The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES

Michel Abdalla[1], Mihir Bellare[1], and Phillip Rogaway[2]

[1] Department of Computer Science & Engineering
University of California at San Diego, La Jolla, California, 92093, USA.
{mabdalla,mihir}@cs.ucsd.edu.
URLs:www-cse.ucsd.edu/users/mabdalla,mihir}.
[2] Department of Computer Science
University of California at Davis, Davis, California 95616, USA.
rogaway@cs.ucsd.edu.
URL:http://www.cs.ucdavis.edu/~rogaway

Abstract. This paper provides security analysis for the public-key encryption scheme DHIES (formerly named DHES and DHAES), which was proposed in [7] and is now in several draft standards. DHIES is a Diffie-Hellman based scheme that combines a symmetric encryption method, a message authentication code, and a hash function, in addition to number-theoretic operations, in a way which is intended to provide security against chosen-ciphertext attacks. In this paper we find natural assumptions under which DHIES achieves security under chosen-ciphertext attack. The assumptions we make about the Diffie-Hellman problem are interesting variants of the customary ones, and we investigate relationships among them, and provide security lower bounds. Our proofs are in the standard model; no random-oracle assumption is required.

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

1 Introduction

DHIES is an extension of the ElGamal encryption scheme. It was suggested in [7] and is now in the draft standards of ANSI X9.63, SECG, and IEEE P1363a [2,12,22]. In this paper we prove the security of DHIES against chosen-ciphertext attacks based on some new variants of the Diffie-Hellman assumption. (We do not appeal to the random-oracle model.) We then look at relationship of the new Diffie-Hellman assumptions to standard ones, and prove a complexity lower bound, in the generic model, about one of them.

Background. The name DHIES stands for "Diffie-Hellman Integrated Encryption Scheme." It is "integrated" in the sense of using several different tools, including private-key and public-key encryption primitives. The scheme was formerly known as DHES and as DHAES. It is all the same scheme. DHIES was designed to be a natural extension of the ElGamal scheme, suitable in a variety of groups, and which enhanced ElGamal in a couple of ways important to cryptographic practice. First, the scheme needed to provide the capability of encrypting arbitrary bit strings (ElGamal requires that message be a group element). And second, the scheme should be secure against chosen-ciphertext attack (ElGamal is not). The above two goal had to be realized without increasing the number of group operations for encryption and decryption, and without increasing key sizes relative to ElGamal. Within these constraints, the designers wanted to provide the best possible provable-security analysis. But efficiency and practicality of the scheme could not be sacrificed in order to reduce assumptions.

The DHIES scheme uses a hash function. In [7] a claim is made that DHIES should achieve plaintext awareness if this hash function is modeled as a public random oracle and one assumes the computational DH assumption. In fact, technical problems would seem to thwart any possibility of pushing through such a result.

Our approach. Our main goal has been to provide a security analysis of DHIES. We want to understand what assumptions suffice to make that particular scheme secure.

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

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

In this paper we capture some of these hardness properties as formal assumptions. We will then show how DHIES can then be proven secure under these assumptions. Then we further explore these assumptions by studying their complexity in the generic model [29], and by studying how the assumptions relate to one other.

Results. First we formalize three new DH assumptions (though one of them, the hash DH assumption, is essentially folklore). The assumption are the *hash* DH assumption (HDH), the *oracle* DH assumption (ODH), and the the *strong* DH assumption (SDH). The HDH and ODH assumptions measure the sense in which a hash function H is "independent" of the underlying Diffie-Hellman problem. One often hears intuition asserting that two primitives are independent. Here is one way to define this. The SDH assumption formalizes, in a simple manner, that the "only" way to compute a value g^{uv} from g^v is to choose a value u and compute $(g^v)^u$. The definitions for both ODH and SDH have oracles which play a central role. See Section 4.

In Section 5 we show that DHIES is secure against chosen-ciphertext attacks. The ODH assumption is what is required to show this. Of course this means that DHIES is also secure against chosen-plaintext attacks [4] based on the

ODH assumption, but in fact we can prove the latter using the HDH assumption (although we do not show it here), a much weaker one.

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

The ODH assumption is somewhat technical; SDH is rather simpler. In Section 6 we show that, in the random-oracle model, the SDH assumption implies the ODH assumption.

In Section 6 we give a lower bound for the difficulty of the SDH assumption in the generic model of Shoup [29]. This rules out a large class of efficient attacks.

Related work. The approach above is somewhat in contrast to related schemes in the literature. More typical is to fix an assumption and then strive to find the lowest cost scheme which can be proven secure under that assumption. Examples of work in this style are that of Cramer and Shoup [13] and that of Shoup [31], who start from the decisional Diffie-Hellman assumption, and then try to find the best scheme they can that will resist chosen-ciphertext attack under this assumption. In fact, the latter can also be proved secure in the RO model based on the weaker computational Diffie-Hellman assumption. These schemes are remarkable, but their costs are about double that of ElGamal, which is already enough to dampen some practical interest. A somewhat different approach was taken by Fujisaki and Okamoto [18], starting from weaker asymmetric and symmetric schemes to construct a stronger hybrid asymmetric scheme. Their scheme can be quite practical, but the proof of security relies heavily on the use of random oracles.

2 Preliminaries

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

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

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

Message Authentication Codes. Let $\mathsf{Message} = \{0,1\}^*$ and let $\mathsf{mKey} = \{0,1\}^{mLen}$ for some number *mLen*. Let $\mathsf{Tag} = \{0,1\}^{tLen}$ for some number *tLen* (a superset of the possible tags). A *message authentication code* is a pair of algorithms $\text{MAC} = (\text{MAC.gen}, \text{MAC.ver})$. Algorithm MAC.gen (the *MAC generation algorithm*) takes a key $k \in \mathsf{mKey}$ and a message $x \in \mathsf{Message}$ and returns a string $\text{MAC.gen}(k, x)$. This string is called the *tag*. Algorithm 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 $\text{MAC.ver}(k, x, \tau) \in \{0,1\}$, 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 $x \in \mathsf{Message}$, $\text{MAC.ver}(k, x, \text{MAC.gen}(k, x)) = 1$. The first argument of either algorithm may be written as a subscript. Candidate algorithms include HMAC [3] or the CBC MAC (but only a version that is correct across messages of arbitrary length).

Symmetric Encryption. Let $\mathsf{Message}$ be as before, and let $\mathsf{eKey} = \{0,1\}^{eLen}$, for some number *eLen*. Let $\mathsf{Ciphertext} = \{0,1\}^*$ (a superset of all possible ciphertexts). Let Coins be a synonym for $\{0,1\}^\infty$ (the set of infinite strings). A *symmetric encryption scheme* is a pair of algorithms $\text{SYM} = (\text{SYM.enc}, \text{SYM.dec})$. 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$\text{SYM.enc}(k, x, r)$. 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 $\text{SYM.dec}(k, y) \in \mathsf{Message} \cup \{\mathsf{BAD}\}$. We require that for all $x \in \mathsf{Message}$, $k \in \mathsf{Key}$, and $r \in \mathsf{Coins}$

$$\text{SYM.dec}(k, \text{SYM.enc}(k, x, r)) = x.$$

Usually we omit mentioning the coins of SYM.enc, thinking of SYM.enc as a probabilistic algorithm, or thinking of $\text{SYM.enc}(k, x)$ 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. One candidate algorithms for the symmetric encryption are CBC encryption and Vernam cipher encryption.

Asymmetric Encryption. Let Coins, $\mathsf{Message}$, $\mathsf{Ciphertext}$ 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 $\text{ASYM} = (\text{ASYM.enc}, \text{ASYM.dec}, \text{ASYM.key})$. 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 $y = \text{ASYM.enc}(k, x, r)$. The decryption algorithm ASYM.dec takes a secret key $sk \in \mathsf{SK}$ and a ciphertext $y \in \mathsf{Ciphertext}$, and returns a plaintext $\text{ASYM.dec}(sk, y) \in \mathsf{Message} \cup \{\mathsf{BAD}\}$. 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

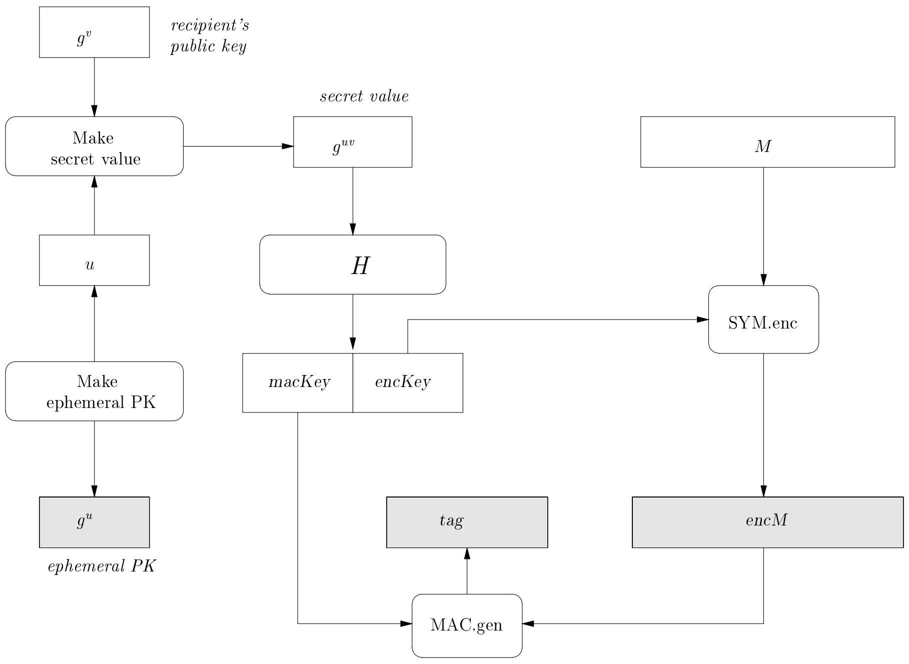

Fig. 1. *Encrypting with the scheme DHIES. We use a symmetric encryption algorithm,* SYM.enc*; a MAC generation algorithm,* MAC.gen*; and a hash function, H. The shaded rectangles comprise the ciphertext.*

$\text{ASYM.dec}(sk, \text{ASYM.enc}(pk, x, r)) = x$. The first argument to ASYM.enc and ASYM.dec may be written as a subscript.

3 The Scheme DHIES

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

Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, where group elements are represented by strings of *gLen* bits. Let SYM = (SYM.enc, SYM.dec) be a symmetric encryption scheme with key length *eLen*, and let MAC = (MAC.gen, 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 DHIES = (DHIES.enc, DHIES.dec, DHIES.key). If we want to explicitly indicate the dependency of DHIES on its associated primitives, then we will write DHIES $[\![\mathcal{G}, \text{SYM}, \text{MAC}, \text{H}]\!]$. The component algorithms of DHIES are defined in Figure 2.

```
**Algorithm** DHIES.key
  $v \leftarrow \{1, \ldots, |G|\}$ ; $pk \leftarrow g \uparrow v$ ; $sk \leftarrow v$
  **return** $(pk, sk)$
```

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

```
**Algorithm** DHIES.dec$(sk, EM)$
  $U \,\|\, encM \,\|\, tag \leftarrow EM$
  $X \leftarrow U \uparrow sk$
  $hash \leftarrow H(X)$
  $macKey \leftarrow hash[1 \,..\, mLen]$
  $encKey \leftarrow hash[mLen + 1 \,..\, mLen + eLen]$
  **if** $\mathrm{MAC.ver}(macKey, encM, tag) = 0$
    **then return** BAD
  $M \leftarrow \mathrm{SYM.dec}(encKey, encM)$
  **return** $M$
```

Fig. 2. *The scheme* $\mathrm{DHIES} = (\mathrm{DHIES.enc}, \mathrm{DHIES.dec}, \mathrm{DHIES.key})$*, 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;* $\mathcal{G} = (G, g, \_, \uparrow)$ *is a represented group whose group elements encoded by strings of length gLen; and* $H : \{0,1\}^{2\,gLen} \to \{0,1\}^{eLen+mLen}$.

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

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

4 Diffie-Hellman Assumptions

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

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

DDH: Decisional Diffie-Hellman Assumption. A stronger assumption that has been gaining popularity is the *decisional Diffie-Hellman assumption*, DDH. (For a nice discussion, see Boneh's survey [10].) It states, roughly, that the distributions (g^u, g^v, g^{uv}) and (g^u, g^v, g^w) are computationally indistinguishable when u, v, w are drawn at random from $\{1, \ldots, |G|\}$. This assumption can only hold in a group G whose order does not contain small prime factors (e.g., subgroup of order q of Z_p^* for large primes p and q). In such groups the assumption suffices to prove the semantic security of the ElGamal scheme.

Hash Diffie-Hellman Assumption: HDH. The assumption we make to prove security for DHIES under chosen-plaintext attack is weaker than DDH but stronger than CDH. It is called the *hash Diffie-Hellman assumption*, HDH. The assumption is a "composite" one—it concerns the interaction between a hash function H and the group operations in G. Here is the definition.

Definition 1. [Hash Diffie-Hellman: HDH] *Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let* hLen *be a number, let $H : \{0,1\}^* \to \{0,1\}^{hLen}$, and let A be an adversary. The advantage of A in violating the hash Diffie-Hellman assumption is*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{hdh}}_{\mathcal{G},\mathrm{H}}(A) = {} & \Pr\left[u, v \leftarrow \{1, \ldots, |G|\} :\ A(g^u, g^v, H(g^{uv})) = 1\right] - \\ & \Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\ r \leftarrow \{0,1\}^{hLen} :\ A(g^u, g^v, r) = 1\right] . \ \blacksquare\end{aligned}$$

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

We now move on to some more novel assumptions.

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

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

Definition 2. [Oracle Diffie-Hellman: ODH] *Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let hLen be a number, let $H : \{0,1\}^* \to \{0,1\}^{hLen}$, and let A be an adversary. Then the advantage of A in violating the oracle Diffie-Hellman assumption is*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{odh}}_{\mathcal{G},H}(A) = \Pr\Big[u, v \leftarrow \{1, \ldots, |G|\} :\ A^{\mathcal{H}_v(\cdot)}(g^u, g^v, H(g^{uv})) = 1\Big] - \\ \Pr\Big[u, v \leftarrow \{1, \ldots, |G|\};\ r \leftarrow \{0,1\}^{hLen} :\ A^{\mathcal{H}_v(\cdot)}(g^u, g^v, r) = 1\Big] .\end{aligned}$$

Here $\mathcal{H}_v(X) \stackrel{\text{def}}{=} H(X^v)$, and A is not allowed to call its oracle on g^u. ▌

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

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

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

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

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

Definition 3. [Strong Diffie-Hellman: SDH] *Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, and let A be an adversary. Then the advantage of A in violating* *the strong Diffie-Hellman assumption is $\mathcal{G}$ is*

$$\mathrm{Adv}^{\mathrm{sdh}}_{\mathcal{G}}(A) = \Pr\left[u, v \leftarrow \{1, \ldots, |G|\};\ \mathcal{O}_v(U, X) \overset{\mathrm{def}}{=} (X = U^v) :\ A^{\mathcal{O}_v(\cdot,\cdot)}(g^u, g^v) = g^{uv}\right] . \blacksquare$$

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

Similar intuition was captured in [21] by saying that for every non-uniform probabilistic polynomial-time algorithm A that, on input g^v, outputs (g^u, g^{uv}), there exists a non-uniform probabilistic polynomial-time algorithm S (the "extractor") that not only outputs (g^u, g^{uv}), but also u. Our approach avoids the complexity of a simulator-based formulation. We emphasize that our oracle does not return a value u (the discrete log of its first argument) but only a bit indicating whether a given pair has the right form.

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

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

where R is a resource measure and the maximum is taken over all adversaries that use resources at most R. The resources of interest in this paper are time (denoted by t) and, when appropriate, number of queries (denoted by q). Any other resources of importance will be mentioned when the corresponding notion is described. Here and throughout this paper "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).

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

5 Security against Chosen-Ciphertext Attack

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

Theorem 1. *Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group, let* SYM *be a symmetric encryption scheme, and let* MAC *be a message authentication scheme. Let* DHIES *be the asymmetric encryption scheme associated to these primitives as defined in Section 3. Then for any numbers t, q, μ, m, and m',*

$$\begin{aligned}\mathrm{Adv}^{\mathrm{cca}}_{\mathrm{DHIES}}(t, q, \mu, m) \ \le\ & \mathrm{Adv}^{\mathrm{sym}}_{\mathrm{SYM}}(t_1, 0, m, m') + 2 \cdot \mathrm{Adv}^{\mathrm{odh}}_{\mathcal{G},H}(t_2, q) + \\ & 2 \cdot q \cdot \mathrm{Adv}^{\mathrm{mac}}_{\mathrm{MAC}}(t_3, q-1)\ ,\end{aligned}$$

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

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 $\mathcal{G}$ 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 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 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 DHIES.dec_{sk} such that $\text{DHIES.dec}_{sk}(y') \neq \text{BAD}$.

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

Refer to the full version of this paper [1] for the actual proof of Theorem 1.

6 ODH and SDH

The following theorem shows that, in the RO model, the strong Diffie-Hellman assumption implies the oracle Diffie-Hellman assumption. The proof is omitted here, but can be found in the full version of this paper [1].

Theorem 2. *Let $\mathcal{G} = (G, g, \_, \uparrow)$ be a represented group and let the associated hash function H be chosen at random. Let q be the total number of queries to H-oracle. Then for any numbers t, q, μ,*

$$\mathrm{Adv}^{\mathrm{odh}}_{\mathcal{G},H}(t, \mu, q) \;\leq\; 2 \cdot \mathrm{Adv}^{\mathrm{sdh}}_{\mathcal{G}}(t_1, (q+\mu)^2)\ ,$$

where $t_1 \in t + O(gLen + hLen)$.

In this section, we prove a lower bound on the complexity of the Diffie-Hellman problem under SDH with respect to generic algorithms.

Generic Algorithms. Generic algorithms in groups are algorithms which do not make use of any special properties of the encoding of group elements other than assuming each element has a unique representation. This model was introduced by Shoup [29] and is very useful in proving lower bounds (with respect to such algorithms) for some problems. In fact, Shoup proved that in such a model both the discrete logarithm and the Diffie-Hellman problems are hard to solve as long as the order of the group contains at least one large prime factor. Following the same approach, we also use this model here to prove lower bounds for some new problems we introduce. Let us proceed now with the formalization of this model.

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

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

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

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

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

Definition 4. [SDH in generic model] *Let Z_n be the additive group of integers modulo n, let S be a set of strings of cardinality at least n, and let σ be a random encoding function of Z_n on S. In addition, let Ω be the set of all mappings Z_n to S. Let A be an generic algorithm making at most q queries to its oracles. Then the advantage of A in violating the strong Diffie-Hellman assumption is*

$$\mathrm{Adv}_A^{\mathrm{sdh}}(n,q) = \Pr\Big[\, u, v \leftarrow \{1, \ldots, |G|\};\ \sigma \leftarrow \Omega;\ \mathcal{O}_v(i,j) \stackrel{\mathrm{def}}{=} (x_j = v x_i) : A^{\mathcal{O}_v(\cdot,\cdot),\sigma}(\sigma(1), \sigma(u), \sigma(v)) = \sigma(uv) \,\Big] . \blacksquare$$

Theorem 3. *Let Z_n be the additive group of integers modulo n, let S be a set of strings of cardinality at least n, and let A be a generic algorithm. Then, for any number q,*

$$\mathrm{Adv}_A^{\mathrm{sdh}}(n,q) \leq O(q^2/p)$$

where p is the largest prime factor of n.

A corollary of Theorem 3 is that any generic algorithm solving the Diffie-Hellman problem under SDH with success probability bounded away from 0 has to perform at least $\Omega(p^{1/2})$ group operations.

Proof. Here we just present a proof sketch using a technique used by Shoup in [29]. Let $n = sp^t$ with $\gcd(s, p) = 1$. Since additional information only reduces the running time, we can assume that solving the Diffie-Hellman problem in the subgroup of order s is easy. Hence, let $n = p^t$ wlog.

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

We can assume that A outputs an element in the encoding list (otherwise $\mathrm{Adv}^{\mathrm{sdh}}_A(n, q) \leq 1/(p - m)$). Then, let us choose u and v at random from Z_{p^t}. Notice that $\mathrm{Adv}^{\mathrm{sdh}}_A(n, q)$ can be upper bounded by the probability of one of the following happening: $F_i(u, v) = F_j(u, v)$ for some F_i and F_j; or $F_i(u, v) = uv$ for some i; or $F_j \neq V_i$ but $F_j(u, v) = vF_i(u, v)$. Otherwise, the algorithm cannot learn anything about u or v except that $F_i(u, v) \neq F_j(u, v)$ for every i and j. But, using results from [29], for fixed i and j, the probability of $F_i - F_j$ vanishes is at most $1/p$; the probability of $F_i - UV$ vanishes is at most $2/p$; and the probability of $F_j - VF_i$ vanishes is at most $2/p$. It follows that the probability of one these happening is $O(q^2/p)$. ▌

References

1. M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An Encryption Scheme Based on the Diffie-Hellman Problem. Full version of current paper, available from authors' web pages.
2. American National Standards Institute (ANSI) X9.F1 subcommittee, ANSI X9.63 Public key cryptography for the Financial Services Industry: Elliptic curve key agreement and key transport schemes, Working draft, January 8, 1999.
3. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology – CRYPTO '96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
4. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
5. 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 *Proc. of the 38th* IEEE FOCS, IEEE, 1997.
6. M. Bellare, J. Kilian and P. Rogaway, The security of cipher block chaining. *Advances in Cryptology – CRYPTO '94*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.

7. 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.
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*, 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*, Lecture Notes in Computer Science Vol. 1070, U. Maurer ed., Springer-Verlag, 1996.
10. 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.
11. 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*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996.
12. Certicom Research, Standards for Efficient Crpytography Group (SECG) — SEC 1: Elliptic Curve Cryptography. Version 1.0, September 20, 2000. See http://www.secg.org/secg_docs.htm.
13. R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
14. W. Diffie and M. Hellman, New directions in cryptography. *IEEE Transactions on Information Theory*, 22, pp. 644–654, 1976.
15. D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. *Proc. of the 23rd* ACM STOC, ACM, 1991.
16. D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. Manuscript, March 1998.
17. T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, vol 31, pp. 469–472, 1985.
18. E. Fujisaki and T. Okamoto Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Advances in Cryptology – CRYPTO '99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999.
19. O. Goldreich, A uniform complexity treatment of encryption and zero-knowledge. *Journal of Cryptology,* vol. 6, 1993, pp. 21-53.
20. S. Goldwasser and S. Micali, Probabilistic encryption. *Journal of Computer and System Sciences,* vol. 28, 270–299, April 1984.
21. S. Hada and T. Tanaka, On the Existence of 3-Round Zero-Knowledge Protocols. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.
22. IEEE P1363a Committee, IEEE P1363a, Version D6, November 9, 2000. Standard specifications for public-key cryptography. See http://www.manta.ieee.org/groups/1363/P1363a/draft.html
23. 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.

24. C. Lim and P. Lee, Another method for attaining security against adaptively chosen ciphertext attacks. *Advances in Cryptology – CRYPTO '93*, Lecture Notes in Computer Science Vol. 773, D. Stinson ed., Springer-Verlag, 1993.
25. S. Micali, C. Rackoff and B. Sloan, The notion of security for probabilistic cryptosystems. *SIAM J. of Computing*, April 1988.
26. M. Naor and O. Reingold, Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Proc. of the 38th* IEEE FOCS, IEEE, 1997.
27. M. Naor and M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks. *Proc. of the 22nd* ACM STOC, ACM, 1990.
28. C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.
29. V. Shoup, Lower bounds for Discrete Logarithms and Related Problems. *Advances in Cryptology – EUROCRYPT '97*, Lecture Notes in Computer Science Vol. 1233, W. Fumy ed., Springer-Verlag, 1997.
30. V. Shoup, Personal Communication.
31. V. Shoup, Using Hash Functions as a Hedge against Chosen Ciphertext Attack. *Advances in Cryptology – EUROCRYPT '00*, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed., Springer-Verlag, 2000.
32. Y. Zheng, Public key authenticated encryption schemes using universal hashing. Contribution to P1363. ftp://stdsbbs.ieee.org/pub/p1363/contributions/aes-uhf.ps
33. 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 Security Definitions

Symmetric Encryption. Security of a symmetric encryption scheme is defined as in [5], in turn an adaptation of the notion of polynomial security as given in [20,25]. We imagine an adversary A that runs in two stages. During either stage the adversary may query an encryption oracle $\mathrm{SYM.enc}(K,\cdot)$ which, on input x, returns $\mathrm{SYM.enc}(K,x,r)$ for a randomly chosen r. In the adversary's find stage it endeavors to come up with a pair of equal-length messages, x_0 and x_1, whose encryptions it wants to try to tell apart. It also retains some state information s. In the adversary's guess stage it is given a random ciphertext y for one of the plaintexts x_0, x_1, together with the saved state s. The adversary "wins" if it correctly identifies which plaintext goes with y. The encryption scheme is "good" if "reasonable" adversaries can't win significantly more than half the time.

Definition 1 [5] *Let* $\mathrm{SYM} = (\mathrm{SYM.enc}, \mathrm{SYM.dec})$ *be a symmetric encryption scheme and let* A *be an adversary. The advantage of* A *in attacking* SYM *is*

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

We define

$$\mathrm{Adv}^{\mathrm{sym}}_{\mathrm{SYM}}(t, \mu, m, m') = \max_{A} \{\mathrm{Adv}^{\mathrm{sym}}_{\mathrm{SYM}}(A)\} ,$$

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 2.

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

Definition 2 *Let* $\mathrm{MAC} = (\mathrm{MAC.gen}, \mathrm{MAC.ver})$ *be a message authentication scheme and let F be an adversary. Then the success (or forging probability) of F on* MAC *is*

$$\mathrm{Adv}^{\mathrm{mac}}_{\mathrm{MAC}}(A) = \Pr\Big[\, K \leftarrow \mathsf{mKey};\ (x^*, \tau^*) \leftarrow F^{\mathrm{MAC.gen}(K,\cdot)} : \\ \mathrm{MAC.ver}_K\,(x^*, \tau^*) = 1 \textit{ and } (x^*, \tau^*) \textit{ is unasked} \,\Big] .$$

The security of MAC *is the function*

$$\mathrm{Adv}^{\mathrm{mac}}_{\mathrm{MAC}}(t, q) = \max_{F} \{\mathrm{Adv}^{\mathrm{mac}}_{\mathrm{MAC}}(F)\} ,$$

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 $\mathrm{MAC.ver}_K\,(x^*, \tau^*) = 1$ and (x^*, τ^*) is unasked.

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

Candidate algorithms were discussed in Section 2.

Privacy Against Adaptive Chosen-Ciphertext Attack. Our definition of chosen-ciphertext security of an asymmetric encryption mimics the find-then-guess notion of [5] and follows [20,25,19], in which the the adversary is given access to a decryption oracle in both the find and guess stages. So we state it without further discussion.

Definition 3 *Let* ASYM = (ASYM.enc, ASYM.dec, 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*

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

Here A is not allowed to call its decryption oracle on y. The security of ASYM *is the function*

$$\mathrm{Adv}^{\mathrm{cca}}_{\mathrm{ASYM}}(t,q,\mu,m)=\max_{A}\left\{\mathrm{Adv}^{\mathrm{cca}}_{\mathrm{ASYM}}(A)\right\}\,,$$

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