP *under non-adaptive DH attack is*

$$Adv_A^{\mathrm{DHI1}}(H, \mathsf{GROUP}) \stackrel{\mathrm{def}}{=}$$
$$\Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\ s \leftarrow A^{\mathrm{HDH}_v(\cdot)}(\mathsf{find}, g^v);\ A(\mathsf{guess}, s, g^u, H(g^u \parallel g^{uv})) = 1\right] -$$
$$\Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\ s \leftarrow A^{\mathrm{HDH}_v(\cdot)}(\mathsf{find}, g^v);\ r \leftarrow \{0,1\}^{hLen} :\ A(\mathsf{guess}, s, g^u, r) = 1\right] .$$

The security of H *on* GROUP *is the function*

$$InSec^{\mathrm{DHI1}}(H, \mathsf{GROUP}\,;\, t, q) \quad \stackrel{\mathrm{def}}{=} \quad \max_A \left\{Adv_A^{\mathrm{DHI1}}(H, \mathsf{GROUP})\right\} ,$$

where the maximum is over all adversaries A *running in time at most* t *and making at most* q *queries to its oracle.*

The formalization in the definition captures the setting explained above: the adversary is given access to a HDH oracle in a first find stage, but at that time is not given any information about u. (In particular is not given g^u, else A could just call the DH oracle on g^u and get back $H(g^u \parallel g^{uv})$.) At the end of this stage A outputs some state information s, which is passed on to the second stage, where A must now guess $H(g^u \parallel g^{uv})$, having only just got g^u. At this point, A no longer has the HDH oracle. (Note that the task before A is to distinguish $H(g^u \parallel g^{uv})$ from a random string r, as in the HDH assumption above. This can certainly be done if A can compute g^{uv}, but we are making the stronger assumption that H extracts some un-predictable hardcore bits from g^{uv}.)

We note that the presence of the hash function H in the definition of the HDH oracle does not appear to be crucial: the assumption appears to be true even if the oracle simply returns the DH key X^v. In weakening the oracle we are weakening the assumption, which is good. This will not be true in the next assumption; there the presence of the hash in the oracle is crucial, as explained above.

ADAPTIVE HDH INDEPENDENCE ASSUMPTION. Given a represented group $\mathsf{GROUP} = (G, g, \_, \uparrow)$, a hash function H and a number $v \in \{1, \ldots, |G|\}$, we let HDH_v be the oracle defined above, namely

$$\mathrm{HDH}_v(X) \quad = \quad H(X \parallel X^v)$$

for all $X \in G$. Now the following definition tells us when an adversary is considered successful. The *Adaptive hash Diffie-Hellman independence assumption*, HDHI2-A, says that the quantity $InSec^{\mathrm{DHI2}}(H, \mathsf{GROUP}\,;\, t, q)$ defined below is small as long as t and q are not too large.

Definition 3 *Let* $\mathsf{GROUP} = (G, g, \_, \uparrow)$ *be a represented group, let hLen be a number, let* $H : \{0,1\}^* \to \{0,1\}^{hLen}$*, and let* A *be an adversary with access to a DH oracle. Then the advantage of* A *in violating* H *being hardcore on* GROUP *under adaptive DH attack is*

$$Adv_A^{\mathrm{DHI2}}(H, \mathsf{GROUP}) \quad \stackrel{\mathrm{def}}{=} \quad \Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\ A^{\mathrm{HDH}_v(\cdot)}(g^u, g^v, H(g^u \parallel g^{uv})) = 1\right] -$$
$$\Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\ r \leftarrow \{0,1\}^{hLen} :\ A^{\mathrm{HDH}_v(\cdot)}(g^u, g^v, r) = 1\right] .$$

Here A *is not allowed to call its oracle on* g^u*. The security of* H *on* GROUP *is the function*

$$InSec^{\mathrm{DHI2}}(H, \mathsf{GROUP}\,;\, t, q) \quad \stackrel{\mathrm{def}}{=} \quad \max_A \left\{Adv_A^{\mathrm{DHI2}}(H, \mathsf{GROUP})\right\} ,$$

where the maximum is over all adversaries A *running in time at most* t *and making at most* q *queries to its oracle.*

That is, 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. The assumption is that this still does not help predict the value of $H(g^u \parallel g^{uv})$.

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. (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 adaptive HDH independence assumption: it should be hard to recover g^{uv} from $H(g^u \parallel g^{uv})$, 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 hardcoreness under the CDH assumption. Under the DDH assumption, however, things get much easier, since this assumption already says that the DH 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 show that application of a function h chosen at random from a family of universal hash functions will suffice [32]. (Zheng and Seberry [39] had earlier suggested the application of a universal hash function to the DH key as a heuristic under the computational DH assumption. The result of [32] 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 [39] 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 independence assumptions above. In particular, the result of Boneh and Venkatesan [13] showing that computing the most significant bits of DH keys is as hard as computing the key itself can be turned on its head to give an algorithm to attack these assumptions. Namely, their results show that for some simple choices of functions H, an adversary can use the HDH oracle HDH_v defined above to solve the DH 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.)

A further drawback of using simple truncation or universal hash functions, even just for semantic security, is that we must make the DDH assumption, and thus must work in a group of prime order; recall one of the goals of our scheme was to be able to work in any group for which the Diffie-Hellman problem is hard.

3.2.2 Symmetric Encryption

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, 31]. We imagine an adversary A that runs in two stages. During either stage the adversary may query an encryption oracle $\mathsf{SYM.enc}(K, \cdot)$ which, on input x, returns $\mathsf{SYM.enc}(K, x, r)$ for a randomly chosen r. In the adversary's find stage she endeavors to come up with a pair of equal-length messages, x_0 and x_1, whose encryptions she wants to try to tell apart. She also retains some state information s. In the adversary's guess stage she 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 she 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 4 [4] *Let* $\mathsf{SYM} = (\mathsf{SYM.enc}, \mathsf{SYM.dec})$ *be a symmetric encryption scheme and let* A *be an adversary. The advantage of* A *in attacking* SYM *is*

$$Adv_A^{\mathrm{Sym}}(\mathsf{SYM}) \stackrel{\text{def}}{=} 2 \cdot \Pr\Big[K \leftarrow \mathsf{eKey};\ (x_0, x_1, s) \leftarrow A^{\mathsf{SYM.enc}(K,\cdot)}(\mathsf{find});\ b \leftarrow \{0,1\};$$
$$y \leftarrow \mathsf{SYM.enc}(K, x_b) :\ A^{\mathsf{SYM.enc}(K,\cdot)}(\mathsf{guess}, y, s) = b\Big] - 1 \ .$$

The security of SYM *is the function*

$$\mathit{InSec}^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t, \mu, m, m') \stackrel{\text{def}}{=} \max_A \{Adv_A^{\mathrm{Sym}}(\mathsf{SYM})\} \ ,$$

where the maximum is taken over all adversaries A *running in time at most* t*, asking queries which total at most* μ *bits, and whose output* x_0 *(and* x_1*) has length at most* m *bits, and* m' *bounds the length of a* SYM.enc*-produced ciphertext whose plaintext is of length* m*.*

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" for an adversary A is always understood to be the sum of the actual running time and the length of A's description.

Candidate algorithms were discussed in Section 1.1.

3.2.3 Message Authentication Codes

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 $\mathsf{MAC.gen}_K(\cdot)$ oracle, we say that F's output (x^*, τ^*) is *unasked* if τ^* is not the response of the $\mathsf{MAC.gen}_K(\cdot)$ oracle to an earlier query of x^*. Our definition of MAC security follows.

Definition 5 *Let* $\mathsf{MAC} = (\mathsf{MAC.gen}, \mathsf{MAC.ver})$ *be a message authentication scheme and let* F *be an adversary. Then the success (or forging probability) of* F *on* MAC *is*

$$\mathit{Succ}_A^{\mathrm{MAC}}(\mathsf{MAC}) \stackrel{\text{def}}{=} \Pr\Big[K \leftarrow \mathsf{mKey};\ (x^*, \tau^*) \leftarrow F^{\mathsf{MAC.gen}(K,\cdot)} :$$
$$\mathsf{MAC.ver}_K\,(x^*, \tau^*) = 1 \text{ and } (x^*, \tau^*) \text{ is unasked}\Big] \ .$$

The security of MAC *is the function*

$$\mathit{InSec}^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t, q) \stackrel{\text{def}}{=} \max_F \{\mathit{Succ}_F^{\mathrm{MAC}}(\mathsf{MAC})\} \ ,$$

where the maximum is taken over all adversaries F *running in time at most* t *and asking at most* q *oracle queries.*

Adversary F is said to have *forged* when, in the experiment above, F outputs an (x^*, τ^*) such that $\mathsf{MAC.ver}_K\,(x^*, \tau^*) = 1$ and (x^*, τ^*) is unasked.

This definition is stronger than the usual one as given in [5]. 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.

Candidate algorithms were discussed in Section 1.1.

3.2.4 Asymmetric Encryption

PRIVACY AGAINST CHOSEN-PLAINTEXT ATTACK. Our treatment mimics the find-then-guess notion of [4] and follows [21, 31, 20]. The definition is similar to Definition 4, so we state it without further discussion.

Definition 6 *Let* $\mathsf{ASYM} = (\mathsf{ASYM.enc}, \mathsf{ASYM.dec}, \mathsf{ASYM.key})$ *be an asymmetric encryption scheme and let* A *an adversary. The advantage of* A *in attacking* ASYM *is*

$$Adv^{\mathrm{Asym}}_{A}(\mathsf{ASYM}) \stackrel{\text{def}}{=} 2 \cdot \Pr\Big[(sk, pk) \leftarrow \mathsf{ASYM.key};\ (x_0, x_1, s) \leftarrow A\,(\mathsf{find}, pk)\,;\ b \leftarrow \{0,1\};$$
$$y \leftarrow \mathsf{ASYM.enc}_{pk}(x_b) :\ A\,(\mathsf{guess}, pk, s, y) = b\Big] - 1\ .$$

The security of ASYM *is the function*

$$\mathit{InSec}^{\mathrm{Asym}}(\mathsf{ASYM}\,;\, t, m) \stackrel{\text{def}}{=} \max_{A} \{Adv^{\mathrm{Asym}}_{A}(\mathsf{ASYM})\}\ ,$$

where the maximum is taken over all adversaries A *running in time at most* t *and whose output* x_0 *(and* x_1*) has length at most* m *bits.*

PRIVACY AGAINST NON-ADAPTIVE CHOSEN-CIPHERTEXT ATTACK. The definition of non-adaptive chosen-ciphertext security of an asymmetric encryption scheme is very similar to that given in Definition 6. The difference is that here the adversary is given access to a decryption oracle in the find stage. So we state it without further discussion.

Definition 7 *Let* $\mathsf{ASYM} = (\mathsf{ASYM.enc}, \mathsf{ASYM.dec}, \mathsf{ASYM.key})$ *be an asymmetric encryption scheme and let* A *an adversary for its non-adaptive chosen-ciphertext security. The advantage of* A *in attacking* ASYM *is*

$$Adv^{\mathrm{CCA1}}_{A}(\mathsf{ASYM}) \stackrel{\text{def}}{=} 2 \cdot \Pr\Big[(sk, pk) \leftarrow \mathsf{ASYM.key};\ (x_0, x_1, s) \leftarrow A^{\mathsf{ASYM.dec}_{sk}}\,(\mathsf{find}, pk)\,;$$
$$b \leftarrow \{0,1\};\ y \leftarrow \mathsf{ASYM.enc}_{pk}(x_b) :\ A\,(\mathsf{guess}, pk, s, y) = b\Big] - 1\ .$$

The security of ASYM *is the function*

$$\mathit{InSec}^{\mathrm{CCA1}}(\mathsf{ASYM}\,;\, t, q, \mu, m) \stackrel{\text{def}}{=} \max_{A} \{Adv^{\mathrm{Asym}}_{A}(\mathsf{ASYM})\}\ ,$$

where the maximum is taken over all adversaries A *running in time* t*, making at most* q *queries to its* $\mathsf{ASYM.dec}_{sk}$*-oracle, all these totaling at most* μ *bits, and whose output* x_0 *(and* x_1*) has length at most* m *bits.*

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 6. 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 8 *Let* $\mathsf{ASYM} = (\mathsf{ASYM.enc}, \mathsf{ASYM.dec}, \mathsf{ASYM.key})$ *be an asymmetric encryption scheme and let* A *an adversary for its chosen-ciphertext security. The advantage of* A *in attacking* ASYM *is*

$$Adv^{\mathrm{CCA2}}_{A}(\mathsf{ASYM}) \stackrel{\text{def}}{=} 2 \cdot \Pr\Big[(sk, pk) \leftarrow \mathsf{ASYM.key};\ (x_0, x_1, s) \leftarrow A^{\mathsf{ASYM.dec}_{sk}}\,(\mathsf{find}, pk)\,;$$
$$b \leftarrow \{0,1\};\ y \leftarrow \mathsf{ASYM.enc}_{pk}(x_b) :\ A^{\mathsf{ASYM.dec}_{sk}}\,(\mathsf{guess}, pk, s, y) = b\Big] - 1\ .$$

The security of ASYM *is the function*

$$InSec^{\mathrm{CCA2}}(\mathsf{ASYM}\,;\,t,q,\mu,m) \stackrel{\text{def}}{=} \max_{A}\{Adv_{A}^{\mathrm{Asym}}(\mathsf{ASYM})\}\,,$$

where the maximum is taken over all adversaries A running in time t, making at most q queries to its $\mathsf{ASYM.dec}_{sk}$*-oracle, all these totaling at most μ bits, and whose output x_0 (and x_1) has length at most m bits.*

3.3 Privacy against Chosen-Plaintext Attack

We show that DHAES $[\![\mathsf{GROUP},\mathsf{SYM},\mathsf{MAC},\mathsf{H}]\!]$ meets the notion of indistinguishability under a chosen-plaintext attack, as defined in Definition 6.

Theorem 1 *Let* GROUP *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* DHAES *be the asymmetric key encryption scheme associated to these primitives, as defined in Section 1.2. Then for any numbers t, m, and m',*

$$InSec^{\mathrm{Asym}}(\mathsf{DHAES}\,;\,t,m) \;\le\; 2\cdot InSec^{\mathrm{DH}}(\mathsf{GROUP},\mathsf{H}\,;\,t_1) + InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\,t_2,0,m,m')\,,$$

where $t_1 \in O(t+\mathsf{TIME}_{\uparrow}+\mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$ *and* $t_2 \in O(t+\mathsf{TIME}_{\mathsf{SYM.enc}}(m)+\mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$.

The time overhead is tiny; for all practical purposes, t_1 and t_2 can be considered the same as t.

Idea of Proof. The assumption is that the symmetric encryption scheme SYM is secure and H is hardcore 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 $y = U \,\|\, encM \,\|\, tag$ 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 hardcoreness of H on GROUP.

The formal proof 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 of Theorem 1. Let A be an adversary attacking DHAES in the sense of semantic security. Assume it has running time at most t and outputs at the end of its find stage a string of length at most m. We construct an adversary B attacking SYM and an adversary C attacking H being hardcore for GROUP, 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 4 that B has access to an oracle for encryption, and runs in two stages. Notice that B never invokes its encryption oracle $\mathcal{O}$. Moreover, B runs in time $t_1 = t + 2\cdot\mathsf{TIME}_{\uparrow}+\mathsf{TIME}_{\mathsf{MAC.gen}}(m')+mLen+2\cdot gLen = O(t+\mathsf{TIME}_{\uparrow}+\mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$.

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 $H(g^u \,\|\, g^{uv})$ or a random string. C outputs at the end a bit indicating its guess as to which of these cases occurs. Notice that C runs in time $t_2 = O(t+\mathsf{TIME}_{\mathsf{SYM.enc}}(m)+\mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$.

Algorithm $B^{\mathcal{O}}(\mathsf{find})$
begin
  $v \leftarrow \{1, \ldots, |G|\}$
  $pk \leftarrow g^v$
  $(x_0, x_1, s) \leftarrow A(\mathsf{find}, pk)$
  return $(x_0, x_1, (x_0, x_1, s, pk))$
end

Algorithm $B^{\mathcal{O}}(\mathsf{guess}, y', s')$
begin
  parse s' **as** (x_0, x_1, s, pk)
  $u \leftarrow \{1, \ldots, |G|\}$
  $macKey \leftarrow \{0,1\}^{mLen}$
  $tag \leftarrow \mathsf{MAC.gen}_{macKey}(y')$
  $y \leftarrow g^u \parallel y' \parallel tag$
  $b \leftarrow A(\mathsf{guess}, pk, s, y)$
  return b
end

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

Algorithm $C(U, V, W)$
begin
  $macKey \leftarrow W[1 \ldots mLen]$
  $encKey \leftarrow W[mLen + 1 \ldots mLen + eLen]$
  $pk \leftarrow V$
  $(x_0, x_1, s) \leftarrow A(\mathsf{find}, pk)$
  $b' \leftarrow \{0,1\}$
  $encM \leftarrow \mathsf{SYM.enc}_{encKey}(x_{b'})$
  $tag \leftarrow \mathsf{MAC.gen}_{macKey}(encM)$
  $y \leftarrow U \parallel encM \parallel tag$
  $b \leftarrow A(\mathsf{guess}, pk, s, y)$
  if $b = b'$ **then return** 1 **else return** 0
end

Figure 4: *Algorithm C for attacking the hardcoreness of H on* GROUP.

Analysis. When $W = H(g^u \parallel g^{uv})$ we notice that C is running A as the latter would be run in its attack on the semantic security of DHAES. From the definition of $Adv_A^{\mathrm{Asym}}(\mathsf{DHAES})$ we have that

$$\Pr[u, v \leftarrow \{1, \ldots, |G|\};\ W \leftarrow H(g^u \parallel g^{uv}) \ :\ C(g^u, g^v, W) = 1] \ =\ \frac{1}{2} + \frac{Adv_A^{\mathrm{Asym}}(\mathsf{DHAES})}{2}\ .$$

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

$$\Pr[u, v \leftarrow \{1, \ldots, |G|\};\ W \leftarrow \{0,1\}^{hLen} \ :\ C(g^u, g^v, W) = 1] \ =\ \frac{1}{2} + \frac{Adv_B^{\mathrm{Sym}}(\mathsf{SYM})}{2}\ .$$

Subtracting gives us

$$Adv_C^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}) \ =\ \frac{1}{2} + \frac{Adv_A^{\mathrm{Asym}}(\mathsf{DHAES})}{2} - \frac{1}{2} - \frac{Adv_B^{\mathrm{Sym}}(\mathsf{SYM})}{2} \ =\ \frac{Adv_A^{\mathrm{Asym}}(\mathsf{DHAES})}{2} - \frac{Adv_B^{\mathrm{Sym}}(\mathsf{SYM})}{2}\ ;$$

whence

$$Adv_A^{\mathrm{Asym}}(\mathsf{DHAES}) \ =\ 2 \cdot Adv_C^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}) + Adv_B^{\mathrm{Sym}}(\mathsf{SYM})\ .$$

Since the running time of C is at most t_2, we conclude that $Adv_C^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}) \leq InSec^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}\,;\, t_2)$. Moreover, since B makes 0 encryption queries and runs in time at most t_1, we also have $Adv_B^{\mathrm{Sym}}(\mathsf{SYM}) \leq$

$InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m')$. Thus from the above we have

$$Adv_A^{\mathrm{Asym}}(\mathsf{DHAES}) \;\leq\; 2 \cdot InSec^{\mathrm{DH}}(\mathsf{H}, \mathsf{GROUP}\,;\, t_2) + InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m')\ .$$

But A was an arbitrary adversary subject to the constraint that it ran for at most t steps and the length of each of its output messages x_i from its find stage is at most m. The theorem follows. ▮

3.4 Privacy against Non-Adaptive Chosen-Ciphertext Attack

We show that $\mathsf{DHAES}[\![\mathsf{GROUP}, \mathsf{SYM}, \mathsf{MAC}, \mathsf{H}]\!]$ meets the notion of indistinguishability under a non-adaptive chosen-ciphertext attack, as defined in Definition 7.

Theorem 2 *Let* GROUP *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* DHAES *be the asymmetric key encryption scheme associated to these primitives, as defined in Section 1.2. Then for any numbers t, q, μ, m, and m',*

$$\begin{aligned} InSec^{\mathrm{CCA1}}(\mathsf{DHAES}\,;\, t, q, \mu, m) \quad \leq \quad & InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m') + 2 \cdot InSec^{\mathrm{DHI1}}(\mathsf{H}, \mathsf{GROUP}\,;\, t_2, q) + \\ & q \cdot 2^{-(gLen + hLen) + 1}\ , \end{aligned}$$

where $t_1 \in O(t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$ and $t_2 \in O(t + \mathsf{TIME}_{\mathsf{SYM.enc}}(m) + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + q \cdot (hLen + gLen))$.

Idea of Proof. The assumption is that the symmetric encryption scheme SYM is secure and H is hardcore for the Diffie-Hellman problem in the underlying group under non-adaptive DH attack. The proof considers an adversary A who defeats the non-adaptive chosen-ciphertext security of the scheme. Let g^v be the recipient public key and let $y = U \,\|\, encM \,\|\, tag$ 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 does not look random.* In this case, we present an algorithm C that breaks the hardcoreness of H on GROUP under non-adaptive DH attack.
- *Case 2 — The output of H looks random.* In this case, we present an adversary B that breaks the encryption scheme SYM.

As in the proof of Theorem 1, given A, we construct B and C and then relate A's advantage to that of B and C.

Proof of Theorem 2. Let A be an adversary attacking DHAES in the sense of non-adaptive chosen-ciphertext security. Assume it has running time at most t, makes at most q queries to its decryption oracle, and outputs at the end of its find stage a string of length at most m. We construct an adversary B attacking SYM and an adversary C attacking H being hardcore for GROUP under non-adaptive DH attack, and then upper bound the advantage of A in terms of the advantages of these adversaries.

Algorithm B. Figure 5 shows algorithm B. Recall from Definition 4 that B has access to an oracle for encryption and runs in two stages. Notice that B never invokes its encryption oracle $\mathcal{O}$. Moreover, since A's running time accounts for the time taken by decryption queries, also notice that B's running time is $t_1 = t + 2 \cdot \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + mLen + 2 \cdot gLen = O(t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$.

Algorithm C. Figure 6 defines the behavior of algorithm C. Recall from Definition 2 that C runs in two stages. In the find stage, C is given as input $V = g^v$ for a random v and outputs some state information that should be carried on to the guess stage. C is also granted access to a HDH oracle HDH_v in this stage. In the guess stage, in addition to the state information, C is given as input U and W, where $U = g^u$ for a

Algorithm $B^{\mathcal{O}}(\mathsf{find})$
begin
  $v \leftarrow \{1, \ldots, |G|\}$
  $pk \leftarrow g^v$
  run A on input (find, pk)
  – For each decryption query y_i
    parse y_i **as** $U_i \parallel encM_i \parallel tag_i$
    $hash_i \leftarrow H(U_i \parallel U_i^v)$
    $macKey_i \leftarrow hash_i[1..mLen]$
    $encKey_i \leftarrow hash_i[mLen+1..mLen+eLen]$
    if $\mathsf{MAC.ver}_{macKey_i}(encM_i, tag_i) = 1$ **then**
      return $\mathsf{SYM.dec}_{encKey_i}(encM_i)$
    else return BAD
  – Let (x_0, x_1, s) be the output of A
  return $(x_0, x_1, (x_0, x_1, s, pk))$
end

Algorithm $B^{\mathcal{O}}(\mathsf{guess}, y', s')$
begin
  parse s' **as** (x_0, x_1, s, pk)
  $u \leftarrow \{1, \ldots, |G|\}$
  $U \leftarrow g^u$
  $macKey \leftarrow \{0,1\}^{mLen}$
  $tag \leftarrow \mathsf{MAC.gen}_{macKey}(y')$
  $y \leftarrow U \parallel y' \parallel tag$
  $b \leftarrow A(\mathsf{guess}, pk, s, y)$
  return b
end

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

Algorithm $C^{\mathrm{HDH}_v(\cdot)}(\mathsf{find}, V)$
begin
  $pk \leftarrow V$
  Hlist $\leftarrow \{\}$
  run A on input (find, pk)
  – For each decryption query y_i
    parse y_i **as** $U_i \parallel encM_i \parallel tag_i$
    $hash_i \leftarrow \mathrm{HDH}_v(U_i)$
    Hlist $\leftarrow$ **Hlist** $\cup \{(U_i, hash_i)\}$
    $macKey_i \leftarrow hash_i[1..mLen]$
    $encKey_i \leftarrow hash_i[mLen+1..mLen+eLen]$
    if $\mathsf{MAC.ver}_{macKey_i}(encM_i, tag_i) = 1$ **then**
      return $\mathsf{SYM.dec}_{encKey_i}(encM_i)$
    else return BAD
  – Let (x_0, x_1, s) be the output of A
  return $(x_0, x_1, s, pk,$ **Hlist**$)$
end

Algorithm $C(\mathsf{guess}, s', U, W)$
begin
  parse s' **as** $(x_0, x_1, s, pk,$ **Hlist**$)$
  if $\exists\ (U_i, hash_i) \in$ **Hlist** $: U_i = U$ **then**
    if $W = hash_i$ **then return** 1
    else return 0
  $macKey \leftarrow W[1 \ldots mLen]$
  $encKey \leftarrow W[mLen+1 \ldots mLen+eLen]$
  $b' \leftarrow \{0,1\}$
  $encM \leftarrow \mathsf{SYM.enc}_{encKey}(x_{b'})$
  $tag \leftarrow \mathsf{MAC.gen}_{macKey}(encM)$
  $y \leftarrow U \parallel encM \parallel tag$
  $b \leftarrow A(\mathsf{guess}, pk, s, y)$
  if $b = b'$ **then return** 1
  else return 0
end

Figure 6: *Algorithm C for attacking the hardcoreness of H on* GROUP *under non-adaptive DH attack.*

random u and W is either $H(g^u \parallel g^{uv})$ or a random string. At the end, C outputs a bit indicating its guess as to which of these cases occurs.

Notice that, since A's running time accounts for the time taken by decryption queries, the queries to the HDH oracle HDH_v do not incur any extra time in the computation. However, we do need to account for the time taken to handle **Hlist**. Therefore, C's running time is $t_2 = O(t + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + \mathsf{TIME}_{\mathsf{SYM.enc}}(m) + q \cdot (hLen + gLen))$.

ANALYSIS. Call Experiment 1 the following:

$$(pk, sk) \leftarrow \mathsf{DHAES.key};\ \ (x_0, x_1, s) \leftarrow A^{\mathsf{DHAES}.\mathsf{dec}_{sk}}(\mathsf{find}, pk);$$

$$b \leftarrow \{0,1\};\ \ y \leftarrow \mathsf{DHAES}.\mathsf{enc}_{pk}(x_b);\ \ b' \leftarrow A(\mathsf{guess}, pk, s, y)\ .$$

and let $\Pr_1[\,\cdot\,]$ denote the probabilities in this experiment. Parse the challenge ciphertext y into $U \parallel encM \parallel tag$ and let u and v be defined by $g^v = pk$ and $g^u = U$, respectively. Let us call a Type 1 query a ciphertext of the form $U \parallel encM' \parallel tag'$. A Type 2 query have the form $U' \parallel encM' \parallel tag'$ with $U' \neq U$. We consider the following events in Experiment 1:

$$\begin{array}{rcl} \text{SUCCA} & : & b = b' \\ \text{ASKA} & : & A \text{ makes a Type 1 query } y' \text{ to its decryption oracle} \end{array}$$

Our goal is to upper bound $\Pr_1[\,\text{SUCCA}\,]$ and, consequently, $Adv_A^{\text{CCA1}}(\mathsf{DHAES})$. For this purpose, we make use of the following three claims.

Claim 3

$$\Pr[\,u, v \leftarrow \{1, \ldots, |G|\};\ s \leftarrow C^{\text{HDH}_v(\cdot)}(\mathsf{find}, g^v);\ W \leftarrow H(g^u \parallel g^{uv}) :$$
$$C(\mathsf{guess}, s, g^u, W) = 1\,] \geq \frac{1}{2} + \frac{Adv_A^{\text{CCA1}}(\mathsf{DHAES})}{2}\ .$$

Proof: When $W = H(g^u \parallel g^{uv})$ and $\overline{\text{ASKA}}$, we notice that C is running A as the latter would be run in its attack on the non-adaptive chosen-ciphertext security of DHAES. On the other hand, when ASKA and $W = H(g^u \parallel g^{uv})$, C always outputs 1. Thus, we have that

$$\begin{array}{rcl} \Pr[\,u, v \leftarrow \{1, \ldots, |G|\};\ s \leftarrow C^{\text{HDH}_v(\cdot)}(\mathsf{find}, g^v);\ W \leftarrow H(g^u \parallel g^{uv})\ :\ C(\mathsf{guess}, s, g^u, W) = 1\,] & & \\ = \quad \Pr_1[\,\text{SUCCA} \mid \overline{\text{ASKA}}\,] \cdot \Pr_1[\,\overline{\text{ASKA}}\,] + 1 \cdot \Pr_1[\,\text{ASKA}\,] & & \\ \geq \quad \Pr_1[\,\text{SUCCA} \mid \overline{\text{ASKA}}\,] \cdot \Pr_1[\,\overline{\text{ASKA}}\,] + \Pr_1[\,\text{SUCCA} \mid \text{ASKA}\,] \cdot \Pr_1[\,\text{ASKA}\,] & & \\ = \quad \Pr_1[\,\text{SUCCA}\,]\ . & & \end{array}$$

The claim follows from Definition 7. ∎

Claim 4

$$\Pr[\,u, v \leftarrow \{1, \ldots, |G|\};\ s \leftarrow C^{\text{HDH}_v(\cdot)}(\mathsf{find}, g^v);\ W \leftarrow \{0,1\}^{hLen} :$$
$$C(\mathsf{guess}, s, g^u, W) = 1 \wedge \overline{\text{ASKA}}\,] \leq \frac{1}{2} + \frac{Adv_B^{\text{Sym}}(\mathsf{SYM})}{2}\ .$$

Proof: Call Experiment 2 the following:

$$K \leftarrow \mathsf{Key};\ \ (x_0, x_1, s) \leftarrow B^{\mathcal{O}}(\mathsf{find});\ \ b \leftarrow \{0,1\};\ \ y \leftarrow \mathcal{O}(x_b);\ \ b' \leftarrow B^{\mathcal{O}}(\mathsf{guess}, y, s)\ ;$$

and let $\Pr_2[\,\cdot\,]$ denote the probabilities in this experiment. We consider SUCCB to be the event $b = b'$ in Experiment 2.

When W is a random string and $\overline{\text{ASKA}}$, we notice that C runs A in the same way as B does. Notice that the definition of event ASKA (and, therefore, $\overline{\text{ASKA}}$) can also be applied in Experiment 2 since B runs A as

a subroutine. Moreover, since A is considered an arbitrary adversary subject to the same constraints in all experiments, the probability for event AskA to happen is the same in all these experiments. Therefore,

$$\begin{aligned}
&\Pr[u,v\leftarrow\{1,\ldots,|G|\};\ s\leftarrow C^{\mathrm{HDH}_v(\cdot)}(\mathsf{find},g^v);\ W\leftarrow\{0,1\}^{hLen}\ :\ C(\mathsf{guess},s,g^u,W)=1\wedge\overline{\textsc{AskA}}]\\
&\quad=\quad \Pr\left[u,v\leftarrow\{1,\ldots,|G|\};\ W\leftarrow\{0,1\}^{hLen}\ :\ W=H(g^u\parallel g^{uv})\ \mid\ \textsc{AskA}\right]\cdot\Pr_1\left[\textsc{AskA}\right]\\
&\quad=\quad \Pr_2\left[\textsc{SuccB}\ \mid\ \overline{\textsc{AskA}}\right]\cdot\Pr_2\left[\overline{\textsc{AskA}}\right]\\
&\quad\leq\quad \Pr_2\left[\textsc{SuccB}\right].
\end{aligned}$$

The claim follows from Definition 4. ∎

Claim 5

$$\begin{aligned}
\Pr[u,v\leftarrow\{1,\ldots,|G|\};\ s\leftarrow C^{\mathrm{HDH}_v(\cdot)}(\mathsf{find},g^v);\ W\leftarrow\{0,1\}^{hLen}:\\
C(\mathsf{guess},s,g^u,W)=1\wedge\textsc{AskA}]\leq 2^{-(gLen+hLen)}\ .
\end{aligned}$$

Proof: When W is a random string and AskA, C outputs 1 only if $W=H(g^u\parallel g^{uv})$. Since all decryption queries are made before A sees the challenge ciphertext, $\Pr_1[\textsc{AskA}]\leq q\cdot 2^{-gLen}$. The claim follows from the fact that $\Pr[u,v\leftarrow\{1,\ldots,|G|\};\ W\leftarrow\{0,1\}^{hLen}\ :\ W=H(g^u\parallel g^{uv})]\ =\ 2^{-hLen}$. ∎

From Definition 2 and Claims 3, 4, and 5, we have that:

$$\begin{aligned}
Adv_C^{\mathrm{DHI1}}(\mathsf{H},\mathsf{GROUP}) &\geq \frac{1}{2}+\frac{Adv_A^{\mathrm{CCA1}}(\mathsf{DHAES})}{2}-\frac{1}{2}-\frac{Adv_B^{\mathrm{Sym}}(\mathsf{SYM})}{2}-q\cdot 2^{-(gLen+hLen)}\\
&= \frac{Adv_A^{\mathrm{CCA1}}(\mathsf{DHAES})}{2}-\frac{Adv_B^{\mathrm{Sym}}(\mathsf{SYM})}{2}-q\cdot 2^{-(gLen+hLen)}\ ;
\end{aligned}$$

whence

$$Adv_A^{\mathrm{CCA1}}(\mathsf{DHAES})\leq Adv_B^{\mathrm{Sym}}(\mathsf{SYM})+2\cdot Adv_C^{\mathrm{DHI1}}(\mathsf{H},\mathsf{GROUP})+q\cdot 2^{-(gLen+hLen)+1}\ .$$

Since B runs in time at most t_1 and makes 0 encryption queries, $Adv_B^{\mathrm{Sym}}(\mathsf{SYM})\leq InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\,t_1,0,m,m')$. As C runs in time at most t_2 and makes at most q queries to its HDH oracle, $Adv_C^{\mathrm{DHI1}}(\mathsf{H},\mathsf{GROUP})\leq InSec^{\mathrm{DHI1}}(\mathsf{H},\mathsf{GROUP}\,;\,t_2,q)$. Thus, from the above, we have

$$Adv_A^{\mathrm{CCA1}}(\mathsf{DHAES})\ \leq\ InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\,t_1,0,m,m')+2\cdot InSec^{\mathrm{DHI1}}(\mathsf{H},\mathsf{GROUP}\,;\,t_2,q)+q\cdot 2^{-(gLen+hLen)+1}\ .$$

But A was an arbitrary adversary subject to the constraint that it ran for at most t steps and the length of each of its output messages x_i from its find stage is at most m. The theorem follows. ∎

3.5 Privacy against Adaptive Chosen-Ciphertext Attack

We show that $\mathsf{DHAES}[\![\mathsf{GROUP},\mathsf{SYM},\mathsf{MAC},\mathsf{H}]\!]$ meets the notion of indistinguishability under an adaptive chosen-ciphertext attack, as in Definition 8.

Theorem 6 *Let* $\mathsf{GROUP}=(G,g,\_,\uparrow)$ *be a represented group, let* SYM *be a symmetric encryption scheme, and let* MAC *be a message authentication scheme. Let* DHAES *be the asymmetric encryption scheme associated to these primitives as defined in Section 1.2. Then for any numbers* t,q,μ,m, *and* m',

$$\begin{aligned}
InSec^{\mathrm{CCA2}}(\mathsf{DHAES}\,;\,t,q,\mu,m)\ \leq\ & InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\,t_1,0,m,m')+2\cdot InSec^{\mathrm{DHI2}}(\mathsf{H},\mathsf{GROUP}\,;\,t_2,q)+\\
& 2\cdot q\cdot InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\,t_3,q-1)\ ,
\end{aligned}$$

where $t_1 \in O(t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$, $t_2 \in O(t + \mathsf{TIME}_{\mathsf{SYM.enc}}(m) + \mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$, *and* $t_3 \in O(t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + \mathsf{TIME}_{\mathsf{SYM.enc}}(m) + q)$.

IDEA OF PROOF. The assumption is that both symmetric encryption scheme SYM and the message authentication scheme MAC are secure and H is hardcore for the Diffie-Hellman problem on GROUP under adaptive DH 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 $y = U \parallel encM \parallel tag$ be the challenge ciphertext that algorithm A gets in its guess stage. Let Type 1 and Type 2 queries be defined as in Section 3.4. We consider three cases depending on whether the output of H looks random and on whether there was a Type 1 query y' to the decryption oracle $\mathsf{DHAES.dec}_{sk}$ such that $\mathsf{DHAES.dec}_{sk}(y') \neq \mathsf{BAD}$.

- *Case 1 — The output of H does not look random.* In this case we present an algorithm C that breaks the hardcoreness of H on GROUP under adaptive DH attack.
- *Case 2 — The output of H looks random and there was a* Type 1 *query y' to* $\mathsf{DHAES.dec}_{sk}$ *such that* $\mathsf{DHAES.dec}_{sk}(y') \neq \mathsf{BAD}$. 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 y' to* $\mathsf{DHAES.dec}_{sk}$ *such that* $\mathsf{DHAES.dec}_{sk}(y') \neq \mathsf{BAD}$. In this case we present an adversary B which breaks the encryption scheme SYM.

PROOF OF THEOREM 6. Let A be an adversary attacking DHAES 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, and outputs at the end of its find stage a string of length at most m. We construct an adversary B attacking SYM, an adversary C attacking H being hardcore for GROUP under non-adaptive DH 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. Figure 7 describes algorithm B. Recall from Definition 4 that B has access to an oracle for encryption and runs in two stages. Since A's running time accounts for the time taken by decryption queries, notice that B runs in time $t_1 = t + 2 \cdot \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + mLen + 2 \cdot gLen = O(t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m'))$.

ALGORITHM C. Figure 8 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 $H(g^u \parallel g^{uv})$ or a random string. Recall from Definition 3 that C is also given access to a HDH oracle HDH_v. At the end, C outputs a bit indicating its guess as to which of these cases occurs.

Notice that, since A's running time accounts for the time taken by decryption queries, the queries to the HDH oracle HDH_v do not incur any extra time in the computation. As a result, C's running time is $t_2 = O(t + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + \mathsf{TIME}_{\mathsf{SYM.enc}}(m))$.

ALGORITHM F. Figure 9 describes algorithm F. Recall from Definition 5 that F has access to a tag-generation oracle $\mathcal{O}$ and outputs a pair message-tag, a possible forgery. Notice that, since A's running time accounts for the time taken by decryption queries, the queries to the oracle $\mathcal{O}$ due to these decryption queries do not incur any extra time in the computation. Consequently, F's running time is $t_3 = t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + \mathsf{TIME}_{\mathsf{SYM.enc}}(m) + eLen + 2 \cdot gLen + \lg(q) + q \cdot O(1) = O(t + \mathsf{TIME}_{\uparrow} + \mathsf{TIME}_{\mathsf{MAC.gen}}(m') + \mathsf{TIME}_{\mathsf{SYM.enc}}(m) + q)$.

ANALYSIS. Call Experiment 1 the following:

$$(pk, sk) \leftarrow \mathsf{DHAES.key};\ \ (x_0, x_1, s) \leftarrow A^{\mathsf{DHAES.dec}_{sk}}(\mathsf{find}, pk);$$

Algorithm $B^{\mathcal{O}}(\mathsf{find})$
begin
  $v \leftarrow \{1,\ldots,|G|\}$
  $pk \leftarrow g^v$
  run $A(\mathsf{find}, pk)$
  – For each decryption query y_i
    parse y_i **as** $U_i \parallel encM_i \parallel tag_i$
    $hash_i \leftarrow H(U_i \parallel U_i^v)$
    $macKey_i \leftarrow hash_i[1..mLen]$
    $encKey_i \leftarrow hash_i[mLen+1..mLen+eLen]$
    if $\mathsf{MAC.ver}_{macKey_i}(encM_i, tag_i) = 1$ **then**
      return $\mathsf{SYM.dec}_{encKey_i}(encM_i)$
    else return BAD
  – Let (x_0, x_1, s) be the output of A
  return $(x_0, x_1, (x_0, x_1, s, v, pk))$
end

Algorithm $B^{\mathcal{O}}(\mathsf{guess}, y', s')$
begin
  parse s' **as** (x_0, x_1, s, v, pk)
  ASK $\leftarrow$ false
  $u \leftarrow \{1,\ldots,|G|\}$
  $U \leftarrow g^u$
  $macKey \leftarrow \{0,1\}^{mLen}$
  $tag \leftarrow \mathsf{MAC.gen}_{macKey}(y')$
  $y \leftarrow U \parallel y' \parallel tag$
  run $A(\mathsf{guess}, pk, s, y)$
  – For each decryption query y_i
    parse y_i **as** $U_i \parallel encM_i \parallel tag_i$
    $hash_i \leftarrow H(U_i \parallel U_i^v)$
    $macKey_i \leftarrow hash_i[1..mLen]$
    $encKey_i \leftarrow hash_i[mLen+1..mLen+eLen]$
    if $\mathsf{MAC.ver}_{macKey_i}(encM_i, tag_i) = 1$ **then**
      if $U_i \neq U$ **then**
        return $\mathsf{SYM.dec}_{encKey_i}(encM_i)$
      else ASK $\leftarrow$ true;
    return BAD
  – **if** ASK = true **then** $b \leftarrow \{0,1\}$
    else let b be the output of A
  return b
end

Figure 7: *Algorithm B for attacking the security of* SYM.

$$b \leftarrow \{0,1\};\ y \leftarrow \mathsf{DHAES.enc}_{pk}(x_b);\ b' \leftarrow A^{\mathsf{DHAES.dec}_{sk}}(\mathsf{guess}, pk, s, y)\ .$$

and let $\Pr_1[\cdot]$ denote the probabilities in this experiment. Let $v = sk$ be defined by $g^v = pk$. We consider the following events:

$$\begin{array}{rcl} \textsc{SuccA} & : & b = b' \\ \textsc{SomeValid} & : & \text{There was a Type 1 query } y' \text{ such that } \mathsf{DHAES.dec}_{sk}(y') \neq \mathsf{BAD} \end{array}$$

As in the proof of non-adaptive chosen-ciphertext security, our goal is to upper bound $\Pr_1[\textsc{SuccA}]$ and, consequently, $Adv_A^{\mathrm{CCA2}}(\mathsf{DHAES})$. For this purpose, we make use of the following three claims.

Claim 7

$$\Pr[u, v \leftarrow \{1,\ldots,|G|\};\ W \leftarrow H(g^u \parallel g^{uv})\ :\ C^{\mathrm{HDH}_v(\cdot)}(g^u, g^v, W) = 1] \ =\ \frac{1}{2} + \frac{Adv_A^{\mathrm{CCA2}}(\mathsf{DHAES})}{2}\ .$$

Proof: When $W = H(g^u \parallel g^{uv})$, we notice that C is running A as the latter would be run in its attack on the adaptive chosen-ciphertext security of DHAES. Therefore, the claim follows from the definition of $Adv_A^{\mathrm{CCA2}}(\mathsf{DHAES})$. ∎

Claim 8

$$\Pr[u, v \leftarrow \{1,\ldots,|G|\};\ W \leftarrow \{0,1\}^{hLen} : C^{\mathrm{HDH}_v(\cdot)}(g^u, g^v, W) = 1 \wedge \overline{\textsc{SomeValid}}]$$
$$\leq \tfrac{1}{2} + \tfrac{InSec^{\mathrm{Sym}}(\mathsf{SYM}; t_1, 0, m, m')}{2}\ .$$

Algorithm $C^{\mathrm{HDH}_v(\cdot)}(U, V, W)$
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
  $b' \leftarrow \{0,1\}$
  $encM \leftarrow \mathsf{SYM.enc}_{encKey}(x_{b'})$
  $tag \leftarrow \mathsf{MAC.gen}_{macKey}(encM)$
  $y \leftarrow U \parallel encM \parallel tag$
  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 $b = b'$ **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
    $hash_i \leftarrow \mathrm{HDH}_v(U_i)$
    $macKey_i \leftarrow hash_i[1..mLen]$
    $encKey_i \leftarrow hash_i[mLen + 1..mLen + eLen]$
  if $\mathsf{MAC.ver}_{macKey_i}(encM_i, tag_i) = 1$ **then**
    return $\mathsf{SYM.dec}_{encKey_i}(encM_i)$
  else return BAD
end

Figure 8: *Algorithm C for attacking the hardcoreness of H on* GROUP *under adaptive DH attack.*

Algorithm $F^{\mathcal{O}}$
begin
  $v \leftarrow \{1, \ldots, |G|\}$; $pk \leftarrow g^v$
  $u \leftarrow \{1, \ldots, |G|\}$; $U \leftarrow g^u$
  $encKey \leftarrow \{0,1\}^{eLen}$
  $i \leftarrow 0$
  $j \leftarrow \{1, \ldots, q\}$
  run $A(\mathsf{find}, pk)$
  – For each decryption query y'
    return Decr-Simulator(y')
  – Let (x_0, x_1, s) be the output of A
  $b' \leftarrow \{0,1\}$
  $encM \leftarrow \mathsf{SYM.enc}_{encKey}(x_{b'})$
  $tag \leftarrow \mathcal{O}(encM)$
  $y \leftarrow U \parallel encM \parallel tag$
  run $A(\mathsf{guess}, pk, s, y)$
  – For each decryption query y'
    return Decr-Simulator(y')
  – Let b be the output of A
  return W
end

Subroutine Decr-Simulator(y')
begin
  parse y' **as** $U' \parallel encM' \parallel tag'$
  $hash' \leftarrow H(U' \parallel U'^v)$
  $macKey' \leftarrow hash'[1..mLen]$
  $encKey' \leftarrow hash'[mLen + 1..mLen + eLen]$
  $i \leftarrow i + 1$
  if $i \neq j$ **then**
    if $U' = U$ **then**
      if $\mathcal{O}(encM') = tag'$ **then**
        return $\mathsf{SYM.dec}_{encKey}(encM')$
      else return BAD
    else
      if $\mathsf{MAC.ver}(macKey', encM', tag') = 1$ **then**
        return $\mathsf{SYM.dec}_{encKey'}(encM')$
      else return BAD
  else
    $W \leftarrow (encM', tag')$
    return $\mathsf{SYM.dec}_{encKey}(encM')$
end

Figure 9: *Algorithm F for attacking the security of* MAC.

Proof: When A does not make a Type 1 query to its decryption oracle nor makes a Type 1 query y' such that $\overline{\mathsf{DHAES}.\mathsf{dec}_{sk}(y') \neq \mathsf{BAD}}$, C runs A in the same way B does. Hence, the probability that C outputs 1 given $\overline{\textsc{SomeValid}}$ is at most $1/2 + Adv_B^{\mathrm{Sym}}(\mathsf{SYM})/2$. Since B makes 0 encryption queries and runs in time at most t_1, the claim follows directly from the assumed security of SYM. ∎

Claim 9

$$\Pr[C^{\mathrm{HDH}_v(\cdot)}(g^u, g^v, W) = 1 \wedge \textsc{SomeValid} : u, v \leftarrow \{1, \ldots, |G|\};\ W \leftarrow \{0,1\}^{hLen}] \leq q \cdot InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t_3, q-1)\ .$$

Proof: When there is a Type 1 query y' to the decryption oracle such that $\mathsf{DHAES}.\mathsf{dec}_{sk}(y') \neq \mathsf{BAD}$, let i be the number of one such query and let $y_i = U \parallel encM_i \parallel tag_i$ be its value. If $j \in \{1, \ldots, q\}$ in algorithm F takes this value, then F succeeds in breaking MAC since $(encM_i, tag_i)$ is a valid pair. Because this can happen with probability at least $1/q$, $\Pr_1[\textsc{SomeValid}] \leq q \cdot Succ_F^{\mathrm{MAC}}(\mathsf{MAC})$. Hence, since F runs in time at most t_3 and makes at most $q-1$ queries to its oracle, $\Pr_1[\textsc{SomeValid}] \leq InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t_3, q-1)$ due to the assumed security of MAC. The claim follows from basic probability properties. ∎

From Definition 3 and Claims 7, 8, and 9, we have that:

$$\begin{aligned} Adv_C^{\mathrm{DHI2}}(\mathsf{H}, \mathsf{GROUP}) &\geq \frac{1}{2} + \frac{Adv_A^{\mathrm{CCA2}}(\mathsf{DHAES})}{2} - \frac{1}{2} - \frac{InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m')}{2} - \\ &\quad q \cdot InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t_3, q-1) \\ &= \frac{Adv_A^{\mathrm{CCA1}}(\mathsf{DHAES})}{2} - \frac{InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m')}{2} - q \cdot InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t_3, q-1)\ ; \end{aligned}$$

whence

$$Adv_A^{\mathrm{CCA2}}(\mathsf{DHAES}) \leq InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m') + 2 \cdot Adv_C^{\mathrm{DHI2}}(\mathsf{H}, \mathsf{GROUP}) + 2 \cdot q \cdot InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t_3, q-1)\ .$$

We conclude that, since C runs in time at most t_2 and makes at most q to its HDH oracle, $Adv_C^{\mathrm{DHI2}}(\mathsf{H}, \mathsf{GROUP}) \leq InSec^{\mathrm{DHI2}}(\mathsf{H}, \mathsf{GROUP}\,;\, t_2, q)$. Thus, from the above, we have

$$\begin{aligned} Adv_A^{\mathrm{CCA2}}(\mathsf{DHAES}) &\leq InSec^{\mathrm{Sym}}(\mathsf{SYM}\,;\, t_1, 0, m, m') + 2 \cdot InSec^{\mathrm{DHI2}}(\mathsf{H}, \mathsf{GROUP}\,;\, t_2, q) + \\ &\quad 2 \cdot q \cdot InSec^{\mathrm{MAC}}(\mathsf{MAC}\,;\, t_3, q-1)\ . \end{aligned}$$

But A was an arbitrary adversary subject to the constraint that it ran for at most t steps and the length of each of its output messages x_i from its find stage is at most m. The theorem follows. ∎

4 Limitations

None known.

5 Intellectual Property Statement

Neither the University of California nor any of the authors has any patents or patent applications relevant to this proposal. Our contributions in this domain have been been placed entirely in the public domain.

References

[1] 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, Working draft version 2.0, July 5, 1998.

[2] M. Bellare, R. Canetti and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[3] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[4] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation. Current version available at URL of first author. Preliminary version in *Proceedings of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.

[5] M. Bellare, J. Kilian and P. Rogaway, The security of cipher block chaining. *Advances in Cryptology – Crypto* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

[6] M. Bellare and P. Rogaway, Minimizing the use of random oracles in authenticated encryption schemes. *Information and Communications Security,* Lecture Notes in Computer Science, vol. 1334, Springer-Verlag, 1997, pp. 1–16.

[7] M. Bellare and P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of the First Annual Conference on Computer and Communications Security*, ACM, 1993.

[8] M. Bellare and P. Rogaway, Optimal asymmetric encryption– How to encrypt with RSA. Current version available at URL of either author. Preliminary version in *Advances in Cryptology – Eurocrypt* 94 *Proceedings*, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.

[9] M. Bellare and P. Rogaway, The exact security of digital signatures– How to sign with RSA and Rabin. Current version available at URL of either author. Preliminary version in *Advances in Cryptology – Eurocrypt* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.

[10] D. Bleichenbacher, A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1. *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[11] D. Boneh, The decision Diffie-Hellman problem. Invited paper for the *Third Algorithmic Number Theory Symposium (ANTS)*, Lecture Notes in Computer Science Vol. 1423, Springer-Verlag, 1998.

[12] D. Boneh and R. Lipton, Algorithms for black-box fields and their application to cryptography. *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[13] D. Boneh and R. Venkatesan, Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[14] R. Canetti, O. Goldreich and S. Halevi, The random oracle methodology, revisited. *Proceedings of the* 30th *Annual Symposium on Theory of Computing*, ACM, 1998.

[15] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[16] W. Diffie and M. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory*, 22, pp. 644–654, 1976.

[17] D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. *Proceedings of the* 23rd *Annual Symposium on Theory of Computing*, ACM, 1991.

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

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

[20] O. Goldreich, A uniform complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, vol. 6, 1993, pp. 21-53.

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

[22] S. Hada and T. Tanaka, On the Existence of 3-Round Zero-Knowledge Protocols. *Advances in Cryptology – Crypto* 98 *Proceedings*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[23] IEEE P1363 Committee, IEEE P1363 / D4 (Draft version 4) — Standard specifications for public-key cryptography. June 1998. See http://grouper.ieee.org/groups/1363/index.html/

[24] D. Johnson and S. Matyas. Asymmetric encryption: evolution and enhancements. RSA's CryptoBytes, Vol. 2, No. 1, Spring 1996.

[25] D. Johnson, A. Lee, W. Martin, S. Matyas and J. Wilkins. Hybrid key distribution scheme giving key record recovery. IBM Technical Disclosure Bulletin, 37(2A), 5–16, February 1994.

[26] D. Johnson, S. Matyas, M. Peyravian, Encryption of long blocks using a short-block encryption procedure. November 1996. Available in http://stdsbbs.ieee.org/groups/1363/index.html.

[27] H. Krawczyk, M. Bellare, and R. Canetti, HMAC: Keyed-Hashing for Message Authentication. Internet RFC 2104, February 1997.

[28] C. Lim and P. Lee, Another method for attaining security against adaptively chosen ciphertext attacks. *Advances in Cryptology – Crypto* 93 *Proceedings*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.

[29] S. Matyas, M. Peyravian, A. Roginsky, Security analysis of Feistel ladder formatting procedure. March 1997. Available in http://stdsbbs.ieee.org/groups/1363/index.html.

[30] U. Maurer and S. Wolf, Diffie-Hellman oracles. *Advances in Cryptology – Crypto* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.

[31] S. Micali, C. Rackoff and B. Sloan, The notion of security for probabilistic cryptosystems. *SIAM J. of Computing*, April 1988.

[32] M. Naor and O. Reingold, Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Proceedings of the* 38th *Symposium on Foundations of Computer Science*, IEEE, 1997.

[33] M. Naor and M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proceedings of the* 22nd *Annual Symposium on Theory of Computing*, ACM, 1990.

[34] National Institute of Standards, FIPS 180-1, Secure hash standard. April 1995.

[35] D. Pointcheval and J. Stern, Security proofs for signatures. *Advances in Cryptology – Eurocrypt* 96 *Proceedings*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.

[36] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *Advances in Cryptology – Crypto* 91 *Proceedings*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[37] M. Wegman and L. Carter, New hash functions and their use in authentication and set equality. *J. of Computer and System Sciences* 22, 265–279 (1981).

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

[39] Y. Zheng and J. Seberry, Immunizing public key cryptosystems against chosen ciphertext attack. *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, 715–724 (1993).

A Attacks on the ElGamal Scheme

One of the main reasons for building *DHAES* is that one usually wants a scheme that not only shields information about the plaintext in the presence of a passive attack, but also achieves stronger goals such as non-malleability and chosen-ciphertext security. One main problem with ElGamal scheme is that it fails to achieve these stronger security notions 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 GROUP, 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 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 $encM'$, then the resulting ciphertext $EM' = (g^u, encM)$ will be an encryption for a message $M' = M \cdot g^k$ 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 $encM'$ be a point in G such that $encM' \neq encM$ and let M' be the decryption of $EM' = (g^u, encM')$. As we know that $M' = encM'/g^{uv}$, we can compute g^{uv} and then $encM/g^{uv}$, which is the decryption of EM.