Using Hash Functions as a Hedge against Chosen Ciphertext Attack*

Victor Shoup
IBM Zurich Research Lab, Säumerstr. 4, 8803 Rüschlikon, Switzerland
sho@zurich.ibm.com

May 9, 2000

Abstract

The cryptosystem recently proposed by Cramer and Shoup [CS98] is a practical public key cryptosystem that is secure against adaptive chosen ciphertext attack provided the Decisional Diffie-Hellman assumption is true. Although this is a reasonable intractability assumption, it would be preferable to base a security proof on a weaker assumption, such as the Computational Diffie-Hellman assumption. Indeed, this cryptosystem in its most basic form is in fact *insecure* if the Decisional Diffie-Hellman assumption is false. In this paper we present a practical hybrid scheme that is just as efficient as the scheme of of Cramer and Shoup; indeed, the scheme is slightly more efficient than the one originally presented by Cramer and Shoup; we *prove* that the scheme is secure if the Decisional Diffie-Hellman assumption is true; we *give strong evidence* that the scheme is secure if the weaker, Computational Diffie-Hellman assumption is true by providing a proof of security in the *random oracle* model.

1 Introduction

It is largely agreed upon in the cryptographic research community that the "right" definition of security for a public key cryptosystem is security against adaptive chosen ciphertext attack, as defined by Rackoff and Simon [RS91] and Dolev, Dwork, and Naor [DDN91]. At least, this is the definition of

*This is a minor modification of the version of the paper in *Proc. Eurocrypt 2000*; it is a major modification of IBM Research Report RZ 3139 (June 21, 1999).

security that allows the cryptosystem to be deployed safely in the widest range of applications.

Dolev, Dwork, and Naor [DDN91] presented a cryptosystem that could be proven secure in this sense using a reasonable intractability assumption. However, their scheme was quite impractical. Subsequently, Bellare and Rogaway [BR93, BR94] presented very practical schemes, and analyzed their security under the standard RSA assumption; more precisely, they proved the security of these schemes in the *random oracle* model, wherein a cryptographic hash function is treated *as if* it were a "black box" containing a random function. However, the security of these schemes in the "real world" (i.e., the standard model of computation) has never been proved.

A proof of security in the random oracle model provides strong evidence that breaking the scheme without breaking the underlying intractability assumptions will be quite difficult to do, although it does not rule out this possibility altogether. The advantage of a proof of security in the "real world" is that it does not just provide such strong evidence, it *proves* that the scheme cannot be broken without breaking the underlying intractability assumptions.

Recently, Cramer and Shoup [CS98] presented a practical cryptosystem and proved its security in the standard model, based on the Decisional Diffie-Hellman (DDH) assumption. It is hard to compare the security of this scheme with that of the schemes of Bellare and Rogaway—although the former scheme can be analyzed in the "real world," and the latter schemes only in the random oracle model, the underlying intractability assumptions are incomparable. Indeed, a proof of security is worthless if the underlying assumptions turn out to be false, and in fact, both the Cramer-Shoup scheme (in its basic form) and the Bellare-Rogaway schemes can be broken if their respective assumptions are false.

Perhaps the strongest criticism against the Cramer-Shoup scheme is that the assumption is too strong; in particular, it has not been studied as extensively as other assumptions, including the RSA assumption.

In this paper, we address this criticism by presenting a hybrid variation of the Cramer-Shoup scheme. This scheme is actually somewhat simpler and more efficient than the original, and a proof of security in the "real world" can also be made based on the DDH assumption. However, the same scheme can also be proved secure in the random oracle model based on the Computational Diffie-Hellman (CDH) assumption. This assumption was introduced by Diffie and Hellman [DH76] in their work that opened the field of public key cryptography, and has been studied at least as intensively as any other intractability assumption used in modern cryptography.

Thus, in comparison to other available practical encryption schemes, the scheme discussed here is arguably no less secure, while still admitting a proof of security in the "real world" under a reasonable, if somewhat strong, intractability assumption.

We believe this "hedging with hash" approach may be an attractive design paradigm. The general form of this approach would be to design practical cryptographic schemes whose security can be proved in the "real world" based on a reasonable, if somewhat strong, intractability assumption, but whose security can also be proved in the random oracle model under a weaker intractability assumption. This same "hedging with hash" security approach has also been applied to digital signature schemes: Cramer and Shoup [CS99] presented and analyzed a practical signature scheme that is secure in the "real world" under the so-called Strong RSA assumption, but is also secure in the random oracle model under the ordinary RSA assumption. Although that paper and this paper both advocate this "hedging with hash" security approach, the technical details and proof techniques are quite unrelated. In the context of encryption or signatures, one can also "hedge" just by combining two schemes based on different intractability assumptions (via composition for encryption and via concatenation for signatures). However, this type of hedging is much more expensive computationally, and much less elegant than the type of hedging we are advocating here.

Other Diffie-Hellman based encryption schemes. [TY98] present a scheme, but it cannot be proved secure against adaptive chosen ciphertext attack under any intractability assumption, even in the random oracle model. There is indeed a security analysis in [TY98], but rather than basing the proof of security on the hardness of a specific problem, it is based on the assumption that the adversary behaves in a specific way, similar to as was done in [ZS92]. [SG98] present two schemes; the first can be proved secure against adaptive chosen ciphertext attack in the random oracle model under the CDH, while the proof of security for the second relies on the DDH. Both schemes are amenable to distributed decryption. Moreover, the techniques in the current paper can be applied to the second scheme to weaken the intractability assumption, replacing the DDH with the CDH (but not the distributed version). [SG98] also discusses an encryption scheme that is essentially the same as that in [TY98], and argues why it would be quite difficult using known techniques to prove that such is scheme is secure against adaptive chosen ciphertext attack even in the random oracle model. [ABR98] present a scheme for which security against adaptive chosen ciphertext attack can only be proved under non-standard assumptions—these assumptions relate to the hardness of certain "interactive" problems, and as such they do not qualify as "intractability assumptions" in the usual sense of the term. Furthermore, using random oracles does not seem to help. [FO99] present a scheme that can be proven secure against adaptive chosen ciphertext attack under the CDH assumption in the random oracle model. Moreover, they present a fairly general method of converting any public-key encryption scheme that is semantically secure into one that can be proved secure against adaptive chosen ciphertext attack in the random oracle model. However, nothing at all can be said about the security of this scheme in the "real world."

2 Security against Adaptive Chosen Ciphertext Attack

We recall the definition of security against adaptive chosen ciphertext attack.

We begin by describing the attack scenario.

First, the key generation algorithm is run, generating the public key and private key for the cryptosystem. The adversary, of course, obtains the public key, but not the private key.

Second, the adversary makes a series of arbitrary queries to a *decryption oracle*. Each query is a ciphertext ψ that is decrypted by the decryption oracle, making use of the private key of the cryptosystem. The resulting decryption is given to the adversary. The adversary is free to construct the ciphertexts in an arbitrary way—it is certainly *not* required to compute them using the encryption algorithm.

Third, the adversary prepares two messages m_0, m_1, and gives these two an *encryption oracle*. The encryption oracle chooses $b \in \{0,1\}$ at random, encrypts m_b, and gives the resulting "target" ciphertext ψ' to the adversary. The adversary is free to choose m_0 and m_1 in an arbitrary way, except that if message lengths are not fixed by the cryptosystem, then these two messages must nevertheless be of the same length.

Fourth, the adversary continues to submit ciphertexts ψ to the decryption oracle, subject only to the restriction that $\psi \neq \psi'$.

Just before the adversary terminates, it outputs $b' \in \{0,1\}$, representing its "guess" of b.

That completes the description of the attack scenario.

The adversary's *advantage* in this attack scenario is defined to be the distance from 1/2 of the probability that $b' = b$.

A cryptosystem is defined to be *secure against adaptive chosen ciphertext attack* if for any efficient adversary, its advantage is negligible.

Of course, this is a complexity-theoretic definition, and the above description suppresses many details, e.g., there is an implicit security parameter which tends to infinity, and the terms "efficient" and "negligible" are technical terms, defined in the usual way. One can work in a *uniform* (i.e., Turing machines) or a *non-uniform* model (i.e., circuits) of computation. This distinction will not affect any results in this paper.

3 Intractability Assumptions

In this section, we discuss the intractability assumptions used in this paper.

Let G be a group of large prime order q.

The Discrete Logarithm (DL) problem is this: given $g \in G$ with $g \neq 1$ and g^x, compute x (modulo q).

The Computational Diffie-Hellman (CDH) problem is this: given $g \in G$ with $g \neq 1$, along with g^x and g^y, compute g^{xy}. A "good" algorithm for this problem is an efficient, probabilistic algorithm such that for all inputs, its output is correct with all but negligible probability. The CDH assumption is the assumption that no such "good" algorithm exists. Using well-known random-self reductions, along with the results of [MW96] or [Sho97], the existence of such a "good" algorithm is equivalent to the existence of a probabilistic algorithm that outputs a correct answer with non-negligible probability, where the probability is taken over the coin flips of the algorithm, as well as a random choice of $g \in G$, and $x, y \in \mathbf{Z}_q$.

The Decisional Diffie-Hellman (DDH) problem is this: given $g \in G$ with $g \neq 1$, along with g^x, g^y, and g^z decide if $z \equiv xy \bmod q$. A "good" algorithm is an efficient, probabilistic algorithm such that for all inputs, its output is correct with all but negligible probability. The DDH assumption is the assumption that no such "good" algorithm exists. Using the random-self reduction presented by Stadler [Sta96], the existence of such a "good" algorithm is equivalent to the existence of a probabilistic statistical test distinguishing the distributions (g, g^x, g^y, g^z) and (g, g^x, g^y, g^{xy}), where $g \in G$, and $x, y, z \in \mathbf{Z}_q$ are randomly chosen.

All of these problems are equally hard in a "generic" model of computation, where an algorithm is not allowed to exploit the representation of the group G [Sho97]; in this model, $O(\sqrt{q})$ group operations are both necessary and sufficient. However, for specific groups, special methods, such as "index calculus" methods, may apply, allowing for more efficient algorithms.

In general, the only known way to solve either the CDH or DDH problems is to first solve the DL problem. However, there remains the possibility that the DL problem is hard and the CDH problem is easy, or that the CDH problem is hard, and the DDH problem is easy. Maurer [Mau94] has shown that under certain circumstances, an algorithm for solving the CDH problem can be used to solve the DL problem. This reduction is a "generic" reduction that does not depend on the representation of the group G. It can also be shown that there is no such generic reduction allowing one to efficiently solve the CDH or DL problems using an algorithm for the DDH problem. This fact could be considered as evidence supporting the claim that the DDH assumption is possibly stronger than the CDH assumption.

It is perhaps worth stressing that although the DDH may be a stronger assumption than either the DL or CDH assumption, these latter two "usual" assumptions have rarely, if ever, been used to prove the security of a practical cryptographic scheme of any kind—except in the random oracle model. Indeed, it appears to be a widely held misconception that the security of the Diffie-Hellman key exchange protocol [DH76] and variants thereof (e.g., [DvOW92]) is implied by the CDH assumption. This is simply not the case—under *any* reasonable definition of security, except in the random oracle model. One can use the DDH assumption, however, as the basis for proving the security of such schemes (see, e.g., [BCK98, Sho99]).

The DDH assumption appears to have first surfaced in the cryptographic literature in [Bra93]. For other applications and discussion of the DDH, see [Bon98, NR97].

As in the previous section, we have suppressed many details in the above discussion, e.g., there is an implicit security parameter that tends to infinity, and for each value of the security parameter, there is an implicit probability distribution of groups.

4 The Encryption Scheme

4.1 The basic Cramer-Shoup scheme

We recall the basic Cramer-Shoup cryptosystem, as presented in [CS98]. The cryptosystem works with a group G of large prime order q.

Key Generation. The key generation algorithm runs as follows. Random elements $g_1, g_2 \in G\backslash\{1\}$ are chosen, and random elements

$$x_1, x_2, y_1, y_2, z \in \mathbf{Z}_q$$

are also chosen. Next, the group elements

$$c = g_1^{x_1} g_2^{x_2}, \; d = g_1^{y_1} g_2^{y_2}, \; h = g_1^z$$

are computed. Finally, a random key k indexing a universal one-way hash function UOWH is chosen. We assume that the output of the hash function is an element of $\mathbf{Z}_q$. The public key is (g_1, g_2, c, d, h, k), and the private key is (x_1, x_2, y_1, y_2, z).

Encryption. To encrypt, we assume a message m can be encoded as an element of G. The encryption algorithm runs as follows. First, it chooses $r \in \mathbf{Z}_q$ at random. Then it computes

$$u_1 = g_1^r, \; u_2 = g_2^r, \; e = h^r m, \; \alpha = \mathsf{UOWH}(k; u_1, u_2, e), \; v = c^r d^{r\alpha}.$$

The ciphertext is

$$(u_1, u_2, e, v).$$

Decryption. Given a ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = \mathsf{UOWH}(k; u_1, u_2, e)$, and tests if

$$u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha} = v.$$

If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs

$$m = e / u_1^z.$$

In [CS98], it was shown that this scheme is secure against adaptive chosen ciphertext attack, under the DDH assumption for G, and assuming UOWH is a secure universal one-way hash function. Although there are theoretical constructions for UOWH [NY89], a reasonable construction would be to use the compression function of SHA-1, in conjunction with the constructions in [BR97] or [Sho00]. With this approach, the security of UOWH can be based on the assumption that the SHA-1 compression function is second-preimage collision resistant, a potentially much weaker assumption than full collision resistance.

4.2 A general hybrid construction

We describe here a general method for constructing a hybrid encryption scheme. To this end, it is convenient to define the notion of a *key encapsulation scheme.* This is a scheme that allows a party to generate a random bit string and send it to another party, encrypted under the receiving party's public key.

A key encapsulation scheme works just like a public key encryption scheme, except that the encryption algorithm takes no input other than the recipient's public key. Instead, the encryption algorithm generates a pair (K, ψ), where K is a random bit string of some specified length, say l, and ψ is an encryption of K, that is, the decryption algorithm applied to ψ yields K.

One can always use a public key encryption scheme for this purpose, generating a random bit string, and then encrypting it under the recipient's public key. However, as we shall see, one can construct a key encapsulation scheme in other ways as well.

One can easily adapt the notion of security against adaptive chosen ciphertext attack to a key encapsulation scheme. The only difference in the attack scenario is the behavior of the encryption oracle. The adversary does not give two messages to the encryption oracle. Rather, the encryption oracle runs the key encapsulation algorithm to obtain a pair (K', ψ'). The encryption oracle then gives the adversary either (K', ψ') or (K'', ψ'), where K'' is an independent random l-bit string; the choice of K' versus K'' depends on the value of the random bit b chosen by the encryption oracle.

Using a key encapsulation scheme that is secure against adaptive chosen ciphertext attack, we can construct a hybrid public key cryptosystem that is secure against adaptive chosen ciphertext attack as follows.

We need a pseudo-random bit generator PRBG. There are theoretical constructions for such a generator, but a perfectly reasonable approach is to construct the generator using a standard block cipher, such as DES, basing its security on a reasonable pseudo-randomness assumption on the underlying block cipher. We assume that PRBG stretches l-bit strings to strings of arbitrary length. We assume here that $1/2^l$ is a negligible quantity.

We need a hash function AXUH suitable for message authentication, i.e., an almost XOR-universal hash function [Kra94]. We assume that AXUH is keyed by an l'-bit string and hashes arbitrary bit strings to l-bit strings. Many efficient constructions for AXUH exist that do not require any intractability assumptions.

To encrypt a message m, we run the key encapsulation scheme to obtain a random string K along with its encryption ψ. Next, we apply PRBG to K to obtain an l'-bit string K_1, an l-bit string K_2, and an $|m|$-bit string f. Finally, we compute

$$e = f \oplus m, \ a = \mathsf{AXUH}(K_1; e) \oplus K_2.$$

The ciphertext is

$$(\psi, e, a).$$

To decrypt (ψ, e, a), we first decrypt ψ to obtain K. Note that decrypting ψ may result in a "reject," in which case we "reject" as well. Otherwise, we apply PRBG to K to obtain an l'-bit string K_1, an l-bit string K_2, and an $|e|$-bit string f. We then test if $a = \mathsf{AXUH}(K_1; e) \oplus K_2$. If this condition does not hold, we "reject." Otherwise, we output $m = e \oplus f$.

Theorem 1 *If the underlying key encapsulation scheme is secure against adaptive chosen ciphertext attack, and* PRBG *is a secure pseudo-random bit generator, then the above hybrid scheme is also secure against adaptive chosen ciphertext attack.*

This appears to be somewhat of a "folk theorem." The proof is straightforward, and is left as an easy exercise for the reader.

4.3 A hybrid Cramer-Shoup scheme

We now describe a key encapsulation scheme based on the Cramer-Shoup encryption scheme. Combined with the general hybrid construction in §4.2, this yields a hybrid encryption scheme. As a hybrid scheme, it is much more flexible than the "basic" version of the scheme described in §4.1, as messages may be arbitrary bit strings and do not need to be encoded as group elements. This flexibility allows one greater freedom in choosing the group G, which can be exploited to obtain a much more efficient implementation as well. Also, the scheme we describe incorporates some modifications that lead to a simpler and more efficient decryption algorithm.

We need a pair-wise independent hash function PIH. We assume that PIH takes a key μ and maps elements $\lambda \in G$ to l-bit strings. Many efficient constructions for PIH exist that do not require any intractability assumptions. We will want to apply the Entropy Smoothing Theorem (see [Lub96, Ch. 8] or [IZ89]) to PIH, assuming that the input λ is a random group element. To do this effectively, the relative sizes of q and l must be chosen appropriately, so that $\sqrt{2^l/q}$ is a negligible quantity.

We also need a "magic" hash function MH mapping elements of $G \times G$ to l-bit strings. This function is not required to satisfy any particular security requirements. A construction using a cryptographic hash like MD5 or SHA-1 is recommended (see [BR93]). This function will only play a role when we analyze the scheme in the random oracle model, where MH will be modeled as a random oracle.

Now we are ready to describe the key encapsulation scheme.

Key Generation. A random element $g_1 \in G \backslash \{1\}$ is chosen, together with $w \in \mathbf{Z}_q \backslash \{0\}$ and $x, y, z \in \mathbf{Z}_q$. Next, the following group elements are computed:

$$g_2 = g_1^w,\ c = g_1^x,\ d = g_1^y,\ h = g_1^z.$$

Finally, a random key k indexing a universal one-way hash function UOWH is chosen, as well as a random key μ for PIH. The public key is $(g_1, g_2, c, d, h, k, \mu)$, and the private key is (w, x, y, z).

Key Encapsulation. The key encapsulation scheme runs as follows. First, it chooses $r \in \mathbf{Z}_q$ at random. Then it computes

$$u_1 = g_1^r,\ u_2 = g_2^r, \alpha = \mathsf{UOWH}(k; u_1, u_2),\ v = c^r d^{r\alpha},\ \lambda = h^r.$$

Finally, it computes

$$K = \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1, \lambda).$$

The ciphertext is

$$(u_1, u_2, v),$$

which is an encryption of the key K.

Decryption. Given a ciphertext (u_1, u_2, v), the decryption algorithm runs as follows. It first computes $\alpha = \mathsf{UOWH}(k; u_1, u_2)$, and tests if

$$u_2 = u_1^w \text{ and } v = u_1^{x+\alpha y}.$$

If this condition does not hold, the decryption algorithm outputs "reject" and halts. Otherwise, it computes $\lambda = u_1^z$, outputs the key $K = \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1, \lambda)$.

Theorem 2 *The above key encapsulation scheme is secure against adaptive chosen ciphertext attack, under the DDH assumption for G, and also assuming that* UOWH *is a secure universal one-way hash function.*

We only briefly sketch the proof, as it differs only slightly from the proof of the main theorem in [CS98]. The structure of the proof is as follows. We make a sequence of transformations to the attack game. In each of these transformations, we argue that we affect the adversary's advantage by a negligible amount, and in the final transformed game, the adversary's advantage is zero. The original game is denoted $\mathbf{G}_0$, and the transformed games are denoted $\mathbf{G}_i$, for $i = 1, 2, \ldots$.

First, some notation. Let $\psi' = (u_1', u_2', v')$ be the "target" ciphertext. For notational convenience and clarity, the internal variables used by the encryption algorithm in generating the target ciphertext will also be referred to in "primed" form, e.g., the value of α for the target ciphertext is denoted α'. Also, we will call a ciphertext (u_1, u_2, v) *valid* if $\log_{g_1} u_1 = \log_{g_2} u_2$; otherwise, it is called *invalid.*

In game $\mathbf{G}_1$, we change the key generation algorithm as follows. It chooses random $g_1, g_2 \in G\backslash\{1\}$ at random, along with

$$x_1, x_2, y_1, y_2, z_1, z_2 \in \mathbf{Z}_q.$$

Next, it computes the following group elements:

$$c = g_1^{x_1} g_2^{x_2},\ d = g_1^{y_1} g_2^{y_2},\ h = g_1^{z_1} g_2^{z_2}.$$

It chooses the keys k and μ as before, and the public key is $(g_1, g_2, c, d, h, k, \mu)$. We also modify the decryption oracle as follows. Given a ciphertext (u_1, u_2, v), it computes $\alpha = \mathsf{UOWH}(k; u_1, u_2)$, and tests if

$$v = u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha}.$$

If this condition does not hold, the decryption oracle outputs "reject." Otherwise, it computes $\lambda = u_1^{z_1} u_2^{z_2}$, and outputs the key $K = \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u_1, \lambda)$.

We now claim that the adversary's advantage in game $\mathbf{G}_1$ differs from its advantage in game $\mathbf{G}_0$ by a negligible amount. The argument is runs along the same lines as that of the proof of Lemma 1 in [CS98]. That is, these two games are equivalent up to the point where an invalid ciphertext is not rejected; however, the probability that this happens is negligible.

In game $\mathbf{G}_2$, we modify the encryption oracle, simply choosing $u_1', u_2' \in G$ at random, setting

$$v' = (u_1')^{x_1 + y_1\alpha'} (u_2')^{x_2 + y_2\alpha'},$$

and computing the rest of the target ciphertext as usual.

It is clear that under the DDH assumption, the adversary's advantage in game $\mathbf{G}_2$ differs from its advantage in game $\mathbf{G}_1$ by a negligible amount.

In game $\mathbf{G}_3$, we move the computation of the target ciphertext $\psi' = (u_1', u_2', v')$ to the very beginning of the game, and if the adversary ever submits a ciphertext $\psi = (u_1, u_2, v)$ to the decryption oracle with $(u_1, u_2) \neq (u_1', u_2')$, but with $\alpha = \alpha'$, we simply halt the game.

It is clear that under the security assumption for UOWH, adversary's advantage in game $\mathbf{G}_3$ differs from its advantage in game $\mathbf{G}_2$ by a negligible amount.

In game $\mathbf{G}_4$, we modify the encryption oracle yet again, choosing λ' as a random group element.

That this only has a negligible impact on the adversary's advantage follows from the line of reasoning in the proof of Lemma 2 in [CS98]. That is, these two games are equivalent up to the point where an invalid ciphertext is not rejected (provided $\log_{g_1} u'_1 \neq \log_{g_2} u'_2$); however, the probability that this happens is negligible (this makes use of the fact that no collisions in UOWH are found).

In game $\mathbf{G}_5$, we modify the encryption oracle again, this time choosing K' as a random l-bit string.

This modification only has a negligible impact on the adversary's advantage. Indeed, since PIH is a pair-wise independent hash function, so is the function

$$\lambda \mapsto \mathsf{PIH}(\mu; \lambda) \oplus \mathsf{MH}(u'_1, \lambda),$$

where we view (μ, u'_1) as the key to this hash function. By the Entropy Smoothing Theorem, the value $K' = \mathsf{PIH}(\mu; \lambda') \oplus \mathsf{MH}(u'_1, \lambda')$ is statistically indistinguishable from a random l-bit string.

It is clear that in game $\mathbf{G}_5$, the adversary's advantage is zero. That completes the proof of the theorem.

Theorem 3 *Modeling* MH *as a random oracle, the above key encapsulation scheme is secure against adaptive chosen ciphertext attack, under the CDH assumption for G, and also assuming that* UOWH *is a secure universal one-way hash function.*

To prove this theorem, suppose there is an adversary that has a non-negligible advantage in the attack game. Now, Theorem 2 remains valid, even if we replace MH by a random oracle. So assuming the security properties of UOWH, the existence of an efficient adversary with non-negligible advantage implies the existence of an efficient algorithm for solving the DDH problem in G. In fact, the proof of Theorem 2 shows how to construct such an algorithm using the adversary as a subroutine; though technically "efficient," this may not be the most practical algorithm for solving the DDH problem in G; a more practical algorithm would certainly make the simulator we describe below more efficient.

In any case, we assume we have an efficient algorithm solving the DDH problem. To be precise, define the function $\mathsf{DHP}(g, g^x, g^y, g^z)$ to be 1 if $g \neq 1$ and $z \equiv xy \bmod q$, and 0 otherwise. Then our assumption is that there is an efficient probabilistic algorithm that on all inputs computes DHP correctly, with negligible error probability.

Now we show how to use such an algorithm for DHP, together with an adversary that has non-negligible advantage in the attack game, to construct an efficient algorithm for solving the CDH problem. We assume that the instance of the CDH problem consists of randomly chosen group elements $g_1, u_1', h \in G$ (with $g_1 \neq 1$), and our goal is to compute $\lambda' \in G$ such that $\mathsf{DHP}(g_1, u_1', h, \lambda') = 1$.

We describe a simulator that simulates the adversary's view in the attack game. The input to the simulator is $g_1, u_1', h \in G$ as above.

The simulator constructs a public key for the cryptosystem as follows. It chooses $w \in \mathbf{Z}_q \backslash \{1\}$ at random and sets $g_2 = g_1^w$. It chooses $x, y \in \mathbf{Z}_q$ at random, and computes $c = g_1^x, d = g_1^y \in G$, It also generates a random key k for UOWH and a random key μ for PIH. The public key is $(g_1, g_2, c, d, h, k, \mu)$.

The simulator is in complete control of the random oracle representing MH. We maintain a set S (initially empty) of tuples

$$(u_1, \lambda, \nu) \in G \times G \times \{0,1\}^l,$$

representing the portion of MH that has been defined so far. That is, $\mathsf{MH}(u_1, \lambda)$ is defined to be ν if and only if $(u_1, \lambda, \nu) \in S$. We also maintain the subset $S_{DDH} \subset S$ of tuples (u_1, λ, ν) satisfying the additional constraint that $\mathsf{DHP}(g_1, u_1, h, \lambda) = 1$. We also maintain a set S' (initially empty) of pairs

$$(u_1, K) \in G \times \{0,1\}^l.$$

To process a request to evaluate the random oracle at a point $(u_1, \lambda) \in G \times G$, the simulator executes the algorithm shown in Figure 1.

The manner in which pairs are added to S' is described below, in the description of the simulation of the decryption oracle.

We next describe how the simulator deals with the encryption oracle. It computes $u_2' = (u_1')^w$, and computes $v' = (u_1')^{x+\alpha' y}$. It outputs a random l-bit string K' and the "target" ciphertext $\psi' = (u_1', u_2', v')$. Note that the output of the encryption oracle is independent of the random bit b.

Now we describe how the simulator deals with the decryption oracle. The algorithm used to process a request to decrypt a ciphertext $\psi = (u_1, u_2, v) \neq \psi'$ is shown in Figure 2.

That completes the description of the simulator. It is easy to verify that the actual attack and the attack played against this simulator are equivalent, at least up to the point where the adversary queries the random oracle at the point (u_1', λ'). But up to that point, the hidden bit b is independent of the adversary's view. Therefore, since we are assuming the adversary does have

if $(u_1, \lambda, \nu) \in S$ for some $\nu \in \{0,1\}^l$
    return ν
else if $\mathsf{DHP}(g_1, u_1, h, \lambda) = 0$ {
    $\nu \leftarrow_R \{0,1\}^l$
    $S \leftarrow S \cup \{(u_1, \lambda, \nu)\}$
    return ν
}
else if $u_1 = u'_1$
    output the solution $\lambda = \lambda'$ to the CDH problem and halt
else {
    if $(u_1, K) \in S'$ for some $K \in \{0,1\}^l$
        $\nu \leftarrow K \oplus \mathsf{PIH}(\mu; \lambda)$
    else
        $\nu \leftarrow_R \{0,1\}^l$
    $S \leftarrow S \cup \{(u_1, \lambda, \nu)\}$
    $S_{DDH} \leftarrow S_{DDH} \cup \{(u_1, \lambda, \nu)\}$
    return ν
}

Figure 1: Simulator's algorithm to evaluate random oracle at (u_1, λ).

$\alpha \leftarrow \mathsf{UOWH}(k; u_1, u_2)$
if $u_2 \neq u_1^w$ or $v \neq u_1^{x+\alpha y}$
    return "reject"
else if $(u_1, K) \in S'$ for some $K \in \{0,1\}^l$
    return K
else if $(u_1, \lambda, \nu) \in S_{DDH}$ for some $\lambda \in G$ and $\nu \in \{0,1\}^l$
    return $\nu \oplus \mathsf{PIH}(\mu; \lambda)$
else {
    $K \leftarrow_R \{0,1\}^l$
    $S' \leftarrow S' \cup \{(u_1, K)\}$
    return K
}

Figure 2: Simulator's algorithm to decrypt $\psi = (u_1, u_2, v)$.

a non-negligible advantage, the adversary must query the random oracle at the point (u_1', λ') with non-negligible probability.

That completes the proof of Theorem 3.

Remarks

Remark 1. The decryption algorithm tests if $u_2 = u_1^w$ and $v = u_1^{x+\alpha y}$. In the proof of Theorem 2, we show that we can replace this test with a different test that is equivalent from the point of view of the data the adversary sees; however, these tests may not be equivalent from the point of view of *timing information.* In particular, if the decryption algorithm returns "reject" immediately after finding that $u_2 \neq u_1^w$, this could perhaps leak timing information to the adversary that is not available in game $\mathbf{G}_1$ in the proof. We therefore recommend that *both* the tests $u_2 = u_1^w$ and $v = u_1^{x+\alpha y}$ are performed, even if the one of them fails.

Remark 2. In a typical implementation, the group G may be a subgroup of $\mathbf{Z}_p^*$ for a prime p, perhaps where p is much larger than q. In this case, after testing if the encodings of u_1, u_2, v properly represent elements of $\mathbf{Z}_p^*$, the decryption algorithm must check that $u_1^q = 1$, so as to ensure that $u_1 \in G$. We need not make any further tests to check that $u_2, v \in G$, since this is already implied by the tests $u_2 = u_1^w$ and $v = u_1^{x+\alpha y}$.

Remark 3. The decryption algorithm must compute either three or four exponentiations, all with respect to the same base u_1. An implementation can and should exploit this to get a significantly more efficient decryption algorithm by using precomputation techniques (see, e.g. [LL94]).

Remark 4. The reduction given in the proof of Theorem 3 is perhaps not as efficient as one would like. If T is the time required to solve the DDH problem, and Q queries are made to the random oracle, then the running time of the simulator will be essentially that of the adversary plus $O(T \cdot Q)$. Also, note that the inclusion of u_1 as an argument to MH is not essential to get a polynomial-time security reduction; however, if we dropped u_1 as an argument to MH, the only simulator we know how to construct has a term of $O(Q \cdot Q' \cdot T)$ in its running time, where Q' is the number of decryption oracle queries.

Remark 5. In the proof of Theorem 3, we argued that if there is an adversary with a non-negligible advantage in the attack game, then there is an efficient algorithm for solving the DDH. This perhaps deserves some elaboration. For such an adversary A, there exists a polynomial $P(\gamma)$ in the security parameter γ, and an infinite set Γ of choices of the security parameter, such that for all $\gamma \in \Gamma$, the advantage of A is at least $1/P(\gamma)$. We are assuming that the group G is generated by a probabilistic function $\mathcal{G}(\gamma)$ that takes the security parameter γ as input. For an algorithm A', a security parameter γ, and $0 \leq \epsilon \leq 1$, define $V(A', \gamma, \epsilon)$ be the set of outputs G of $\mathcal{G}(\gamma)$ such that A' computes DHP on G with error probability at most ϵ. As in the previous remark, let Q be (an upper bound on) the number or random oracle queries made by A. Then the existence of A, together with Theorem 2, implies that there exists an efficient algorithm A' and a polynomial $P'(\gamma)$, such that for all $\gamma \in \Gamma$, $\Pr[\mathcal{G}(\gamma) \in V(A', \gamma, 1/(2Q))] \geq 1/P'(\gamma)$. The reduction described in the proof of Theorem 3 only works when $\gamma \in \Gamma$ and $\mathcal{G}(\gamma) \in V(A', \gamma, 1/(2Q))$, but this is enough to contradict the CDH assumption.

References

[ABR98] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: an encryption scheme based on the Diffie-Hellma problem. Submission to IEEE P1363, 1998.

[BCK98] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *30th Annual ACM Symposium on Theory of Computing*, 1998.

[Bon98] D. Boneh. The Decision Diffie-Hellman Problem. In *Ants-III*, pages 48–63, 1998. Springer LNCS 1423.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—Crypto '94*, pages 92–111, 1994.

[BR97] M. Bellare and P. Rogaway. Collision-resistant hashing: towards making UOWHFs practical. In *Advances in Cryptology–Crypto '97*, 1997.

[Bra93] S. Brands. An efficient off-line electronic cash system based on the representation problem, 1993. CWI Technical Report, CS-R9323.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology–Crypto '98*, pages 13–25, 1998.

[CS99] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *6th ACM Conf. on Computer and Communications Security*, 1999.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Info. Theory*, 22:644–654, 1976.

[DvOW92] W. Diffie, P. van Oorschot, and M. Wiener. Authentication and authenticated key exchange. *Designs, Code, and Cryptography*, 2:107–125, 1992.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology–Crypto '99*, pages 537–554, 1999.

[IZ89] R. Impagliazzo and D. Zuckermann. How to recycle random bits. In *30th Annual Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[Kra94] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology—Crypto '94*, pages 129–139, 1994.

[LL94] C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In *Advances in Cryptology–Crypto '94*, pages 95–107, 1994.

[Lub96] M. Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

[Mau94] U. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology–Crypto '94*, pages 271–281, 1994.

[MW96] U. Maurer and S. Wolf. Diffie-Hellman oracles. In *Advances in Cryptology–Crypto '96*, pages 268–282, 1996.

[NR97] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, 1989.

[RS91] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology–Crypto '91*, pages 433–444, 1991.

[SG98] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Advances in Cryptology–Eurocrypt '98*, 1998.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, 1997.

[Sho99] V. Shoup. On formal models for secure key exchange. IBM Research Report RZ 3120, April 1999.

[Sho00] V. Shoup. A composition theorem for universal one-way hash functions. In *Advances in Cryptology–Eurocrypt 2000*, 2000.

[Sta96] M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology–Eurocrypt '96*, pages 190–199, 1996.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In *PKC '98*, 1998.

[ZS92] Y. Zheng and J. Seberry. Practical approaches to attaining security against adaptively chosen ciphertext attacks. In *Advances in Cryptology–Crypto '92*, pages 292–304, 1992.