Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack

Ronald Cramer
Dept. of Computer Science, Aarhus University
cramer@brics.dk

Victor Shoup
New York University
shoup@cs.nyu.edu

August 14, 2003

Abstract

A new public key encryption scheme, along with several variants, is proposed and analyzed. The scheme and its variants are quite practical, and are proved secure against adaptive chosen ciphertext attack under standard intractability assumptions. These appear to be the first public-key encryption schemes in the literature that are simultaneously practical and provably secure.

This paper is a significantly revised and extended version of the extended abstract "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack" [R. Cramer and V. Shoup, in *Advances in Cryptology – Crypto '98*], and also includes results originally presented in the extended abstract "Using hash functions as a hedge against chosen ciphertext attack" [V. Shoup, in *Advances in Cryptology – Eurocrypt 2000*].

Contents

1 Introduction

In this paper, we present and analyze a new public-key encryption scheme, and several variants, proving that they are secure against adaptive chosen ciphertext attack (as defined by Rackoff and Simon [RS91]) under standard intractability assumptions. The schemes are quite practical, requiring just a few exponentiations in a group for both encryption and decryption. Moreover, the proofs of security of these schemes rely only on standard intractability assumptions: one variant relies only on the hardness of the Decisional Diffie-Hellman problem, while other, somewhat more practical, variants rely on a couple of other standard intractability assumptions.

The hardness of the Decisional Diffie-Hellman problem is essentially equivalent to the semantic security of the basic ElGamal encryption scheme [ElG85]. Thus, with just a bit more computation, we get security against adaptive chosen ciphertext attack, whereas the basic ElGamal scheme is completely insecure against this type of attack.

While there are several provably secure public-key encryption schemes in the literature, they are all quite impractical. Also, there are several practical encryption schemes that have been proposed, but none of them has been proven secure under standard intractability assumptions. The significance of our results is that they provide several schemes that are provably secure and practical at the same time. There appear to be no other public-key encryption schemes in the literature that enjoy both of these properties simultaneously.

This paper is a significantly revised and extended version of the extended abstract [CS98], and also includes results originally presented in the extended abstract [Sho00b].

1.1 Chosen ciphertext security

Semantic security, defined by Goldwasser and Micali [GM84], captures the intuition that an adversary should not be able to obtain any partial information about a message given its encryption. However, this guarantee of secrecy is only valid when the adversary is completely passive, i.e., can only eavesdrop. Indeed, semantic security offers no guarantee of secrecy at all if an adversary can mount an active attack, i.e., inject messages into a network or otherwise influence the behavior of parties in the network.

To deal with active attacks, Rackoff and Simon [RS91] defined the notion of security against an *adaptive chosen ciphertext attack*. If an adversary can inject messages into a network, these messages may be ciphertexts, and the adversary may be able to extract partial information about the corresponding cleartexts through its interactions with the parties in the network. Rackoff and Simon's definition models this type of attack by simply allowing an adversary to obtain decryptions of its choice, i.e., the adversary has access to a "decryption oracle." Now, given an encryption of a message — the "target" ciphertext — we want to guarantee that the adversary cannot obtain any partial information about the message. To achieve this, we have to restrict the adversary's behavior in some way, otherwise the adversary could simply submit the target ciphertext itself to the decryption oracle. The restriction proposed by Rackoff and Simon is the weakest possible: the adversary is not allowed to submit the target ciphertext itself to the oracle; however, it may submit any other ciphertext, including ciphertexts that are related to the target ciphertext.

A different notion of security against active attacks, called *non-malleability*, was proposed by Dolev, Dwork, and Naor [DDN91, DDN00]. Here, the adversary also has access to a decryption oracle, but his goal is not to obtain partial information about the target ciphertext, but rather, to create another encryption of a different message that is related in some interesting way to the original, encrypted message. For example, for a non-malleable encryption scheme, given an encryption of n, it should be infeasible to create an encryption of $n+1$. It turns out that non-malleability and security against adaptive chosen ciphertext attack are equivalent [BDPR98, DDN00].

An encryption scheme secure against adaptive chosen ciphertext attack is a very powerful cryptographic primitive. It is essential in designing protocols that are secure against active adversaries. For example, this primitive is used in protocols for authentication and key exchange [DN96, DDN00, Sho99] and in protocols for escrow, certified e-mail, and more general fair exchange [ASW00]. It is by now generally recognized in the cryptographic research community that security against adaptive chosen ciphertext attack is the "right" notion of security for a general-purpose public-key encryption scheme. This is exemplified by the adoption of Bellare and Rogaway's OAEP scheme [BR94] (a practical but only heuristically secure scheme) as the internet encryption standard RSA PKCS#1 version 2, and for use in the SET protocol for electronic commerce. Another motivation for security against adaptive chosen ciphertext attack is Bleichenbacher's attack [Ble98] on the the widely used SSL key establishment protocol, which is based on RSA PKCS#1 version 1 — Bleichenbacher showed how to break this protocol by mounting a specific chosen ciphertext attack (SSL still uses RSA PKCS#1 version 1, but the protocol has been patched so as to avoid Bleichenbacher's attack).

There are also intermediate notions of security, between semantic security and adaptive chosen ciphertext security. Naor and Yung [NY90] propose an attack model where the adversary has access to the decryption oracle only *prior* to obtaining the target ciphertext, and the goal of the adversary is to obtain partial information about the encrypted message. Naor and Yung called this type of attack a *chosen ciphertext attack*; it has also been called a "lunch-time" or "midnight" attack, and also an *indifferent* chosen ciphertext attack. In this paper, we will use the phrase *adaptive* chosen ciphertext attack for Rackoff and Simon's definition, to distinguish it from Naor and Yung's definition. Also, throughout this paper, unless otherwise specified, by "security" we will always mean "security against adaptive chosen ciphertext attack."

1.2 Previous work

Provably Secure Schemes. Naor and Yung [NY90] presented the first scheme provably secure against lunch-time attacks. Subsequently, Dolev, Dwork, and Naor [DDN91] presented a scheme that is provably secure against adaptive chosen ciphertext attack.

Rackoff and Simon [RS91] present and prove the security of an encryption scheme, but their scheme is actually not a public-key scheme in the traditional sense: in their scheme, *all users* — both senders and receivers — require public keys, and moreover, a trusted center is required to perform certain functions. In contrast, all other schemes mentioned in this paper, including our own, are traditional public-key systems: encryption is a probabilistic function of the message and the receiver's public key, decryption is a function of the ciphertext and the receiver's secret key, and no trusted center is required. This distinction can be important: adding extra system requirements as in the Rackoff and Simon scheme can greatly restrict the range of application of the scheme.

All of the previously known schemes provably secure under standard intractability assumptions are completely impractical (albeit polynomial time), as they rely on general and expensive constructions for non-interactive zero-knowledge proofs. This includes non-standard schemes like Rackoff and Simon's as well.

Practical Schemes. Damgård [Dam91] proposed a practical scheme that he conjectured to be secure against lunch-time attacks; however, this scheme is not known to be provably secure in this sense, and is in fact demonstrably insecure against adaptive chosen ciphertext attack.

Zheng and Seberry [ZS92] proposed practical schemes that are conjectured to be secure against chosen ciphertext attack, but again, no proof based on standard intractability assumptions is known. Lim and Lee [LL93] also proposed practical schemes that were later broken by Frankel and Yung [FY95].

Bellare and Rogaway [BR93, BR94] have presented practical schemes for which they give heuristic proofs of adaptive chosen ciphertext security; namely, they prove security based on the assumption of a one-way trapdoor permutation in an idealized model of computation, the so-called *random oracle* model, wherein a hash function is represented by a random oracle. Actually, it turns out that the proof of security of the OAEP scheme in [BR94] is flawed: as demonstrated in [Sho01], there can be no standard "black box" security proof based on an arbitrary one-way trapdoor permutation. However, the negative result in [Sho01] does not rule out the possibility that OAEP in conjunction with a specific one-way trapdoor permutation scheme is secure. Indeed, it is shown in [Sho01] that OAEP with exponent-3 RSA is secure, and this result is generalized in [FOPS01] to arbitrary-exponent RSA. A new scheme, OAEP+, is also presented in [Sho01], which can be proven secure in the random oracle model, using an arbitrary one-way trapdoor permutation. Further variations of OAEP and OAEP+ are discussed in [Bon01].

Shoup and Gennaro [SG98] also give ElGamal-like schemes that are secure against adaptive chosen ciphertext attack in the random oracle model, and that are also amenable to efficient threshold decryption.

We stress that although a security proof in the random oracle model is of some value, it is still only a heuristic proof. In particular, these types of proofs do not rule out the possibility of breaking the scheme without breaking the underlying intractability assumption. Nor do they even rule out the possibility of breaking the scheme without finding some kind of weakness in the hash function, as shown by Canetti, Goldreich, and Halevi [CGH98].

1.3 Further progress

Subsequent to the publication of the extended abstract [CS98] on which the present paper is based, some further progress in this area has been made. Canetti and Goldwasser [CG99] presented a threshold-decryption variant of our scheme. Also, the authors of the present paper [CS02] have generalized and extended the basic ideas underlying our encryption scheme, obtaining new and quite practical encryption schemes that are secure against adaptive chosen ciphertext attack under different assumptions — one scheme relies on Paillier's Decision Composite Residuosity assumption [Pai99], while the other (somewhat less practical) scheme relies on the classical Quadratic Residuosity assumption.

1.4 Outline of paper

Our paper consists of two parts.

Part 1. In the first part, we take care of a number of preliminaries, after which we present a basic version of our new scheme, along with a few variants. This first part is organized as follows:

§2: We introduce some basic notation that will be used throughout the paper.

§3: We state the formal definition of a public-key encryption scheme and the notion of security against adaptive chosen ciphertext attack. We also discuss some implications of the definition of security that illustrate its utility.

§4: We state the formal definitions of several intractability assumption related to the Discrete Logarithm problem: the Discrete Logarithm assumption, the Computational Diffie-Hellman assumption, and the Decisional Diffie-Hellman assumption. In doing this, we introduce the notion of a *computational group scheme*, which is a general framework that allows us to discuss in an abstract, yet sufficiently concrete way, the different families of groups that may be used in cryptographic applications.

§5: We define the notion of a *target collision resistant hash function*, which is a special type of a *universal one-way hash function*. We will use this primitive in the most efficient variants of our encryption scheme.

§6: We present and analyze the basic version of our encryption scheme, which we call $\mathsf{CS1}$, along with two variants, called $\mathsf{CS1a}$ and $\mathsf{CS1b}$. We prove the security of these schemes based on the Decisional Diffie-Hellman assumption, and the assumption that a given family of hash functions is target collision resistant. We also present and analyze a somewhat less efficient scheme, called $\mathsf{CS2}$, which does not require a target collision resistant hash function.

Part 2. The schemes presented in §6 suffer from two drawbacks. First, the schemes require that plaintexts are, or can be encoded as, group elements, which may significantly restrict the range of application of the encryption scheme and/or the choice of computational group scheme; it would be nice to relax this restriction, allowing plaintexts to be, say, bit strings of arbitrary length. Second, if the Decisional Diffie-Hellman assumption is false, these schemes can be trivially broken; it would be nice if we could provide a second level of defense, so that if Decisional Diffie-Hellman assumption turns out to be false, we have a scheme that still offers some security — even if only heuristically.

It turns out that both of these drawbacks can be dealt with by using a technique called *hybrid encryption*. Basically, a hybrid encryption scheme uses public-key encryption techniques to derive a shared key that is then used to encrypt the actual message using standard symmetric-key techniques. The second part of the paper is devoted to developing the formal theory behind this technique, and to designing and analyzing variations on our basic scheme that utilize this technique. This part is organized as follows:

§7: We lay the theoretical foundations for hybrid encryption. Although most of the ideas in this section appear to be "folklore," they have not been treated rigorously in the literature. In §7.1, we introduce the notion of a *key encapsulation mechanism*, and an appropriate notion of security against adaptive chosen ciphertext attack. A key encapsulation mechanism is like a public-key encryption scheme, except that the encryption algorithm can only be used to generate and encrypt a random bit string of fixed length, which we shall use as a key for a symmetric-key encryption scheme. In §7.2, we state the formal properties of a symmetric-key encryption scheme that we need for use in a hybrid encryption scheme, and discuss some simple constructions based on pseudo-random bit generators and message authentication codes. In §7.3, we prove that an appropriately secure key encapsulation mechanism, combined with an appropriately secure symmetric-key encryption scheme, yields a public-key encryption scheme that is secure against adaptive chosen ciphertext attack.

In what follows, we concentrate exclusively on the problem of constructing secure key encapsulation mechanisms, since the problem of constructing symmetric-key encryption schemes is essentially solved.

§8: We discuss the notion of a secure *key derivation function*, which is a function that should map random group elements to pseudo-random bit strings of given length. A key derivation function is an essential ingredient in our constructions of key encapsulation mechanisms.

§9: We present and analyze a key encapsulation mechanism, CS3, along with two variants, CS3a and CS3b, and prove their security under the Decisional Diffie-Hellman assumption, and also assuming a target collision resistant hash function and a secure key derivation function.

§10: The hybrid encryption scheme obtained from CS3b is by far the most practical of the encryption schemes presented in this paper; moreover, it has other interesting security properties. We show that CS3b is no less secure than a more traditional key encapsulation mechanism that is a hashed variant of ElGamal encryption, which we call HEG. Second, we also show that CS3b is secure in the random oracle model (viewing the key derivation function as a random function), under the weaker *Computational* Diffie-Hellman assumption, and also assuming a target collision resistant hash function. The results in this section show that there is virtually no risk in using scheme CS3b relative to more traditional encryption schemes, while at the same time, CS3b provides a security guarantee that more traditional schemes do not.

We conclude in §11 with a brief summary.

2 Some Preliminaries

2.1 Basic mathematical notation

$\mathbf{Z}$ denotes the ring of integers, $\mathbf{Z}_{\geq 0}$ denotes the set of non-negative integers, and for positive integer k, $\mathbf{Z}_k$ denotes the ring of integers modulo k, and $\mathbf{Z}_k^*$ denotes the corresponding multiplicative group of units.

2.2 Algorithms and probability spaces

We write $\nu \leftarrow \alpha$ to denote the algorithmic action of assigning the value of α to the variable ν.

Let X be a finite probability space, i.e., a probability space on a finite set S. For $\alpha \in S$, we let $\Pr_X[\alpha]$ denote the probability that X assigns to α, and for $S' \subset S$, we let $\Pr_X[S']$ denote the probability that X assigns to S'.

We write $\nu \stackrel{R}{\leftarrow} X$ to denote the algorithmic action of sampling an element of S according to the distribution X, and assigning the result of this sampling experiment to the variable ν. We sometimes write $\nu_1, \ldots, \nu_k \stackrel{R}{\leftarrow} X$ as a shorthand for $\nu_1 \stackrel{R}{\leftarrow} X; \ldots; \nu_k \stackrel{R}{\leftarrow} X$.

For any finite set S, $\mathbf{U}(S)$ denotes the uniform distribution on S. We write $\nu \stackrel{R}{\leftarrow} S$ as a shorthand for $\nu \stackrel{R}{\leftarrow} \mathbf{U}(S)$.

For any probability space X on a finite set S, we denote by $[X]$ the subset of elements of S that are assigned non-zero probability by X, i.e., the "support" of X.

If $X_1, X_2, \ldots, X_k$ are finite probability spaces, and ϕ is a k-ary predicate, then we write

$$\Pr[\phi(\nu_1, \ldots, \nu_k) : \nu_1 \stackrel{R}{\leftarrow} X_1; \ldots; \nu_k \stackrel{R}{\leftarrow} X_k]$$

to denote the probability that $\phi(\nu_1, \ldots, \nu_k)$ holds when ν_1 is sampled from X_1, ν_2 is sampled from X_2, etc. More generally, for $1 \le i \le k$, X_i may be family of finite probability spaces parameterized by $(\nu_1, \ldots, \nu_{i-1})$, and we write

$$\Pr[\phi(\nu_1, \ldots, \nu_k) : \nu_1 \stackrel{R}{\leftarrow} X_1; \nu_2 \stackrel{R}{\leftarrow} X_2(\nu_1); \ldots ; \nu_k \stackrel{R}{\leftarrow} X_k(\nu_1, \ldots, \nu_{k-1})]$$

to denote the probability that $\phi(\nu_1, \ldots, \nu_k)$ holds when ν_1 is sampled from X_1, after which ν_2 is sampled from $X_2(\nu_1)$, and so on. In this case, it is important that $\nu_1, \ldots, \nu_k$ are sampled in the order given.

Similarly, if F is a k-ary function function, then

$$\{F(\nu_1, \ldots, \nu_k) : \nu_1 \stackrel{R}{\leftarrow} X_1; \nu_2 \stackrel{R}{\leftarrow} X_2(\nu_1); \ldots ; \nu_k \stackrel{R}{\leftarrow} X_k(\nu_1, \ldots, \nu_{k-1})\}$$

denotes the probability space defined by sampling ν_1 from X_1, ν_2 from $X_2(\nu_1)$, and so on, and then evaluating the function $F(\nu_1, \ldots, \nu_k)$.

We shall consider polynomial-time probabilistic algorithms A. We shall insist that for all $\lambda \in \mathbf{Z}_{\ge 0}$ and all inputs of length λ, algorithm A *always* halts in time bounded by a polynomial in λ, regardless of the random choices that A may make. In particular, for any input tuple $(\alpha_1, \ldots, \alpha_k)$, the random choices made by A as well as the output of A on this input are finite probability spaces. We denote this output probability space of A for a given input $(\alpha_1, \ldots, \alpha_k)$ by $\mathsf{A}(\alpha_1, \ldots, \alpha_k)$. We stress that $\mathsf{A}(\alpha_1, \ldots, \alpha_k)$ is a *probability space*, and not a *value*. As such, we may write $\nu \stackrel{R}{\leftarrow} \mathsf{A}(\alpha_1, \ldots, \alpha_k)$ to denote the algorithmic action of running A on input $(\alpha_1, \ldots, \alpha_k)$, and assigning the output to the variable ν. When we speak of the "running time" of A, we mean the worst-case running time of A for inputs of a given length.

To exercise the above notation a bit, note that $[\mathsf{A}(\alpha_1, \ldots, \alpha_k)]$ denotes the set of possible outputs of A on input $(\alpha_1, \ldots, \alpha_k)$. For a tertiary predicate ϕ, and polynomial-time probabilistic algorithms A_1 and A_2, and a value α_0,

$$\Pr[\phi(\alpha_0, \alpha_1, \alpha_2) : \alpha_1 \stackrel{R}{\leftarrow} \mathsf{A}_1(\alpha_0); \alpha_2 \stackrel{R}{\leftarrow} \mathsf{A}_2(\alpha_0, \alpha_1)]$$

denotes the probability that $\phi(\alpha_0, \alpha_1, \alpha_2)$ holds when A_1 is run on input α_0, yielding an output α_1, and then A_2 is run on input (α_0, α_1), yielding an output α_2.

For $\lambda \in \mathbf{Z}_{\ge 0}$, $\mathtt{1}^\lambda$ denotes the bit string consisting of λ copies of the bit $\mathtt{1}$. The string $\mathtt{1}^\lambda$ will often be an input to an algorithm: this is a technical device that allows a polynomial-time algorithm to run in time bounded by a polynomial in λ, even if there are no other inputs to the algorithm, or those inputs happen to be very short.

2.3 Statistical distance and negligible functions

Let X and Y be probability spaces on a finite set S. Define the *statistical distance* $\Delta(X, Y)$ *between* X *and* Y as

$$\Delta(X, Y) := \frac{1}{2} \sum_{\alpha \in S} |\Pr_X[\alpha] - \Pr_Y[\alpha]|.$$

One can easily verify that

$$\Delta(X, Y) = \max_{S' \subset S} |\Pr_X[S'] - \Pr_Y[S']|.$$

A function F mapping non-negative integers to non-negative reals is called *negligible* if for all positive numbers c, there exists an integer $\lambda_0(c) \ge 0$ such that for all $\lambda > \lambda_0(c)$, we have $F(\lambda) < 1/\lambda^c$.

3 Secure Public Key Encryption

In this section, we state the basic properties of a public-key encryption scheme, along with the definition of security against adaptive chosen ciphertext attack. Although the notions here are relatively standard, we treat a number of details here that are not often dealt with in the literature. We also discuss some implications of the definition of security that illustrate its utility.

3.1 Public Key Encryption Schemes

A *public-key encryption scheme* PKE consists of the following algorithms:

- A probabilistic, polynomial-time *key generation algorithm* $\mathsf{PKE.KeyGen}$ that on input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, outputs a public key/secret key pair $(\mathsf{PK}, \mathsf{SK})$. The structure of PK and SK depends on the particular scheme.

  For $\lambda \in \mathbf{Z}_{\geq 0}$, we define the probability spaces

  $$\mathsf{PKE.PKSpace}_\lambda := \{\mathsf{PK} : (\mathsf{PK}, \mathsf{SK}) \overset{R}{\leftarrow} \mathsf{PKE.KeyGen}(1^\lambda)\},$$

  and

  $$\mathsf{PKE.SKSpace}_\lambda := \{\mathsf{SK} : (\mathsf{PK}, \mathsf{SK}) \overset{R}{\leftarrow} \mathsf{PKE.KeyGen}(1^\lambda)\}.$$

- A probabilistic, polynomial-time encryption algorithm $\mathsf{PKE.Encrypt}$ that takes as input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a public key $\mathsf{PK} \in [\mathsf{PKE.PKSpace}_\lambda]$, a message m, and outputs a ciphertext ψ.

  A ciphertext is a bit string. The structure of a message may depend on the particular scheme; see below (§3.1.1) for a discussion.

- A deterministic, polynomial-time decryption algorithm $\mathsf{PKE.Decrypt}$ that takes as input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key $\mathsf{SK} \in [\mathsf{PKE.SKSpace}_\lambda]$, a ciphertext ψ, and outputs either a message m or the special symbol reject.

3.1.1 Message spaces

Different public-key encryption schemes might specify different message spaces, and these message spaces might in fact vary with the choice of public key. Let us denote by $\mathsf{PKE.MSpace}_{\lambda,\mathsf{PK}}$ the message space associated with $\lambda \in \mathbf{Z}_{\geq 0}$ and $\mathsf{PK} \in [\mathsf{PKE.PKSpace}_\lambda]$. Although there may be other ways of categorizing message spaces, we shall work with schemes that specify message spaces in one of two ways:

unrestricted message space: $\mathsf{PKE.MSpace}_{\lambda,\mathsf{PK}} = \{0,1\}^*$ for all λ and PK.

restricted message space: $\mathsf{PKE.MSpace}_{\lambda,\mathsf{PK}}$ is a finite set that may depend on λ and PK.

There should be a deterministic, polynomial-time algorithm that on input 1^λ, PK, and α, determines if $\alpha \in \mathsf{PKE.MSpace}_{\lambda,\mathsf{PK}}$.

Clearly, a public-key encryption scheme with an unrestricted message space will be most suitable in a setting where a very general-purpose encryption scheme is required. However, encryption schemes with restricted message spaces can be useful in some settings as well.

3.1.2 Soundness

A public-key encryption scheme should be *sound* in the sense that decrypting an encryption of a message should yield the original message.

Requiring that this *always* holds is a very strong condition which will not be satisfied by many otherwise quite acceptable encryption schemes.

A definition of soundness that is adequate for our purposes runs as follows. Let us say a public key/secret key pair $(\mathsf{PK}, \mathsf{SK}) \in [\mathsf{PKE.KeyGen}(1^\lambda)]$ is *bad* if for some $m \in \mathsf{PKE.MSpace}_{\lambda,\mathsf{PK}}$ and some $\psi \in [\mathsf{PKE.Encrypt}(1^\lambda, \mathsf{PK}, m)]$, we have $\mathsf{PKE.Decrypt}(1^\lambda, \mathsf{SK}, \psi) \neq m$. Then our requirement is that the probability that the key generation algorithm outputs a bad key pair grows negligibly in λ.

One could formulate even weaker notions of soundness that would still be adequate for many applications, but we shall not pursue this here.

3.2 Security against adaptive chosen ciphertext attack

An adversary A in an adaptive chosen ciphertext attack (CCA) is a probabilistic, polynomial-time *oracle query* machine.

The attack game is defined in terms of an interactive computation between the adversary and its *environment*. The adversary's environment responds to the oracle queries made by the adversary: each oracle query response is sampled from a probability space that is a function of the adversary's input and all the previous oracle queries made by the adversary. We require that A runs in time *strictly* bounded by a polynomial in the length of its input, regardless of its probabilistic choices, and regardless of the responses to its oracle queries from its environment.

We now describe the attack game used to define security against adaptive chosen ciphertext attack; that is, we define (operationally) the environment in which A runs. We assume that the input to A is 1^λ for some $\lambda \in \mathbf{Z}_{\geq 0}$.

Stage 1: The adversary queries a *key generation oracle*. The key generation oracle computes $(\mathsf{PK}, \mathsf{SK}) \overset{R}{\leftarrow} \mathsf{PKE.KeyGen}(1^\lambda)$ and responds with PK.

Stage 2: The adversary makes a sequence of calls to a *decryption oracle*.

For each decryption oracle query, the adversary submits a bit string ψ, and the decryption oracle responds with $\mathsf{PKE.Decrypt}(1^\lambda, \mathsf{SK}, \psi)$.

We emphasize that ψ may be an arbitrary bit string, concocted by A in an arbitrary fashion — it certainly need not be an output of the encryption algorithm.

Stage 3: The adversary submits two messages $m_0, m_1 \in \mathsf{PKE.MSpace}_{\lambda,\mathsf{PK}}$ to an *encryption oracle*. In the case of an unrestricted message space, we require that $|m_0| = |m_1|$.

On input m_0, m_1, the encryption oracle computes:

$$\sigma \overset{R}{\leftarrow} \{0, 1\};\ \psi^* \overset{R}{\leftarrow} \mathsf{PKE.Encrypt}(1^\lambda, \mathsf{PK}, m_\sigma);$$

and responds with the "target" ciphertext ψ^*.

Stage 4: The adversary continues to make calls to the decryption oracle, subject only to the restriction that a submitted bit string ψ is not *identical* to ψ^*.

Again, we emphasize that ψ is arbitrary, and may even be computed by A as a function of ψ^*.

Stage 5: The adversary outputs $\hat{\sigma} \in \{0, 1\}$.

We define the *CCA advantage of* A *against* PKE *at* λ, denoted $\mathsf{AdvCCA}_{\mathsf{PKE},\mathsf{A}}(\lambda)$, to be $|\Pr[\sigma = \hat{\sigma}] - 1/2|$ in the above attack game.

We say that PKE is *secure against adaptive chosen ciphertext attack* if

> *for all probabilistic, polynomial-time oracle query machines* A*, the function* $\mathsf{AdvCCA}_{\mathsf{PKE},\mathsf{A}}(\lambda)$ *grows negligibly in* λ*.*

3.3 Alternative characterization of security

In applying the above definition of security, one typically works directly with the quantity

$$\mathsf{AdvCCA}'_{\mathsf{PKE},\mathsf{A}}(\lambda) := |\Pr[\hat{\sigma} = 1 \mid \sigma = 0] - \Pr[\hat{\sigma} = 1 \mid \sigma = 1]|\,.$$

If we view A as a statistical test, then the quantity $\mathsf{AdvCCA}'_{\mathsf{PKE},\mathsf{A}}(\lambda)$ measures A's advantage in distinguishing a game in which m_0 is always encrypted from a game in which m_1 is always encrypted. It is easy to verify that

$$\mathsf{AdvCCA}'_{\mathsf{PKE},\mathsf{A}}(\lambda) = 2 \cdot \mathsf{AdvCCA}_{\mathsf{PKE},\mathsf{A}}(\lambda).$$

We present here a sketch of another characterization of this notion of security that illustrates more fully its utility in reasoning about the security of higher-level protocols. This alternative characterization is a natural, high level, simulation-based definition that in some ways provides a justification for the rather low level, technical definition given above. Our treatment here will be somewhat less formal than elsewhere in this paper.

We start by defining the notion of a *channel*, which is an object that implements the following operations:

- KeyGen — outputs a public key PK.
- $\mathsf{Encrypt}$ — takes as input a message m, and outputs a ciphertext ψ.
- $\mathsf{Decrypt}$ — takes as input a ciphertext ψ, and outputs a message m (possibly a special reject symbol).

Additionally, a channel is parameterized by a security parameter λ.

To initialize a channel, the KeyGen operation is invoked, after which, an arbitrary number of $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ operations may be invoked. A channel may maintain state between invocations of these operations. We shall assume that messages are arbitrary bit strings.

A channel may be implemented in several ways. One way, of course, is to simply "plug in" a public-key encryption scheme. We call such an implementation of a channel a *real channel.* We wish to describe another implementation, which we call an *ideal channel.*

Loosely speaking, an ideal channel acts essentially like a private storage and retrieval service: when an $\mathsf{Encrypt}$ operation is invoked with a message m, the ideal channel generates a corresponding ciphertext ψ without even "looking" at m, and stores the pair (m, ψ) in a table; when a $\mathsf{Decrypt}$ operation is invoked with a ciphertext ψ such that (m, ψ) is in the table for some m, the ideal channel returns the message m. Thus, the "encryption" ψ of a message m is completely independent of m, and essentially plays the role of a "receipt," presentation of which to the $\mathsf{Decrypt}$ operation yields the message m. As such, the $\mathsf{Encrypt}$ operation might be better named Store, and the $\mathsf{Decrypt}$ operation $\mathsf{Retrieve}$.

We now describe the operation of an ideal channel in a bit more detail.

An ideal channel is built using a *channel simulator*. A channel simulator is an object that implements an interface that is identical to that of a channel, except that the $\mathsf{Encrypt}$ operation does not take as input a message, but rather just the *length* of a message.

An ideal channel uses a channel simulator as follows. The KeyGen operation of the ideal channel is implemented directly in terms of the KeyGen operation of the channel simulator. The ideal channel maintains a set S of message/ciphertext pairs (m, ψ) and a set T of ciphertexts, both initially empty.

When the $\mathsf{Encrypt}$ operation of the ideal channel is invoked with input m, the ideal channel invokes the channel simulator with input $|m|$, obtaining a ciphertext ψ. If $\psi \in T$ or $(m', \psi) \in S$ for some m', the ideal channel becomes "broken," and this and all subsequent invocations of either $\mathsf{Encrypt}$ or $\mathsf{Decrypt}$ return a special symbol indicating this; otherwise, the ideal channel adds the pair (m, ψ) to S and returns ψ as the result of the $\mathsf{Encrypt}$ operation.

When the $\mathsf{Decrypt}$ operation of the ideal channel is invoked with input ψ, the ideal channel first checks if $(m, \psi) \in S$ for some m; if so, it simply returns the message m; otherwise, it adds ψ to T, invokes the $\mathsf{Decrypt}$ operation of the channel simulator to obtain m, and returns m.

That completes the description of how an ideal channel is implemented using a channel simulator.

Now we define a notion of security based on the indistinguishability of real and ideal channels for a public-key encryption scheme PKE with an unrestricted message space. Consider a game in which a polynomial-time probabilistic adversary A interacts with an arbitrary number of channels, and at the end of the game, outputs a 0 or 1. We say that PKE is *secure in the sense of channel indistinguishability* if there exists an efficient *channel simulator* such that for the resulting ideal channel, A cannot effectively distinguish between a game played with all real channels and a game played with all ideal channels; i.e., the absolute difference between the probabilities that A outputs a 1 in the two games grows negligibly in the security parameter.

Note that since real channels never become broken, this definition of security implies that ideal channels become broken with only negligible probability.

It straightforward to show that if PKE is secure against adaptive chosen ciphertext attack, then it is also secure in the sense of channel indistinguishability. To prove this, the channel simulator is implemented using the KeyGen and $\mathsf{Decrypt}$ algorithms of PKE, and the $\mathsf{Encrypt}$ operation of the channel simulator on input ℓ simply runs the $\mathsf{Encrypt}$ algorithm of PKE on input 1^ℓ. It can be shown using a standard "hybrid" argument that the resulting ideal channel is indistinguishable from the real channel.

In analyzing a higher-level protocol, one may substitute all real channels by ideal channels. Presumably, it is much more straightforward to then analyze the resulting idealized protocol, since in the idealized protocol, ciphertexts are just "receipts" that are *completely independent* of the corresponding messages. Security implies that any (polynomial-time recognizable) event in the original protocol occurs with essentially the same probability as in the idealized protocol.

3.4 Further discussion

The definition of security we have presented here is from [RS91]. It is called *IND-CCA2* in [BDPR98]. It is known to be equivalent to other notions, such as non-malleability [DDN91, BDPR98, DDN00], which is called *NM-CCA2* in [BDPR98].

There are other, weaker notions of security for a public-key encryption scheme. For example, [NY90] define a notion that is sometimes called *security against indifferent chosen ciphertext attack*, or *security against lunchtime attack*. This definition of security is exactly the same as the one above in §3.2, except that Stage 4 of the attack is omitted — that is, the adversary does not have access to the decryption oracle after it obtains the target ciphertext. While this notion of security may seem natural, it is actually not sufficient in many applications; in particular, in the channel model discussed above in §3.3, one could not allow the interleaving of $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ operations. This notion is called *IND-CCA1* in [BDPR98].

An even weaker notion of security for a public-key encryption scheme is that of *security against a passive attack*, also known as *semantic security.* This definition of security is exactly the same as the one above in §3.2, except that both Stages 2 and 4 of the attack are omitted — that is, the adversary does not have access to the decryption oracle at all. This notion was introduced in [GM84] and is called IND-CPA in [BDPR98]. This notion of security is quite limited: it is only adequate in situations where the adversary only has the power to eavesdrop network traffic, but cannot modify network traffic or otherwise actively participate in a protocol using the encryption scheme.

For a similar, but slightly different, approach to modeling encryption as an "idealized" process, see [Can00]. See also [BBM00] for another generalization of the definition of adaptive chosen ciphertext attack to a setting involving many users and messages.

Another notion of security is that of *plaintext awareness,* introduced in [BR94] and further refined in [BDPR98], where in a certain sense, an adversary who submits a ciphertext for decryption must already "know" the corresponding plaintext, and hence, the decryption oracle does not really help. The notion defined in these papers only makes sense in the random oracle model, but in that model, security in the sense of [BDPR98] implies security against adaptive chosen ciphertext attack. Conversely, while security against adaptive chosen ciphertext attack does not imply that an adversary "knows" the plaintext corresponding to a submitted ciphertext, it does imply that it does not "hurt" to tell him the plaintext, and in this sense, the notion of security against adaptive chosen ciphertexrt attack is probably just as good as any notion of plaintext awareness.

4 Intractability Assumptions Related to the Discrete Logarithm Problem

In this section, we recall the Discrete Logarithm (DL) assumption, the Computational Diffie-Hellman (CDH) assumption, and the Decisional Diffie-Hellman (DDH) assumption. All of these assumptions are formulated with respect to a suitable group G of large prime order q generated by a given element g.

Informally, the DL assumption is this:

given g^x, it is hard to compute x.

Informally, the CDH assumption is this:

given g^x and g^y for random $x, y \in \mathbf{Z}_q$, it is hard to compute g^{xy}.

Informally, the DDH assumption is this:

it is hard to distinguish triples of the form (g^x, g^y, g^z) *for random* $x, y, z \in \mathbf{Z}_q$ *from triples of the form* (g^x, g^y, g^{xy}) *for random* $x, y \in \mathbf{Z}_q$.

It is clear that the DDH assumption is at least as strong as the CDH assumption, which in turn is at least as strong as the DL assumption. The rest of this section is devoted to describing these assumptions more formally, discussing appropriate groups, and discussing some variations and consequences of these assumptions.

4.1 Computational group schemes

To state these intractability assumptions in a general but precise way, and in an appropriate asymptotic setting, we introduce the notion of a *computational group scheme*.

A computational group scheme $\mathcal{G}$ specifies a *sequence* $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ *of group distributions*. For every value of a security parameter $\lambda \in \mathbf{Z}_{\geq 0}$, $\mathbf{S}_\lambda$ is a probability distribution of group descriptions. A group description Γ specifies a finite abelian group $\hat{G}$, along with a prime-order subgroup G, a generator g of G, and the order q of G. We use multiplicative notation for the group operation in $\hat{G}$, and we denote the identity element of $\hat{G}$ by 1_G.

We will write $\Gamma[\hat{G}, G, g, q]$ to indicate that Γ specifies $\hat{G}$, G, g, and q as above. As a simple example of this notation: "for all $\lambda \in \mathbf{Z}_{\geq 0}$, for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we have $g^q = 1_G$."

As usual, mathematical objects like a group description Γ and elements of a group $\hat{G}$ are represented for computational purposes as bit strings bounded in length by a polynomial in λ. The interpretation of these bit strings is up to the algorithms comprising the group scheme (see below). However, we require that the encoding scheme used to represent group elements as bit strings be *canonical*; that is, every element of a group $\hat{G}$ has a *unique* binary encoding.

The group scheme should also provide several algorithms:

- a deterministic, polynomial-time algorithm for computing the group operation that takes as input $\mathtt{1}^\lambda$ for $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, along with $h_1, h_2 \in \hat{G}$, and outputs the group element $h_1 \cdot h_2 \in \hat{G}$;

- a deterministic, polynomial-time algorithm for computing the group inversion operation that takes as input $\mathtt{1}^\lambda$ for $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and $h \in \hat{G}$, and outputs $h^{-1} \in \hat{G}$;

- a deterministic, polynomial-time algorithm that takes as input $\mathtt{1}^\lambda$ for $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and $\alpha \in \{0,1\}^*$, and determines if α is a valid binary encoding of an element of $\hat{G}$;

- a deterministic, polynomial-time algorithm that takes as input $\mathtt{1}^\lambda$ for $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and $h \in \hat{G}$, and determines if $h \in G$;

- a deterministic, polynomial-time algorithm that takes as input $\mathtt{1}^\lambda$ for $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and outputs g and q.

- a probabilistic, polynomial-time approximate sampling algorithm $\hat{S}$ that on input $\mathtt{1}^\lambda$ approximately samples $\mathbf{S}_\lambda$. The distributions $\mathbf{S}_\lambda$ and $\hat{S}(\mathtt{1}^\lambda)$ should be statistically close; that is, the statistical distance $\Delta(\mathbf{S}_\lambda, \hat{S}(\mathtt{1}^\lambda))$ should be a negligible function in λ.

Notice that we do not require that the output distribution $\hat{S}(\mathtt{1}^\lambda)$ of the sampling algorithm is identical to $\mathbf{S}_\lambda$, but only that the distributions have a negligible statistical distance. In particular, not all elements of $[\hat{S}(\mathtt{1}^\lambda)]$ are necessarily valid group descriptions. It would be impractical to require that these two distributions are identical.

Note that the requirement that the group order be easily computable from the group description is not a trivial requirement: it is easy to exhibit groups whose orders are not easy to compute, e.g., subgroups of $\mathbf{Z}_n^*$ for composite n.

The requirement that group elements have unique encodings is also an important, non-trivial requirement. It is easy to exhibit quotient groups in which the problem of computing canonical representatives of residue classes is non-trivial. An example of this is the group underlying Paillier's encryption scheme [Pai99].

Let $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$. The value 1_G may be directly encoded in Γ, but if not, we can always compute it as $g \cdot g^{-1}$.

Although we will not require it, typical group schemes will have the property that for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, the only elements of $\hat{G}$ of order q lie in G. When this is the case, testing whether a given $h \in \hat{G}$ lies in the subgroup G can be implemented by testing if $h^q = 1_G$. However, a group scheme may provide a more efficient subgroup test.

Let $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$. For $a \in G \setminus \{1_G\}$ and $b \in G$, we denote by $\log_a b$ the discrete logarithm of b to the base a; that is, $\log_a b$ is the unique element $x \in \mathbf{Z}_q$ such that $b = a^x$.

As a notational convention, throughout this paper, the letters a–h (and decorated versions thereof) will denote elements of $\hat{G}$, and the letters r–z (and decorated versions thereof) will denote elements of $\mathbf{Z}_q$.

4.2 Examples of appropriate computational group schemes

There are several examples of computational group schemes that are appropriate for cryptographic applications.

Example 1. Let $\ell_1(\lambda)$ and $\ell_2(\lambda)$ be polynomially bounded integer-valued functions in λ, such that $1 < \ell_1(\lambda) < \ell_2(\lambda)$ for all $\lambda \in \mathbf{Z}_{\geq 0}$. It should be the case that the function $2^{-\ell_1(\lambda)}$ is negligible. For a given $\lambda \in \mathbf{Z}_{\geq 0}$, the distribution $\mathbf{S}_\lambda$ is defined as the distribution of triples (q, p, g), where

- q is a random $\ell_1(\lambda)$-bit prime,
- p is a random $\ell_2(\lambda)$-bit prime with $p \equiv 1 \pmod{q}$, and
- g is a random generator of G, the unique subgroup of order q of the cyclic group $\hat{G} = \mathbf{Z}_p^*$.

Elements in $\mathbf{Z}_p^*$ can be encoded canonically as bit strings of length $\ell_2(\lambda)$. Group operations in $\mathbf{Z}_p^*$ are efficiently implemented using arithmetic modulo p, and group inversion is implemented using the extended Euclidean algorithm. To test if an element $(\alpha \bmod p) \in \mathbf{Z}_p^*$ lies in G, we can test if $\alpha^q \equiv 1 \pmod{p}$.

A random generator g of G may be obtained by generating a random element in $\mathbf{Z}_p^*$ and raising it to the power $(p-1)/q$ (repeating if necessary if this yields $(1 \bmod p)$).

The sampling algorithm $\hat{S}$ may use standard, practical algorithms for primality testing that may err with a small probability that grows negligibly in λ. See, e.g., [BS96] for more information on primality testing. Not all elements of $[\hat{S}(\mathtt{1}^\lambda)]$ are valid group descriptions. Moreover, depending on other aspects of the implementation, the distribution on the valid group descriptions may also be slightly skewed away from $\mathbf{S}_\lambda$. In our formulation of various intractability assumptions, it is much more convenient to work with the natural distribution $\mathbf{S}_\lambda$ than the more awkward distribution $\hat{S}(1^\lambda)$.

We should comment the density of primes p such that $p \equiv 1 \pmod{q}$ has never been proven to be sufficiently large to ensure fast termination of the group generation algorithm. Dirichlet's Theorem on primes in arithmetic progressions only applies to the case where q is fixed relative to p. However, provided $\ell_2(\lambda) \geq (2+\delta)\ell_1(\lambda)$ for some fixed $\delta > 0$, for any $\ell_1(\lambda)$-bit prime q, the probability that a random $\ell_2(\lambda)$-bit number of the form $qk+1$ is prime is $\Omega(1/\ell_2(\lambda))$, assuming the Extended Riemann Hypothesis (ERH). This follows from Theorem 8.8.18 in [BS96].

If the density of primes p such that $p \equiv 1 \pmod{q}$ cannot be proven to be sufficiently large to ensure fast termination of the group generation algorithm, even assuming the ERH, it may not be unreasonable to anyway conjecture that this is the case.

Example 2. This is the same as Example 1, except that $p = 2q+1$, where q is a random $\ell_1(\lambda)$-bit prime. Such a prime q is known as a Sophie Germain prime. It is unknown if there exist infinitely many Sophie Germain primes. However, it is conjectured that there are, and specific conjectures on their density have been made [BH62, BH65] that empirically seem to be valid. In particular, it is conjectured that the probability that a random $\ell_1(\lambda)$-bit number is a Sophie Germain prime is $\Omega(1/\ell_1(\lambda)^2)$. If such a density estimate were true, then a simple trial and error method for finding Sophie Germain primes would terminate quickly. See [CS00] for more information on efficiently generating such primes.

Since the subgroup G of $\mathbf{Z}_p^*$ of order q is just the subgroup of quadratic residues, testing if a given element $(\alpha \bmod p) \in \mathbf{Z}_p^*$ lies in G can be performed by computing the Legendre symbol $(\alpha \mid p)$, which is generally much more efficient than computing $\alpha^q \bmod p$.

A nice property of this construction is that the numbers $\{1, \ldots, q\}$ are easily encoded as elements of G. Given $\alpha \in \{1, \ldots, q\}$, we test if $(\alpha \mid p) = 1$, if so, then we encode α as $(\alpha \bmod p) \in G$, and otherwise, we encode α as $(-\alpha \bmod p)$. Given a group element $h = (\alpha \bmod p) \in G$ with $1 \leq \alpha \leq p-1$, we decode h as α if $\alpha \leq q$, and otherwise, we decode h as $p - \alpha$.

This encoding scheme clearly allows us to also easily encode arbitrary bit strings of length $\ell_1(\lambda) - 1$ as elements of G.

Example 3. One can also construct G as a prime order subgroup of an elliptic curve over a finite field. Elliptic curves and their application to cryptography is a very rich field, and we refer the reader to [BSS99] for an introduction and further references. We only note here that some of the same minor technical problems that arose above in Example 1 also arise here; namely, that (1) the known procedures for generating elliptic curves whose orders have a suitably large prime factor are somewhat heuristic, simply because not enough has been proven about how the order of a randomly generated elliptic curve factors into primes, and (2) it is in general not easy to encode arbitrary bit strings of a given length as points on an elliptic curve. We also note that it is fairly easy to generate elliptic curves of prime order so that we do not have to work in a sub-group, i.e., we can take $G = \hat{G}$. This is useful, as then the sub-group test becomes trivial.

4.3 Intractability assumptions

4.3.1 The DL assumption

Let $\mathcal{G}$ be a computational group scheme, specifying a sequence $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ of group distributions.

For all probabilistic, polynomial-time algorithms A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, we define the *DL advantage of* A *against* $\mathcal{G}$ *at* λ as

$$\mathsf{AdvDL}_{\mathcal{G},\mathsf{A}}(\lambda) := \Pr[\ y = x : \Gamma[\hat{G}, G, g, q] \xleftarrow{R} \mathbf{S}_\lambda;\ x \xleftarrow{R} \mathbf{Z}_q;\ y \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, g^x)\ ].$$

The DL assumption for $\mathcal{G}$ is this:

> *For every probabilistic, polynomial-time algorithm* A*, the function* $\mathsf{AdvDL}_{\mathcal{G},\mathsf{A}}(\lambda)$ *is negligible in* λ*.*

4.3.2 The CDH assumption

Let $\mathcal{G}$ be a computational group scheme, specifying a sequence $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ of group distributions.

For all probabilistic, polynomial-time algorithms A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, we define the *CDH advantage of* A *against* $\mathcal{G}$ *at* λ as

$$\mathsf{AdvCDH}_{\mathcal{G},\mathsf{A}}(\lambda) := \Pr[\ c = g^{xy} : \Gamma[\hat{G}, G, g, q] \xleftarrow{R} \mathbf{S}_\lambda;\ x, y \xleftarrow{R} \mathbf{Z}_q;\ c \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, g^x, g^y)\ ].$$

The CDH assumption for $\mathcal{G}$ is this:

> *For every probabilistic, polynomial-time algorithm* A*, the function* $\mathsf{AdvCDH}_{\mathcal{G},\mathsf{A}}(\lambda)$ *is negligible in* λ*.*

For all probabilistic, polynomial-time algorithms A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we define the *CDH advantage of* A *against* $\mathcal{G}$ *at* λ *given* Γ as

$$\mathsf{AdvCDH}_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) := \Pr[\ c = g^{xy} : x \xleftarrow{R} \mathbf{Z}_q;\ y \xleftarrow{R} \mathbf{Z}_q;\ c \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, g^x, g^y)\ ].$$

4.3.3 The DDH assumption

Let $\mathcal{G}$ be a computational group scheme, specifying a sequence $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ of group distributions.

For all $\lambda \in \mathbf{Z}_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we define the sets $\mathcal{D}_{\lambda,\Gamma}$ and $\mathcal{T}_{\lambda,\Gamma}$ as follows:

$$\begin{aligned} \mathcal{D}_{\lambda,\Gamma} &:= \{(g^x, g^y, g^{xy}) \in G^3 : x, y \in \mathbf{Z}_q\}; \\ \mathcal{T}_{\lambda,\Gamma} &:= G^3. \end{aligned}$$

The set $\mathcal{D}_{\lambda,\Gamma}$ is the set of "Diffie-Hellman triples." Also, for $\rho \in G^3$, define $\mathsf{DHP}_{\lambda,\Gamma}(\rho) = 1$ if $\rho \in \mathcal{D}_{\lambda,\Gamma}$, and otherwise, define $\mathsf{DHP}_{\lambda,\Gamma}(\rho) = 0$.

For all 0/1-valued, probabilistic, polynomial-time algorithms A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, we define the *DDH advantage of* A *against* $\mathcal{G}$ *at* λ as

$$\begin{aligned} &\mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}}(\lambda) := \\ &\quad \Big| \Pr[\ \tau = 1 : \Gamma \xleftarrow{R} \mathbf{S}_\lambda;\ \rho \xleftarrow{R} \mathcal{D}_{\lambda,\Gamma};\ \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] - \\ &\qquad \Pr[\ \tau = 1 : \Gamma \xleftarrow{R} \mathbf{S}_\lambda;\ \rho \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma};\ \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] \Big| \end{aligned}$$

The DDH assumption for $\mathcal{G}$ is this:

> *For every probabilistic, polynomial-time, 0/1-valued algorithm* A*, the function* $\mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}}(\lambda)$ *is negligible in* λ*.*

For all 0/1-valued, probabilistic, polynomial-time algorithms A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we define the *DDH advantage of* A *against* $\mathcal{G}$ *at* λ *given* Γ as

$$\begin{aligned}
&\mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) := \\
&\quad \Big| \Pr[\ \tau = 1 : \rho \overset{R}{\leftarrow} \mathcal{D}_{\lambda,\Gamma};\ \tau \overset{R}{\leftarrow} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] - \\
&\qquad \Pr[\ \tau = 1 : \rho \overset{R}{\leftarrow} \mathcal{T}_{\lambda,\Gamma};\ \tau \overset{R}{\leftarrow} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] \Big|
\end{aligned}$$

A minor variation

We will need the following variation on the DDH assumption.

For all $\lambda \in \mathbf{Z}_{\geq 0}$, for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and we define the sets $\mathcal{D}'_{\lambda,\Gamma}$ and $\mathcal{T}'_{\lambda,\Gamma}$ as follows:

$$\begin{aligned}
\mathcal{D}'_{\lambda,\Gamma} &:= \{g^x, g^y, g^{xy} : x, y \in \mathbf{Z}_q, x \neq 0\}; \\
\mathcal{T}'_{\lambda,\Gamma} &:= \{g^x, g^y, g^z : x, y, z \in \mathbf{Z}_q, x \neq 0, z \neq xy\}.
\end{aligned}$$

That is, $\mathcal{D}'_{\lambda,\Gamma}$ is the set of triples $(\hat{g}, a, \hat{a}) \in G^3$, such that $\hat{g} \neq 1_G$ and $\log_g a = \log_{\hat{g}} \hat{a}$, and $\mathcal{T}'_{\lambda,\Gamma}$ is the set of triples $(\hat{g}, a, \hat{a}) \in G^3$, such that $\hat{g} \neq 1_G$ and $\log_g a \neq \log_{\hat{g}} \hat{a}$.

It is easy to verify the following:

$$\Delta(\mathbf{U}(\mathcal{D}_{\lambda,\Gamma}), \mathbf{U}(\mathcal{D}'_{\lambda,\Gamma})) \leq 1/q; \tag{1}$$
$$\Delta(\mathbf{U}(\mathcal{T}_{\lambda,\Gamma}), \mathbf{U}(\mathcal{T}'_{\lambda,\Gamma})) \leq 2/q. \tag{2}$$

For all 0/1-valued, probabilistic, polynomial-time algorithms A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, we define

$$\begin{aligned}
&\mathsf{AdvDDH}'_{\mathcal{G},\mathsf{A}}(\lambda) := \\
&\quad \Big| \Pr[\ \tau = 1 : \Gamma \overset{R}{\leftarrow} \mathbf{S}_\lambda;\ \rho \overset{R}{\leftarrow} \mathcal{D}'_{\lambda,\Gamma};\ \tau \overset{R}{\leftarrow} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] - \\
&\qquad \Pr[\ \tau = 1 : \Gamma \overset{R}{\leftarrow} \mathbf{S}_\lambda;\ \rho \overset{R}{\leftarrow} \mathcal{T}'_{\lambda,\Gamma};\ \tau \overset{R}{\leftarrow} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] \Big|
\end{aligned}$$

For all 0/1-valued, probabilistic, polynomial-time algorithms A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and for all $\Gamma \in [\mathbf{S}_\lambda]$, we define

$$\begin{aligned}
&\mathsf{AdvDDH}'_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) := \\
&\quad \Big| \Pr[\ \tau = 1 : \rho \overset{R}{\leftarrow} \mathcal{D}'_{\lambda,\Gamma};\ \tau \overset{R}{\leftarrow} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] - \\
&\qquad \Pr[\ \tau = 1 : \rho \overset{R}{\leftarrow} \mathcal{T}'_{\lambda,\Gamma};\ \tau \overset{R}{\leftarrow} \mathsf{A}(1^\lambda, \Gamma, \rho)\ ] \Big|
\end{aligned}$$

The inequalities (1) and (2) imply the following:

Lemma 1 *For all 0/1-valued, probabilistic, polynomial-time algorithms* A*, for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, and for all* $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$*,*

$$\Big| \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) - \mathsf{AdvDDH}'_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) \Big| \leq 3/q.$$

In particular, the DDH assumption holds for $\mathcal{G}$ *if and only if for every probabilistic, polynomial-time 0/1-valued algorithm* A*, the function* $\mathsf{AdvDDH}'_{\mathcal{G},\mathsf{A}}(\lambda)$ *is negligible in* λ*.*

Random self-reducibility

In this section, we discuss the random self-reducibility property of the DDH problem, and its implications.

The following lemma states the random self-reducibility property for the DDH problem.

Lemma 2 *There exists a probabilistic, polynomial-time algorithm* RSR *such that for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, for all* $\Gamma \in [\mathbf{S}_\lambda]$*, and for all* $\rho \in \mathcal{T}_{\lambda,\Gamma}$*, the distribution* $\mathsf{RSR}(1^\lambda, \Gamma, \rho)$ *is* $\mathbf{U}(\mathcal{D}_{\lambda,\Gamma})$ *if* $\rho \in \mathcal{D}_{\lambda,\Gamma}$*, and is* $\mathbf{U}(\mathcal{T}_{\lambda,\Gamma})$ *if* $\rho \notin \mathcal{D}_{\lambda,\Gamma}$*.*

This was first observed by Stadler [Sta96], who needed the result to prove the security of a particular protocol, and later by Naor and Reingold [NR97], who also pointed out some of its broader implications.

The algorithm RSR is very simple. Given 1^λ, the group description $\Gamma[\hat{G}, G, g, q]$, and $\rho = (a, b, c) \in G^3$, the algorithm computes $(a', b', c') \in G^3$ as follows:

$$r \xleftarrow{R} \mathbf{Z}_q;\ s \xleftarrow{R} \mathbf{Z}_q;\ t \xleftarrow{R} \mathbf{Z}_q;\ a' \leftarrow a^r g^s;\ b' \leftarrow b g^t;\ c' \leftarrow c^r a^{rt} b^s g^{st}.$$

The implication of this random self-reduction is that if Diffie-Hellman tuples can be efficiently distinguished from random tuples with a non-negligible advantage, then Diffie-Hellman tuples can be efficiently *recognized* with negligible error probability. More formally, we have the following:

Lemma 3 *For every be a 0/1-valued, probabilistic, polynomial-time algorithm* A*, and every polynomial* P *(with integer coefficients, taking positive values on* $\mathbf{Z}_{\geq 0}$*), there exists a 0/1-valued, probabilistic, polynomial-time algorithm* A_1 *such that for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, for all* $\Gamma \in [\mathbf{S}_\lambda]$*, for all* $\rho \in \mathcal{T}_{\lambda,\Gamma}$*, and for all* $\kappa \in \mathbf{Z}_{\geq 0}$*,*

$$\textit{if } \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) \geq 1/P(\lambda), \textit{ then } \Pr[\tau \neq \mathsf{DHP}_{\lambda,\Gamma}(\rho) : \tau \xleftarrow{R} \mathsf{A}_1(1^\lambda, \Gamma, \rho, 1^\kappa)] \leq 2^{-\kappa}.$$

Lemma 3 follows from Lemma 2 using standard "amplification" techniques, making use of standard results on tail inequalities for the binomial distribution (c.f., Section C.5 in [CLRS02]). Given 1^λ, Γ, ρ, and 1^κ, algorithm A_1 invokes algorithm A as a subroutine $O(P(\lambda)^2\kappa)$ times with inputs $(1^\lambda, \Gamma, \rho')$, where each $\rho' \in \mathcal{T}_{\lambda,\Gamma}$ is independently sampled from $\mathsf{RSR}(1^\lambda, \Gamma, \rho)$; additionally, algorithm A_1 has to run algorithm A as a subroutine $O(P(\lambda)^2\kappa)$ times to "calibrate" A, calculating an estimate of

$$\Pr[\ \tau = 1 : \rho' \xleftarrow{R} \mathcal{T}_{\lambda,\Gamma};\ \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \rho')\ ].$$

4.4 Further discussion

The CDH assumption was introduced informally by [DH76]. Since then, there have been many papers that deal with the DL and CDH assumptions, and cryptographic applications based on them. The DDH assumption appears to have first surfaced in the cryptographic literature in [Bra93], although as that paper notes, the DDH assumption is actually needed to prove the security of a number of previously proposed protocols. Indeed, the famous Diffie-Hellman key exchange cannot be proved secure in any reasonable and standard way just based on the CDH assumption: the DDH assumption (or some variant thereof) is required.

The DDH assumption underpins a number of cryptographic applications. See, for example, the work of Stadler [Sta96] on publicly verifiable secret sharing, and the construction by Naor and

Reingold [NR97] of pseudo-random functions. Also, the well-known encryption scheme of ElGamal [ElG85] relies on the DDH for its security against passive attacks (i.e., semantic security).

One variant of the ElGamal scheme is as follows. Let G be a group of prime order q generated by an element g. The public key consists of a group element $h = g^z$, where $z \in \mathbf{Z}_q$ is chosen at random; the secret key is z. To encrypt a message m, where we assume that $m \in G$, we compute

$$u \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ a \leftarrow g^u;\ b \leftarrow h^u;\ c \leftarrow b \cdot m;$$

to form a ciphertext $\psi = (a, c)$. To decrypt such a ciphertext using the secret key, one computes

$$b \leftarrow a^z;\ m \leftarrow c \cdot b^{-1};$$

to obtain the message m.

It is easy to show that the security of this encryption scheme against passive attack is equivalent to the DDH assumption. It is also easy to see that this scheme is completely insecure against adaptive chosen ciphertext attack: if (a, c) is an encryption of $m \in G$, then for any $m' \in G$, $(a, c \cdot m')$ is an encryption of $m \cdot m'$; thus, one can submit $(a, c \cdot m')$ to the decryption oracle, obtaining $m \cdot m'$, from which one then computes m.

There are some very special families of elliptic curves for which the DDH assumption does not hold, but for which the CDH assumption still appears to stand [JN01]. How these results are to be interpreted is a bit unclear. On the one hand, perhaps they cast some doubt on the DDH assumption in general. On the other hand, perhaps they only illustrate that very specially crafted families of elliptic curves may exhibit some surprising security weaknesses, which would seem to counsel against using such special families of elliptic curves for cryptographic applications, and instead, to use generic, randomly generated elliptic curves; indeed, for another special class of elliptic curves, the DL assumption is false [Sma99].

We refer the reader to two excellent surveys [MW00] and [Bon98]. The latter focuses exclusively on the DDH assumption, while the former discusses both the CDH and DDH assumptions. Also see [Sho97], where it is shown that the DDH is hard in a "generic" model of computation.

5 Target Collision Resistant Hash Functions

In this section, we define the notion of a *target collision resistant hash function*, which is a special kind of *universal one-way hash function*, tailored somewhat for our particular application.

We informally summarize this section as follows. We shall be working with a group G of order q, and we want to hash tuples of group elements to elements of $\mathbf{Z}_q$. For this purpose, we will use a family of keyed hash functions, such that given a randomly chosen tuple of group elements and randomly chosen hash function key, it is computationally infeasible to find a different tuple of group elements that hashes to the same value using the given hash key.

5.1 Definitions

Let k be a fixed positive integer, and let $\mathcal{G}$ be a computational group scheme, specifying a sequence $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ of group distributions.

A *k-ary group hashing scheme* HF *associated with* $\mathcal{G}$ specifies two items:

- A family of *key spaces* indexed by $\lambda \in \mathbf{Z}_{\geq 0}$ and $\Gamma \in [\mathbf{S}_\lambda]$. Each such key space is a probability space on bit strings denoted by $\mathsf{HF.KeySpace}_{\lambda,\Gamma}$.

  There must exist a probabilistic, polynomial-time algorithm whose output distribution on input 1^λ and Γ is equal to $\mathsf{HF.KeySpace}_{\lambda,\Gamma}$.

- A family of *hash functions* indexed by $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and $\mathsf{hk} \in [\mathsf{HF.KeySpace}_{\lambda,\Gamma}]$, where each such function $\mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}$ maps a k-tuple $\rho \in G^k$ of group elements to an element of $\mathbf{Z}_q$.

  There must exist a deterministic, polynomial-time algorithm that on input 1^λ, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, $\mathsf{hk} \in [\mathsf{HF.KeySpace}_{\lambda,\Gamma}]$, and $\rho \in G^k$, outputs $\mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\rho)$.

Let A be a probabilistic, polynomial-time algorithm. For $\lambda \in \mathbf{Z}_{\geq 0}$, we define

$$
\begin{aligned}
&\mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}}(\lambda) := \\
&\quad \Pr[\ \rho \in G^k \ \wedge\ \rho \neq \rho^* \ \wedge\ \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\rho^*) = \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\rho) : \\
&\qquad \Gamma[\hat{G}, G, g, q] \xleftarrow{R} \mathbf{S}_\lambda;\ \rho^* \xleftarrow{R} G^k;\ \mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ \rho \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \rho^*, \mathsf{hk})\ ].
\end{aligned}
$$

The target collision resistance (TCR) assumption for HF is this:

> *For every probabilistic, polynomial-time algorithm* A*, the function* $\mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}}(\lambda)$ *is negligible in* λ*.*

It will also be convenient to define the following. Let A be a probabilistic, polynomial-time algorithm. For $\lambda \in \mathbf{Z}_{\geq 0}$ and $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we define

$$
\begin{aligned}
&\mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}}(\lambda \mid \Gamma) := \\
&\quad \Pr[\ \rho \in G^k \ \wedge\ \rho \neq \rho^* \ \wedge\ \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\rho^*) = \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\rho) : \\
&\qquad \rho^* \xleftarrow{R} G;\ \mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ \rho \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \rho^*, \mathsf{hk})\ ].
\end{aligned}
$$

5.2 Further discussion

As already mentioned, our notion of a target collision resistant hash function is a special case of the more general notion of a universal one-way hash function, introduced by Naor and Yung [NY89]. In their presentation, the hash functions mapped bit strings to bit strings, but of course, using appropriate formatting, we can easily make such a function a map from tuples of elements of the group G to elements of $\mathbf{Z}_q$. The notion of security presented in [NY89] was also slightly stronger than ours: in their paper, the first input to the hash function (i.e. the "target" input) is chosen adversarially, but independent of the key of the hash function, whereas in our application, the target input is a random tuple of group elements. Note that our usage of the term "target collision resistance" differs from that in [BR97], where it is used to mean a "non-asymptotic" version of security, but is otherwise identical to the notion of security for universal one-way hash functions.

As was shown in [NY89], universal one-way hash functions can be built from arbitrary one-way permutations. This result was extended by [Rom90], who showed that universal one-way hash functions can be built (albeit less efficiently) from arbitrary one-way functions.

In practice, to build a universal one-way hash function, one can use a dedicated cryptographic hash function, like SHA-1 [SHA95]. Constructions in [BR97] and [Sho00a] show how to build a general-purpose universal one-way hash function using the underlying compression function of SHA-1, assuming the latter is second pre-image collision resistant. In practice, one might simply use SHA-1 directly, without a key — it is not unreasonable to assume that this already satisfies our definition of target collision resistance.

Note that the notion of target collision resistance is both qualitatively and quantitatively weaker than the notion of (full) collision resistance, which is why we prefer to rely on the former rather than the latter. A *collision resistant* hash function is one where it is hard for an adversary to find two different inputs that hash to the same value; the difference between our notion of target collision resistance and collision resistance is that in the former, one of the two inputs is not under the control of the adversary, while in the latter, both inputs are under the control of the adversary. Simon [Sim98], in fact, gives a kind of separation result, which suggests that collision resistance is a strictly stronger notion of security than target collision resistance.

6 The New Encryption Scheme: Basic Version

6.1 Description of the scheme

In this section, we present the basic version, $\mathsf{CS1}$, of our new scheme.

The scheme makes use of a computational group scheme $\mathcal{G}$ as described in §4.1, defining a sequence $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ of distributions of group descriptions, and providing a sampling algorithm $\hat{S}$, where the output distribution $\hat{S}(1^\lambda)$ closely approximates $\mathbf{S}_\lambda$.

The scheme also makes use of a group hashing scheme HF associated with $\mathcal{G}$, as described in §5.

The scheme is described in detail in Figure 1.

Remark 1 Note that this encryption scheme has a *restricted message space*: messages are elements of the group G. This limits to some degree the applicability of the scheme and the choice of group scheme; indeed, if one wants to encrypt arbitrary bit strings of some bounded length, then among the examples of group schemes discussed in §4.2, only Example 2, based on Sophie Germain primes, is suitable.

Remark 2 Note that in step **D2** of the decryption algorithm, we test if a, $\hat{a}$, and c belong to the subgroup G. This test is essential to the security of the scheme. Although some group schemes may provide a more efficient method for performing these tests, in a typical implementation, one may have to compute a^q, $\hat{a}^q$, and c^q, testing that each of these is 1_G.

Remark 3 Note that the key generation algorithm samples a group description Γ from $\hat{S}(1^\lambda)$. However, in describing the encryption scheme, we assume that Γ is a valid group description. With negligible probability (in λ), Γ may not be a valid group description, in which case the behavior of the key generation, encryption, and decryption algorithms is implementation dependent.

Remark 4 It is straightforward to verify that this encryption scheme satisfies the basic requirements that any public key encryption scheme should satisfy, as described in §3.1. In particular, the *soundness* property will always hold when Γ is a valid group description.

Remark 5 Technically speaking, the output ψ of the encryption algorithm is actually a canonical binary encoding of the 4-tuple $(a, \hat{a}, c, d) \in G^4$. In particular, it is critical that for any two ciphertexts $\psi' \neq \psi$, the parsing algorithm in step **D1** of the decryption algorithm should not output the same 4-tuple of group elements.

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\begin{array}{l} \Gamma[\hat{G}, G, g, q] \xleftarrow{R} \hat{S}(1^\lambda);\ \mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma}; \\ w \xleftarrow{R} \mathbf{Z}_q^*;\ x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{R} \mathbf{Z}_q; \\ \hat{g} \leftarrow g^w;\ e \leftarrow g^{x_1}\hat{g}^{x_2};\ f \leftarrow g^{y_1}\hat{g}^{y_2};\ h \leftarrow g^{z_1}\hat{g}^{z_2}; \end{array}$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \hat{g}, e, f, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2)$.

Encryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a public key

$$\mathsf{PK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, \hat{g}, e, f, h) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times G^4,$$

along with a message $m \in G$, compute

E1: $u \xleftarrow{R} \mathbf{Z}_q$;
E2: $a \leftarrow g^u$;
E3: $\hat{a} \leftarrow \hat{g}^u$;
E4: $b \leftarrow h^u$;
E5: $c \leftarrow b \cdot m$;
E6: $v \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(a, \hat{a}, c)$;
E7: $d \leftarrow e^u f^{uv}$;

and output the ciphertext $\psi = (a, \hat{a}, c, d)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q^6,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 4-tuple $(a, \hat{a}, c, d) \in \hat{G}^4$; output reject and halt if ψ is not of this form.
D2: Test if a, $\hat{a}$, and c belong to G; output reject and halt if this is not the case.
D3: Compute $v \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(a, \hat{a}, c)$.
D4: Test if $d = a^{x_1+y_1 v} \cdot \hat{a}^{x_2+y_2 v}$; output reject and halt if this is not the case.
D5: Compute $b \leftarrow a^{z_1}\hat{a}^{z_2}$.
D6: Compute $m \leftarrow c \cdot b^{-1}$, and output m.

Figure 1: The public-key encryption scheme CS1

6.2 Security analysis of the scheme

We shall prove that $\mathsf{CS1}$ is secure against adaptive chosen ciphertext attack if the DDH assumption holds for $\mathcal{G}$ and the TCR assumption holds for HF. However, we wish to state and prove a concrete security reduction. To this end, we need some auxiliary definitions.

Suppose PKE is a public-key encryption scheme that uses a group scheme in the following natural way: on input 1^λ, the key generation algorithm runs the sampling algorithm of the group scheme on input 1^λ, yielding a group description Γ. For a given probabilistic, polynomial-time oracle query machine A, $\lambda \in \mathbf{Z}_{\geq 0}$, and group description Γ, let us define $\mathsf{AdvCCA}_{\mathsf{PKE},\mathsf{A}}(\lambda \mid \Gamma)$ to be A's advantage in an adaptive chosen ciphertext attack where the key generation algorithm uses the given value of Γ, instead of running the sampling algorithm of the group scheme.

For all probabilistic, polynomial-time oracle query machines A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, let $Q_{\mathsf{A}}(\lambda)$ be an upper bound on the number of decryption oracle queries made by A on input 1^λ. We assume that $Q_{\mathsf{A}}(\lambda)$ is a strict bound in the sense that it holds regardless of the probabilistic choices of A, and regardless of the responses to its oracle queries from its environment.

Theorem 1 *If the DDH assumption holds for $\mathcal{G}$ and the TCR assumption holds for* HF*, then* $\mathsf{CS1}$ *is secure against adaptive chosen ciphertext attack.*

In particular, for all probabilistic, polynomial-time oracle query machines A, there exist probabilistic algorithms A_1 and A_2, whose running times are essentially the same as that of A, such that the following holds. For all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we have

$$\mathsf{AdvCCA}_{\mathsf{CS1},\mathsf{A}}(\lambda \mid \Gamma) \leq \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + \mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}_2}(\lambda \mid \Gamma) + (Q_{\mathsf{A}}(\lambda) + 4)/q. \tag{3}$$

The precise running times of algorithms A_1 and A_2 depend a good deal on details of the model of computation and on implementation details, and so we make no attempt to be more precise on this matter.

Before continuing, we state the following simple but useful lemma, which we leave to the reader to verify.

Lemma 4 *Let U_1, U_2, and F be events defined on some probability space. Suppose that the event $U_1 \wedge \neg F$ occurs if and only if $U_2 \wedge \neg F$ occurs. Then $|\Pr[U_1] - \Pr[U_2]| \leq \Pr[F]$.*

To prove Theorem 1, let us fix a probabilistic, polynomial-time oracle query machine A, the value of the security parameter $\lambda \in \mathbf{Z}_{\geq 0}$, and the group description $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$.

The attack game is as described in §3.2. We now describe the relevant random variables to be considered in analyzing the adversary's attack.

Suppose that the public key is $(\Gamma, \mathsf{hk}, \hat{g}, e, f, h)$ and that the secret key is $(\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2)$. Let $w := \log_g \hat{g}$, and define $x, y, z \in \mathbf{Z}_q$ as follows:

$$x := x_1 + x_2 w, \;\; y := y_1 + y_2 w, \;\; z := z_1 + z_2 w.$$

That is, $x = \log_g e$, $y = \log_g f$, and $z = \log_g h$.

As a notational convention, whenever a particular ciphertext ψ is under consideration in some context, the following values are also implicitly defined in that context:

- $a, \hat{a}, b, c, d \in G$, where $\psi = (a, \hat{a}, c, d)$ and $b := a^{z_1}\hat{a}^{z_2}$;

- $u, \hat{u}, v, r, s, t \in \mathbf{Z}_q$, where

$$u := \log_g a,\ \ \hat{u} := \log_{\hat{g}} \hat{a},\ \ v := \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a, \hat{a}, c),\ \ r := \log_g c,\ \ s := \log_g d,$$

and

$$t := x_1 u + y_1 uv + x_2 \hat{u} w + y_2 \hat{u} v w.$$

For the target ciphertext ψ^*, we also denote by $a^*, \hat{a}^*, b^*, c^*, d^* \in G$ and $u^*, \hat{u}^*, v^*, r^*, s^*, t^* \in \mathbf{Z}_q$ the corresponding values.

The probability space defining the attack game is then determined by the following, mutually independent, random variables:

- the coin tosses Coins of A;
- the values $\mathsf{hk}, w, x_1, x_2, y_1, y_2, z_1, z_2$ generated by the key generation algorithm;
- the values $\sigma \in \{0, 1\}$ and $u^* \in \mathbf{Z}_q$ generated by the encryption oracle.

Let $\mathbf{G}_0$ be the original attack game, let $\hat{\sigma} \in \{0, 1\}$ denote the output of A, and let T_0 be the event that $\sigma = \hat{\sigma}$ in $\mathbf{G}_0$, so that $\mathsf{AdvCCA}_{\mathsf{CS1},\mathsf{A}}(\lambda \mid \Gamma) = |\Pr[T_0] - 1/2|$.

Our overall strategy for the proof is as follows. We shall define a sequence $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_\ell$ of modified attack games. Each of the games $\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_\ell$ operates on the same underlying probability space. In particular, the public key and secret key of the cryptosystem, the coin tosses Coins of A, and the hidden bit σ take on *identical* values across all games. Only some of the rules defining how the environment responds to oracle queries differ from game to game. For any $1 \le i \le \ell$, we let T_i be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_i$.

To assist the reader, here is a high level "road map" of the games:

- In game G_1, we modify the encryption oracle so that it uses the secret key to do the encryption, rather than the public key. This change is purely conceptual, and $\Pr[T_1] = \Pr[T_0]$.
- In game $\mathbf{G}_2$, we modify the encryption oracle again, so that the Diffie-Hellman triple $(\hat{g}, a^*, \hat{a}^*)$ is replaced by a random triple. Under the DDH assumption, A will hardly notice, and in particular, $|\Pr[T_2] - \Pr[T_1]|$ will be negligible.
- In game $\mathbf{G}_3$, we modify the decryption oracle, so that it rejects all ciphertexts ψ such that $u \neq \hat{u}$, i.e., such that $(\hat{g}, a, \hat{a})$ is not a Diffie-Hellman triple. We will see that $\Pr[T_2]$ and $\Pr[T_3]$ differ by an amount bounded by $\Pr[R_3]$, where R_3 is the event that a ciphertext is rejected in $\mathbf{G}_3$ that would not have been under the rules of game $\mathbf{G}_2$.
- In game $\mathbf{G}_4$, we modify the encryption oracle again, so that now, the ciphertext is constructed without even looking at either σ, m_0, or m_1. We will see that $1/2 = \Pr[T_4] = \Pr[T_3]$, and that $\Pr[R_4] = \Pr[R_3]$ (where R_4 is defined in the same way as R_3, but with respect to game $\mathbf{G}_4$).
- In game $\mathbf{G}_5$, we add another rejection rule to the decryption oracle, this time rejecting all ciphertexts ψ such that $(a, \hat{a}, c) \neq (a^*, \hat{a}^*, c^*)$, but $v = v^*$. If A were able to produce such a ciphertext, this would represent a collision in the hash function, and so this rejection rule will be applied with negligible probability. Under the TCR assumption, this implies that $|\Pr[R_5] - \Pr[R_4]|$ is negligible (where R_5 is defined in the same way as R_3, but with respect to game $\mathbf{G}_5$). We will also see that $\Pr[R_5]$ is negligible (unconditionally).

Tracing through the above steps, one sees that $|\Pr[T_0] - 1/2|$ is negligible.

We now present the details, but we shall defer the proofs of all lemmas to the end of the proof of the theorem.

Game $\mathbf{G}_1$. We now modify game $\mathbf{G}_0$ to obtain a new game $\mathbf{G}_1$. These two games are identical, except for a small modification to the encryption oracle. Instead of using the encryption algorithm as given to compute the target ciphertext ψ^*, we use a modified encryption algorithm, in which steps **E4** and **E7** are replaced by:

E4′: $b \leftarrow a^{z_1}\hat{a}^{z_2}$;

E7′: $d \leftarrow a^{x_1+y_1v} \cdot \hat{a}^{x_2+y_2v}$.

The change we have made is purely conceptual: the values of b^* and d^* are exactly the same in game $\mathbf{G}_1$ as they were in $\mathbf{G}_0$. Therefore,

$$\Pr[T_1] = \Pr[T_0]. \tag{4}$$

Note that the encryption oracle now makes use of some components of the secret key, which is something the original encryption oracle does not do.

Game $\mathbf{G}_2$. We now modify game $\mathbf{G}_1$ to obtain a new game $\mathbf{G}_2$. We again modify the encryption oracle, replacing step **E3** of the encryption algorithm by

E3′: $\hat{u} \xleftarrow{R} \mathbf{Z}_q \setminus \{u\}$; $\hat{a} \leftarrow \hat{g}^{\hat{u}}$.

Note that whereas in games $\mathbf{G}_0$ and $\mathbf{G}_1$ we had $u^* = \hat{u}^*$, in game $\mathbf{G}_2$, u^* and $\hat{u}^*$ are nearly independent, being subject only to $u^* \neq \hat{u}^*$. However, observe that games $\mathbf{G}_1$ and $\mathbf{G}_2$ are the same, except that in game $\mathbf{G}_1$, the triple $(\hat{g}, a^*, \hat{a}^*)$ is uniformly distributed in $\mathcal{D}'_{\lambda,\Gamma}$, and in game $\mathbf{G}_2$, the triple $(\hat{g}, a^*, \hat{a}^*)$ is uniformly distributed in $\mathcal{T}'_{\lambda,\Gamma}$. Thus, any difference in behavior between these two games immediately yields a statistical test for distinguishing Diffie-Hellman triples from random triples. More precisely, we have:

Lemma 5 *There exists a probabilistic algorithm A_1, whose running time is essentially the same as that of A, such that*

$$|\Pr[T_2] - \Pr[T_1]| \leq \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + 3/q. \tag{5}$$

Game $\mathbf{G}_3$. In this game, we modify the *decryption* oracle in game $\mathbf{G}_2$ to obtain a new game $\mathbf{G}_3$. Instead of using the original decryption algorithm, we modify the decryption algorithm, replacing steps **D4** and **D5** with:

D4′: Test if $\hat{a} = a^w$ and $d = a^{x+yv}$; output reject and halt if this is not the case.

D5′: $b \leftarrow a^z$.

Note that the decryption oracle now make use of w, but does not make use of $x_1, y_2, y_1, y_2, z_1, z_2$, except indirectly through the values x, y, z.

Now, let R_3 be the event that in game $\mathbf{G}_3$, some ciphertext ψ is submitted to the decryption oracle that is rejected in step **D4′** but that would have passed the test in step **D4**.

Note that if a ciphertext passes the test in **D4′**, it would also have passed the test in **D4**.

It is clear that games $\mathbf{G}_2$ and $\mathbf{G}_3$ proceed identically until the event R_3 occurs. In particular, the event $T_2 \wedge \neg R_3$ and $T_3 \wedge \neg R_3$ are identical. So by Lemma 4, we have

$$|\Pr[T_3] - \Pr[T_2]| \leq \Pr[R_3], \tag{6}$$

and so it suffices to bound $\Pr[R_3]$. We introduce auxiliary games $\mathbf{G}_4$ and $\mathbf{G}_5$ below to do this.

Game $\mathbf{G}_4$. This game is identical to game $\mathbf{G}_3$, except for a small modification to the *encryption* oracle. We again modify the algorithm used by the encryption oracle, replacing step **E5** by

E5′: $r \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ c \leftarrow g^r$.

It is clear by construction that

$$\Pr[T_4] = 1/2, \tag{7}$$

since in game $\mathbf{G}_4$, the variable σ is never used at all, and so the adversary's output is independent of σ.

Define the event R_4 to be the event in game $\mathbf{G}_4$ analogous to the event R_3 in game G_3; that is, R_4 is the event that in game $\mathbf{G}_4$, some ciphertext ψ is submitted to the decryption oracle that is rejected in step **D4′** but that would have passed the test in step **D4**.

We show that this modification has no effect; more precisely:

Lemma 6 *We have*

$$\Pr[T_4] = \Pr[T_3], \tag{8}$$

$$\Pr[R_4] = \Pr[R_3]. \tag{9}$$

Game $\mathbf{G}_5$. This game is the same as game $\mathbf{G}_4$, except for the following modification.

We modify the *decryption* oracle so that it applies the following *special rejection rule*: if the adversary submits a ciphertext ψ for decryption at a point in time after the encryption oracle has been invoked, such that $(a, \hat{a}, c) \neq (a^*, \hat{a}^*, c^*)$ but $v = v^*$, then the decryption oracle immediately outputs reject and halts (before executing step **D4′**).

To analyze this game, we define two events.

First, we define the event C_5 to be the event that the decryption oracle in game $\mathbf{G}_5$ rejects a ciphertext using the special rejection rule.

Second, we define the event R_5 to be the event in game $\mathbf{G}_5$ that some ciphertext ψ is submitted to the decryption oracle that is rejected in step **D4′** but that would have passed the test in step **D4**. Note that such a ciphertext is not rejected by the special rejection rule, since that rule is applied before step **D4′** is executed.

Now, it is clear that games $\mathbf{G}_4$ and $\mathbf{G}_5$ proceed identically until event C_5 occurs. In particular, the events $R_4 \wedge \neg C_5$ and $R_5 \wedge \neg C_5$ are identical. So by Lemma 4, we have

$$|\Pr[R_5] - \Pr[R_4]| \leq \Pr[C_5]. \tag{10}$$

Now, if event C_5 occurs with non-negligible probability, we immediately get an algorithm that contradicts the target collision resistance assumption; more precisely:

Lemma 7 *There exists a probabilistic algorithm A_2, whose running time is essentially the same as that of A, such that*

$$\Pr[C_5] \leq \mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}_2}(\lambda \mid \Gamma) + 1/q. \tag{11}$$

Finally, we show that event R_5 occurs with negligible probability, based on purely information-theoretic considerations:

Lemma 8 *We have*

$$\Pr[R_5] \le Q_{\mathsf{A}}(\lambda)/q. \tag{12}$$

The detailed proof of this lemma is presented below. However, the basic idea of the proof runs as follows. For a decryption query ψ, the only information the adversary has about (x_1, x_2, y_1, y_2) are the values of x, y, and possibly s^*, which are linear combinations of (x_1, x_2, y_1, y_2). As we will prove, the value of t, which the adversary must successfully guess in order to make the event R_5 happen, is an independent linear combination of (x_1, x_2, y_1, y_2), and is therefore unpredictable.

Inequality (3) now follows immediately from (4)-(12).

Proofs of Lemmas

To complete the proof of Theorem 1, we now present the proofs of Lemmas 5, 6, 7, and 8.

Proof of Lemma 5. We describe the algorithm A_1 in detail. For a given value of $\lambda \in \mathbf{Z}_{\ge 0}$, it takes as input 1^λ, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and $\rho = (\hat{g}, a^*, \hat{a}^*) \in G^3$.

Algorithm A_1 provides an environment for A, interacting with A as follows.

First, A_1 computes

$$\mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{R} \mathbf{Z}_q;\ e \leftarrow g^{x_1}\hat{g}^{x_2};\ f \leftarrow g^{y_1}\hat{g}^{y_2};\ h \leftarrow g^{z_1}\hat{g}^{z_2};$$

to generate a public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \hat{g}, e, f, h)$ and a secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2)$. It then gives PK to A.

Whenever A submits a ciphertext $\psi = (a, \hat{a}, c, d)$ to the decryption oracle, A_1 simply runs the decryption algorithm, using the secret key SK.

When A submits (m_0, m_1) to the encryption oracle, A_1 computes

$$\sigma \xleftarrow{R} \{0,1\};\ b^* \leftarrow (a^*)^{z_1}(\hat{a}^*)^{z_2};\ c^* \leftarrow b^* \cdot m_\sigma;\ v^* \leftarrow \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a^*, \hat{a}^*, c^*);\ d^* \leftarrow (a^*)^{x_1+y_1v^*}(\hat{a}^*)^{x_2+y_2v^*};$$

and responds with the "ciphertext" $\psi^* = (a^*, \hat{a}^*, c^*, d^*)$.

When A outputs $\hat{\sigma}$ and halts, A_1 outputs 1 if $\sigma = \hat{\sigma}$ and 0 if $\sigma \ne \hat{\sigma}$.

That completes the description of A_1. By construction, it is clear that for fixed λ and $\Gamma \in [\mathbf{S}_\lambda]$,

$$\begin{aligned}\Pr[T_1] &= \Pr[\ \tau = 1 : \rho \xleftarrow{R} \mathcal{D}'_{\lambda,\Gamma};\ \tau \xleftarrow{R} \mathsf{A}_1(1^\lambda, \Gamma, \rho)\ ];\\ \Pr[T_2] &= \Pr[\ \tau = 1 : \rho \xleftarrow{R} \mathcal{T}'_{\lambda,\Gamma};\ \tau \xleftarrow{R} \mathsf{A}_1(1^\lambda, \Gamma, \rho)\ ].\end{aligned}$$

Thus,

$$|\Pr[T_2] - \Pr[T_1]| = \mathsf{AdvDDH}'_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma),$$

and so (5) now follows directly from this and Lemma 1. $\square$

Before continuing, we state and prove a simple but useful lemma. First, some notation: for a field K and positive integers k, n, we denote by $K^{k\times n}$ the set of all $k \times n$ matrices over K, i.e., matrices with k rows and n columns whose entries are in K; for a matrix M, we denote by M^T its transpose.

Lemma 9 *Let k, n be integers with $1 \le k \le n$, and let K be a finite field. Consider a probability space with random variables $\vec{\alpha} \in K^{n\times 1}$, $\vec{\beta} = (\beta_1, \ldots, \beta_k)^T \in K^{k\times 1}$, $\vec{\gamma} \in K^{k\times 1}$, and $M \in K^{k\times n}$, such that $\vec{\alpha}$ is uniformly distributed over $K^{n\times 1}$, $\vec{\beta} = M\vec{\alpha} + \vec{\gamma}$, and for $1 \le i \le k$, the ith rows of M and $\vec{\gamma}$ are determined by $\beta_1, \ldots, \beta_{i-1}$.*

Then conditioning on any fixed values of $\beta_1, \ldots, \beta_{k-1}$ such that the resulting matrix M has rank k, the value of β_k is uniformly distributed over K in the resulting conditional probability space.

Proof. Consider fixed values of $\beta_1, \ldots, \beta_{k-1} \in K$, which determine M and $\vec{\gamma}$, and assume that the matrix M has rank k. For any $\beta_k \in K$, consider the corresponding vector $\vec{\beta} = (\beta_1, \ldots, \beta_k)^T$; there are exactly $|K|^{n-k}$ vectors $\vec{\alpha}$ such that $\vec{\beta} = M\vec{\alpha} + \vec{\gamma}$. Therefore, each possible value $\beta_k \in K$ is equally likely. □

Proof of Lemma 6. Consider the quantity

$$X := (\mathsf{Coins}, \mathsf{hk}, w, x_1, x_2, y_1, y_2, \sigma, u^*, \hat{u}^*)$$

and the quantity z. Note that X and z take on the same values in games $\mathbf{G}_3$ and $\mathbf{G}_4$.

Consider also the quantity r^*. This quantity takes on different values in games $\mathbf{G}_3$ and $\mathbf{G}_4$. For clarity, let us denote these values as $[r^*]_3$ and $[r^*]_4$, respectively.

It is clear by inspection that the events R_3 and T_3 are determined as functions of X, z, and $[r^*]_3$. Also, the events R_4 and T_4 have precisely the same functional dependence on X, z, and $[r^*]_4$. So to prove the lemma, it suffices to show that the distributions of $(X, z, [r^*]_3)$ and $(X, z, [r^*]_4)$ are identical. Observe that by construction, conditioning on any fixed values of X and z, the distribution of $[r^*]_4$ is uniform over $\mathbf{Z}_q$. So it will suffice to show that conditioning on any fixed values of X and z, the distribution of $[r^*]_3$ is also uniform over $\mathbf{Z}_q$.

We have

$$\begin{pmatrix} z \\ [r^*]_3 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w \\ u^* & w\hat{u}^* \end{pmatrix}}_{=:\, M} \cdot \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \log_g m_\sigma \end{pmatrix}.$$

Conditioning only on a fixed value of X, the matrix M is fixed, but the values z_1 and z_2 are still uniformly and independently distributed over $\mathbf{Z}_q$. Observe that $\det(M) = w(\hat{u}^* - u^*) \neq 0$. If we further condition on a fixed value of z, the value of m_σ is fixed, and by Lemma 9, the distribution of $[r^*]_3$ is uniform over $\mathbf{Z}_q$. □

Proof of Lemma 7. Algorithm A_2 provides an environment for A, interacting with A as follows.

Algorithm A_2 takes as input 1^λ, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, $\rho^* = (a^*, \hat{a}^*, c^*) \in G^3$, and $\mathsf{hk} \in [\mathsf{HF.KeySpace}_{\lambda,\Gamma}]$. It first constructs a public key PK and secret key SK for the encryption scheme using the standard key generation algorithm, except that the given values of Γ and hk are used. It also constructs the target ciphertext $\psi^* = (a^*, \hat{a}^*, c^*, d^*)$, where $a^*, \hat{a}^*, c^*$ are the given inputs as above, and where d^* is computed as

$$v^* \leftarrow \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a^*, \hat{a}^*, c^*);\ d^* \leftarrow (a^*)^{x_1+y_1 v^*}(\hat{a}^*)^{x_2+y_2 v^*}.$$

Here, hk is the given input as above, and x_1, y_1, x_2, y_2 are the values taken from the secret key SK as computed above.

Now A_2 interacts with A using the rules of game $\mathbf{G}_5$ for the decryption oracle, and giving A the target ciphertext ψ^* when A invokes the encryption oracle. However, if the decryption oracle ever invokes the special rejection rule in game $\mathbf{G}_5$ for a given ciphertext ψ, algorithm A_2 immediately outputs $(a, \hat{a}, c)$ corresponding to ψ and halts. Also, if the attack terminates without the special rejection rule ever having been invoked, then A_2 also halts (without producing any output).

That completes the description of A_2. If the input $(a^*, \hat{a}^*, c^*)$ to A_2 is sampled uniformly over all triples of group elements, subject to $\log_g a^* \neq \log_{\hat{g}} \hat{a}^*$, then algorithm A_2 succeeds in finding a collision with probability exactly $\Pr[C_5]$. However, in the definition of AdvTCR, the input is sampled from the uniform distribution over all triples, not subject to the above restriction. The bound (11) follows from the fact that the statistical distance between these two input distributions is $1/q$. $\Box$

Proof of Lemma 8. To prove (12), for $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$, let us define $R_5^{(i)}$ to be the event that there is an ith ciphertext submitted to the decryption oracle in game $\mathbf{G}_5$, and that the submitted ciphertext is rejected in step **D4′** but would have passed the test in step **D4**. For $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$, let us define $B_5^{(i)}$ to be the event that the ith decryption oracle query occurs *before* the encryption oracle query, and that the submitted ciphertext passes the test in steps **D1** and **D2** of the decryption oracle. For $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$, let us define $\hat{B}_5^{(i)}$ to be the event that the ith decryption oracle query occurs *after* the encryption oracle query, and that the submitted ciphertext passes the tests in steps **D1** and **D2** of the decryption oracle.

The bound (12) will follow immediately from Lemmas 10 and 11 below. $\Box$

Lemma 10 *Notation as in the proof of Lemma 8. For all* $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$*, we have* $\Pr[R_5^{(i)} | B_5^{(i)}] \leq 1/q$*.*

Proof. Fix $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$. Consider the quantities

$$X := (\mathsf{Coins}, \mathsf{hk}, w, z)$$

and

$$X' := (x, y).$$

The values of X and X' completely determine the behavior of the adversary up until the point when the encryption oracle is invoked, and in particular, they completely determine the event $B_5^{(i)}$. Let us call X and X' *relevant* if the event $B_5^{(i)}$ occurs.

It will suffice to prove that conditioned on any fixed, relevant values of X and X', the probability that $R_5^{(i)}$ occurs is bounded by $1/q$.

Once relevant values of X and X' are fixed, the value ψ of the ith decryption query is also fixed, along with the corresponding values $a, \hat{a}, b, c, d, u, \hat{u}, v, r$, and s.

The test in **D4′** fails if and only if one of the two mutually exclusive conditions ($\hat{a} \neq a^w$) or ($\hat{a} = a^w$ and $d \neq a^{x+yv}$) holds. It is easy to verify that if the second condition holds, then in fact the test in **D4** fails. Thus, if the test in **D4′** fails but that in **D4** passes, it must be the case that $\hat{a} \neq a^w$ and $d = a^{x_1+y_1 v} \hat{a}^{x_2+y_2 v}$. So we only need to consider values of X and X' such that $\hat{a} \neq a^w$. The condition $\hat{a} \neq a^w$ is equivalent to the condition $u \neq \hat{u}$, and the condition $d = a^{x_1+y_1 v} \hat{a}^{x_2+y_2 v}$ is equivalent to the condition $s = t$.

We have

$$\begin{pmatrix} x \\ y \\ t \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ u & \hat{u}w & uv & \hat{u}vw \end{pmatrix}}_{=:\,M} \cdot \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{pmatrix}.$$

Let us first condition only on a fixed value of X, which fixes the first two rows of M, but leaves the values x_1, x_2, y_1, and y_2 still uniformly distributed over $\mathbf{Z}_q$ and mutually independent. Let us further condition on a fixed value of X' such that X and X' are relevant, and that $u \neq \hat{u}$. The third row of M is also fixed, along with the values x, y, and s. It is easy to see by inspection that the rows of M are linearly independent, since $\hat{u} \neq u$ and $w \neq 0$. From this, it follows by Lemma 9 that t is still uniformly distributed over $\mathbf{Z}_q$, but since s is fixed, we have $\Pr[s = t] = 1/q$. $\square$

Lemma 11 *Notation as in the proof of Lemma 8. For all* $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$*, we have* $\Pr[R_5^{(i)} | \hat{B}_5^{(i)}] \leq 1/q$.

Proof. Fix $1 \leq i \leq Q_{\mathsf{A}}(\lambda)$. Consider the quantities

$$X := (\mathsf{Coins}, \mathsf{hk}, w, z, u^*, \hat{u}^*, r^*)$$

and

$$X' := (x, y, s^*).$$

The values of X and X' completely determine the adversary's entire behavior in game $\mathbf{G}_5$, and in particular, they completely determine the event $\hat{B}_5^{(i)}$. Let us call X and X' *relevant* if the event $\hat{B}_5^{(i)}$ occurs.

It will suffice prove that conditioned on any fixed, relevant values of X and X', the probability that $R_5^{(i)}$ occurs is bounded by $1/q$.

Once X and X' are fixed, the value ψ of the ith decryption query is also fixed, along with the corresponding values $a, \hat{a}, b, c, d, u, \hat{u}, v, r$, and s. As in the proof of Lemma 10, it suffices to consider values of X and X' for which $u \neq \hat{u}$, and then to show that $\Pr[s = t] \leq q$. Notice that the value of X determines the value of v^*, and we may also assume that $v \neq v^*$. To see why we may do so, if $v = v^*$, then either $(a, \hat{a}, c) = (a^*, \hat{a}^*, c^*)$, or ψ is rejected by the special rejection rule. In the first case, since $\psi \neq \psi^*$, we must have $d \neq d^*$, but this implies that ψ fails the test in **D4**. In the second case, step **D4′** is not even executed.

We have

$$\begin{pmatrix} x \\ y \\ s^* \\ t \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ u^* & \hat{u}^*w & u^*v^* & \hat{u}^*v^*w \\ u & \hat{u}w & uv & \hat{u}vw \end{pmatrix}}_{=:\,M} \cdot \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \end{pmatrix}.$$

Let us first condition only on a fixed value of X, which fixes the first three rows of M, but leaves the values x_1, x_2, y_1, and y_2 still uniformly distributed over $\mathbf{Z}_q$ and mutually independent. Let us further condition on a fixed value of X' such that X and X' are relevant, and that $u \neq \hat{u}$ and

$v \neq v^*$. The fourth row of M is also fixed, along with the values x, y, s^*, and s. It is easy to see that the rows of M are linearly independent, since

$$\det(M) = w^2(\hat{u} - u)(\hat{u}^* - u^*)(v^* - v) \neq 0.$$

From this, it follows by Lemma 9 that t is still uniformly distributed over $\mathbf{Z}_q$, but since s is fixed, we have $\Pr[s = t] = 1/q$. $\square$

6.3 Two variations

Scheme CS1 was presented because it is in a form that is particularly easy to analyze. We now describe and analyze two variations of the scheme CS1, which we call CS1a and CS1b, that are a bit simpler than CS1, but that require a bit more work to analyze. For both of these schemes, the public key has the same format and indeed, the same probability distribution, as in CS1, and the encryption algorithm is the same as in CS1. The key generation and decryption algorithms are slightly different, however, and are described in detail in Figures 2 and 3.

Remark 6 Scheme CS1a is essentially the same scheme that was originally presented as the "main scheme" in [CS98]. Scheme CS1b is a minor variation of a scheme originally presented in [Sho00b].

Remark 7 Note that in scheme CS1b, we do not have to separately test if $\hat{a}$ belongs to the subgroup G in step **D2′**, since this is already implied by the test in step **D4′**. The test that a and c belong to G may in some cases be implemented by testing if $a^q = 1_G$ and $c^q = 1_G$.

Remark 8 Note also in scheme CS1b, the decryption algorithm has to compute either three or four (if we test if $a^q = 1_G$) powers of a, and possibly one power of c (if we test if $c^q = 1_G$). Special algorithmic techniques [BGMW92, LL94] can be employed to compute these several powers of a significantly faster than computing several powers of different group elements.

Remark 9 In an actual implementation, it is strongly recommended to compute both exponentiations in step **D4′** of CS1b before rejecting the ciphertext, even if the first exponentiation performed already implies that the ciphertext should be rejected. The reason is that if the ciphertext is rejected after just one exponentiation, this may reveal some timing information that could be exploited by an attacker. Indeed, if we reject immediately upon detecting that $\hat{a} \neq a^w$, then based upon timing information, an attacker could use the decryption box as a kind Diffie-Hellman decision oracle. Our formal model of security does not model any notion of time at all, so such attacks fall outside of the model. While some cryptosystems are vulnerable to actual attacks given this type of timing information — notably, Manger's attack [Man01] on RSA PKCS #1 version 2 — we know of no actual timing attack along these lines on CS1b.

Remark 10 For the same reasons as discussed in the previous remark, it is important that any "error code" returned by the decryption algorithm in scheme CS1b not reveal the precise reason why a ciphertext was rejected. Again, we know of no actual "side channel" attack along these lines on CS1b.

Theorem 2 *If the DDH assumption holds for $\mathcal{G}$ and the TCR assumption holds for* HF*, then* CS1a *and* CS1b *are secure against adaptive chosen ciphertext attack.*

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\Gamma[\hat{G}, G, g, q] \stackrel{R}{\leftarrow} \hat{S}(1^\lambda);\ \mathsf{hk} \stackrel{R}{\leftarrow} \mathsf{HF.KeySpace}_{\lambda,\Gamma};$$
$$w \stackrel{R}{\leftarrow} \mathbf{Z}_q^*;\ x_1, x_2, y_1, y_2, z \stackrel{R}{\leftarrow} \mathbf{Z}_q;$$
$$\hat{g} \leftarrow g^w;\ e \leftarrow g^{x_1}\hat{g}^{x_2};\ f \leftarrow g^{y_1}\hat{g}^{y_2};\ h \leftarrow g^z;$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \hat{g}, e, f, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, x_1, x_2, y_1, y_2, z) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q^5,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 4-tuple $(a, \hat{a}, c, d) \in \hat{G}^4$; output reject and halt if ψ is not of this form.
D2: Test if a, $\hat{a}$, and c belong to G; output reject and halt if this is not the case.
D3: Compute $v \leftarrow \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a, \hat{a}, c)$.
D4: Test if $d = a^{x_1+y_1v}\hat{a}^{x_2+y_2v}$; output reject and halt if this is not the case.
D5′: Compute $b \leftarrow a^z$.
D6: Compute $m \leftarrow c \cdot b^{-1}$, and output m.

Figure 2: Key generation and decryption algorithms for $\mathsf{CS1a}$

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\Gamma[\hat{G}, G, g, q] \stackrel{R}{\leftarrow} \hat{S}(1^\lambda);\ \mathsf{hk} \stackrel{R}{\leftarrow} \mathsf{HF.KeySpace}_{\lambda,\Gamma};$$
$$w \stackrel{R}{\leftarrow} \mathbf{Z}_q^*;\ x, y, z \stackrel{R}{\leftarrow} \mathbf{Z}_q;$$
$$\hat{g} \leftarrow g^w;\ e \leftarrow g^x;\ f \leftarrow g^y;\ h \leftarrow g^z;$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \hat{g}, e, f, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, w, x, y, z)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, x, y, z) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q^3,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 4-tuple $(a, \hat{a}, c, d) \in \hat{G}^4$; output reject and halt if ψ is not of this form.
D2′: Test if a and c belong to G; output reject and halt if this is not the case.
D3: Compute $v \leftarrow \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a, \hat{a}, c)$.
D4′: Test if $\hat{a} = a^w$ and $d = a^{x+yv}$; output reject and halt if this is not the case.
D5′: Compute $b \leftarrow a^z$.
D6: Compute $m \leftarrow c \cdot b^{-1}$, and output m.

Figure 3: Key generation and decryption algorithms for $\mathsf{CS1b}$

In particular, for all probabilistic, polynomial-time oracle query machines A*, for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, and all* $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$*, we have*

$$|\mathsf{AdvCCA}_{\mathsf{CS1a},\mathsf{A}}(\lambda \mid \Gamma) - \mathsf{AdvCCA}_{\mathsf{CS1},\mathsf{A}}(\lambda \mid \Gamma)| \leq Q_{\mathsf{A}}(\lambda)/q \tag{13}$$

and

$$|\mathsf{AdvCCA}_{\mathsf{CS1b},\mathsf{A}}(\lambda \mid \Gamma) - \mathsf{AdvCCA}_{\mathsf{CS1},\mathsf{A}}(\lambda \mid \Gamma)| \leq Q_{\mathsf{A}}(\lambda)/q. \tag{14}$$

To prove this theorem, let us fix A, λ, and $\Gamma[\hat{G}, G, g, q]$. Consider the attack game $\mathbf{G}_0$ as defined in §6.2: this is game that A plays against the scheme $\mathsf{CS1}$ for the given values of λ and Γ. We adopt all the notational conventions established at the beginning of §6.2 (i.e., prior to the description of game $\mathbf{G}_1$).

We begin by defining two modifications of game $\mathbf{G}_0$.

Game $\mathbf{G}_{-1a}$. In this game, we modify the decryption oracle so that in place of step **D5**, we execute step **D5′** as in the scheme $\mathsf{CS1a}$. We emphasize that in game $\mathbf{G}_{-1a}$, we have $z = z_1 + z_2 w$, where w, z_1, and z_2 are generated by the key generation algorithm of $\mathsf{CS1}$.

Game $\mathbf{G}_{-1b}$. In this game, we modify the decryption oracle so that in place of steps **D4** and **D5**, we execute steps **D4′** and **D5′** as in the scheme $\mathsf{CS1b}$. We emphasize that in game $\mathbf{G}_{-1b}$, we have $x = x_1 + x_2 w$, $y = x_1 + x_2 w$, and $z = z_1 + z_2 w$, where w, x_1, x_2, y_1, y_2, z_1, and z_2 are generated by the key generation algorithm of $\mathsf{CS1}$.

Let T_{-1a} be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_{-1a}$ and T_{-1b} be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_{-1b}$.

We remind the reader that games $\mathbf{G}_0$, $\mathbf{G}_{-1a}$, and $\mathbf{G}_{-1b}$ all operate on the *same* underlying probability space: all of the variables

$$\mathsf{Coins}, \mathsf{hk}, w, x_1, x_2, y_1, y_2, z_1, z_2, \sigma, u^*$$

that ultimately determine the events T_0, T_{-1a}, and T_{-1b} have the same values in games $\mathbf{G}_0$, $\mathbf{G}_{-1a}$, and $\mathbf{G}_{-1b}$; all that changes is the functional behavior of the decryption oracle.

It is straightforward to verify that and that

$$\mathsf{AdvCCA}_{\mathsf{CS1a},\mathsf{A}}(\lambda \mid \Gamma) = |\Pr[T_{-1a} - 1/2]|$$

and

$$\mathsf{AdvCCA}_{\mathsf{CS1b},\mathsf{A}}(\lambda \mid \Gamma) = |\Pr[T_{-1b} - 1/2]|.$$

Let us define the event R_{-1b} to be the event that some ciphertext is rejected in game $\mathbf{G}_{-1b}$ in step **D4′** that would have passed the test in **D4**. It is clear that games $\mathbf{G}_0$, $\mathbf{G}_{-1a}$, and $\mathbf{G}_{-1b}$ all proceed identically until event R_{-1b} occurs. In particular, we the events $T_0 \wedge \neg R_{-1b}$, $T_{-1a} \wedge \neg R_{-1b}$, and $T_{-1b} \wedge \neg R_{-1b}$ are identical. So by Lemma 4, we have

$$|\Pr[T_0] - \Pr[T_{-1a}]| \leq \Pr[R_{-1b}]$$

and

$$|\Pr[T_0] - \Pr[T_{-1b}]| \leq \Pr[R_{-1b}].$$

So it suffices to show that

$$\Pr[R_{-1b}] \leq Q_{\mathsf{A}}(\lambda)/q. \tag{15}$$

To do this, for $1 \le i \le Q_{\mathsf{A}}(\lambda)$, let $R_{-1b}^{(i)}$ be the event that there is an ith ciphertext submitted to the decryption oracle in game $\mathbf{G}_{-1b}$, and that this ciphertext is rejected in step $\mathbf{D4'}$, but would have passed the test in step **D4**.

The bound (15) will follow immediately from the following lemma.

Lemma 12 *For all $1 \le i \le Q_{\mathsf{A}}(\lambda)$, we have $\Pr[R_{-1b}^{(i)}] \le 1/q$.*

Proof. The proof of this is lemma is almost identical to that of Lemma 10. Note that in game $\mathbf{G}_{-1b}$, the encryption oracle uses the "real" encryption algorithm, and so itself does not leak any additional information about (x_1, x_2, y_1, y_2). This is in contrast to game $\mathbf{G}_5$, where the encryption oracle does leak additional information.

Fix $1 \le i \le Q_{\mathsf{A}}(\lambda)$. Consider the quantities

$$X := (\mathsf{Coins}, \mathsf{hk}, w, z, \sigma, u^*).$$

and

$$X' := (x, y).$$

The values of X and X' completely determine the adversary's entire behavior in game $\mathbf{G}_5$, and hence determine if there is an ith decryption oracle query, and if so, the value of the corresponding ciphertext. Let us call X and X' *relevant* if for these values of X and X', there is an ith decryption oracle query, and the corresponding ciphertext passes steps **D1** and **D2**.

It will suffice to prove that conditioned on any fixed, relevant values of X and X', the probability that $R_{-1b}^{(i)}$ occurs is bounded by $1/q$.

The remainder of the argument is *exactly* as in Lemma 10, except using X, X', and the notion of *relevant* as defined here. $\Box$

6.4 A hash-free variant

Our basic scheme $\mathsf{CS1}$ requires a target collision resistant hash function. Qualitatively, the TCR assumption is much weaker than the DDH assumption, since one can build a target collision resistant hash function based on an arbitrary one-way function. Indeed, one can build a collision resistant hash function under the DL assumption; however, the hash functions arising from such a construction produce an output that is in G, whereas we need a hash function that maps into $\mathbf{Z}_q$. We cannot in general expect to find an easy-to-compute, injective map from G onto $\mathbf{Z}_q$; in Example 2 in §4.2, we in fact do have such a map, but that is an exceptional case.

For these reasons, we present a variation $\mathsf{CS2}$ of our basic scheme that does not require a hash function.

This scheme requires a family $\{\mathsf{Chop}_{\lambda,\Gamma}\}$ of "chopping" functions associated with the group scheme $\mathcal{G}$ with the following properties. For $\lambda \in \mathbf{Z}_{\ge 0}$ and $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, the function $\mathsf{Chop}_{\lambda,\Gamma}$ *injectively* maps triples $\rho \in G^3$ of group elements to N-tuples $(v_1, \ldots, v_N) \in \mathbf{Z}_q^N$. Here, $N = N(\lambda, \Gamma)$ is bounded by a polynomial in λ, and the function $\mathsf{Chop}_{\lambda,\Gamma}$ should be computable by a deterministic, polynomial-time function that takes inputs 1^λ, Γ, and ρ.

In principle, such chopping functions always exist, since we can write down the binary representation of ρ, and chop it into bit strings of length $\lfloor \log_2 q \rfloor$.

We present the details of scheme $\mathsf{CS2}$ in Figure 4.

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\begin{aligned}
&\Gamma[\hat{G}, G, g, q] \xleftarrow{R} \hat{S}(1^\lambda);\\
&w \xleftarrow{R} \mathbf{Z}_q^*;\ x_1, x_2, z_1, z_2 \xleftarrow{R} \mathbf{Z}_q;\\
&\text{for } i = 1, \ldots, N\text{: } y_1^{(i)}, y_2^{(i)} \xleftarrow{R} \mathbf{Z}_q;\\
&\hat{g} \leftarrow g^w;\ e \leftarrow g^{x_1}\hat{g}^{x_2};\ h \leftarrow g^{z_1}\hat{g}^{z_2};\\
&\text{for } i = 1, \ldots, N\text{: } f_i \leftarrow g^{y_1^{(i)}}\hat{g}^{y_2^{(i)}};
\end{aligned}$$

and output the public key $\mathsf{PK} = (\Gamma, \hat{g}, e, (f_i)_{i=1}^N, h)$ and the secret key $\mathsf{SK} = (\Gamma, x_1, x_2, (y_1^{(i)}, y_2^{(i)})_{i=1}^N, z_1, z_2)$.

Encryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a public key

$$\mathsf{PK} = (\Gamma[\hat{G}, G, g, q], \hat{g}, e, (f_i)_{i=1}^N, h) \in [\mathbf{S}_\lambda] \times G^{N+3},$$

along with a message $m \in G$, compute

E1: $u \xleftarrow{R} \mathbf{Z}_q$;
E2: $a \leftarrow g^u$;
E3: $\hat{a} \leftarrow \hat{g}^u$;
E4: $b \leftarrow h^u$;
E5: $c \leftarrow b \cdot m$;
E6: $(v_1, \ldots, v_N) \leftarrow \mathsf{Chop}_{\lambda,\Gamma}(a, \hat{a}, c)$;
E7: $d \leftarrow e^u \prod_{i=1}^N f_i^{uv_i}$;

and output the ciphertext $\psi = (a, \hat{a}, c, d)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], x_1, x_2, (y_1^{(i)}, y_2^{(i)})_{i=1}^N, z_1, z_2) \in [\mathbf{S}_\lambda] \times \mathbf{Z}_q^{N+4},$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 4-tuple $(a, \hat{a}, c, d) \in \hat{G}^4$; output reject and halt if ψ is not of this form.
D2: Test if a, $\hat{a}$, and c belong to G; output reject and halt if this is not the case.
D3: Compute $(v_1, \ldots, v_N) \leftarrow \mathsf{Chop}_{\lambda,\Gamma}(a, \hat{a}, c)$.
D4: Test if $d = a^{x_1 + \sum_{i=1}^N y_1^{(i)} v_i} \cdot \hat{a}^{x_2 + \sum_{i=1}^N y_2^{(i)} v_i}$; output reject and halt if this is not the case.
D5: Compute $b \leftarrow a^{z_1}\hat{a}^{z_2}$.
D6: Compute $m \leftarrow c \cdot b^{-1}$, and output m.

Figure 4: The public-key encryption scheme $\mathsf{CS2}$, where $N = N(\lambda, \Gamma)$

Theorem 3 *If the DDH assumption holds for $\mathcal{G}$, then* CS2 *is secure against adaptive chosen ciphertext attack.*

In particular, for all probabilistic, polynomial-time oracle query machines A*, there exists a probabilistic algorithm* A_1*, whose running time is essentially the same as that of* A*, such that the following holds. For all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we have*

$$\mathsf{AdvCCA}_{\mathsf{CS2},\mathsf{A}}(\lambda \mid \Gamma) \leq \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + (Q_\mathsf{A}(\lambda) + 3)/q.$$

The proof of this theorem follows the same lines as the proof of Theorem 1. We present here a sketch of the proof, appealing in several places to arguments found in the proof of Theorem 1 so as to avoid repeating arguments that are identical or nearly identical.

Let us fix a probabilistic, polynomial-time oracle query machine A, the value of the security parameter $\lambda \in \mathbf{Z}_{\geq 0}$, and the group description $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$.

We define $x, z \in \mathbf{Z}_q$ as follows:

$$x := x_1 + x_2 w, \;\; z := z_1 + z_2 w.$$

We also define $y^{(i)} \in \mathbf{Z}_q$, for $1 \leq i \leq N$, as

$$y^{(i)} := y_1^{(i)} + y_2^{(i)} w.$$

As a notational convention, whenever a particular ciphertext ψ is under consideration in some context, the following values are also implicitly defined in that context:

- $a, \hat{a}, c, d \in G$, where $\psi = (a, \hat{a}, c, d)$;
- $u, \hat{u}, v_1, \ldots, v_N, r, s \in \mathbf{Z}_q$, where

$$u := \log_g a, \;\; \hat{u} := \log_{\hat{g}} \hat{a}, \;\; (v_1, \ldots, v_N) := \mathsf{Chop}_{\lambda,\Gamma}(a, \hat{a}, c), \;\; r := \log_g c, \;\; s := \log_g d.$$

For the target ciphertext ψ^*, we also denote by $a^*, \hat{a}^*, c^*, d^* \in G$ and $u^*, \hat{u}^*, v_1^*, \ldots, v_N^*, r^*, s^* \in \mathbf{Z}_q$ the corresponding values.

The probability space defining the attack game is then determined by the following, mutually independent, random variables:

- the coin tosses of A;
- the values $w, x_1, x_2, y_1^{(1)}, \ldots, y^{(N)}, y_2^{(1)}, \ldots, y_2^{(N)}, z_1, z_2$ generated by the key generation algorithm;
- the values $\sigma \in \{0, 1\}$ and $u^* \in \mathbf{Z}_q$ generated by the encryption oracle.

Let $\mathbf{G}_0$ be the original attack game, let $\hat{\sigma} \in \{0, 1\}$ denote the output of A, and let T_0 be the event that $\sigma = \hat{\sigma}$ in $\mathbf{G}_0$, so that $\mathsf{AdvCCA}_{\mathsf{CS2},\mathsf{A}}(\lambda \mid \Gamma) = |\Pr[T_0] - 1/2|$.

As in the proof of Theorem 1, we shall define a sequence of modified games $\mathbf{G}_i$, for $i = 1, 2, \ldots$, and in game $\mathbf{G}_i$, the event T_i will be the event corresponding to event T_0, but in game $\mathbf{G}_i$. We remind the reader that all of these games operate on the same underlying probability space, and except as otherwise specified, random variables have identical values between games.

Game $\mathbf{G}_1$. In game $\mathbf{G}_1$, we modify the algorithm used by the encryption oracle as follows. Steps **E4** and **E7** are replaced by:

E4′: $b \leftarrow a^{z_1}\hat{a}^{z_2}$;

E7′: $d \leftarrow a^{x_1+\sum_{i=1}^{N} y_1^{(i)} v_i} \cdot \hat{a}^{x_2+\sum_{i=1}^{N} y_2^{(i)} v_i}$.

By the same reasoning as in the proof of Theorem 1, we have $\Pr[T_1] = \Pr[T_0]$.

Game $\mathbf{G}_2$. We again modify the encryption oracle, replacing step **E3** by

E3′: $\hat{u} \xleftarrow{R} \mathbf{Z}_q \setminus \{u\};\ \hat{a} \leftarrow \hat{g}^{\hat{u}}$.

By the same reasoning as in the proof of Theorem 1, one sees that there exists a probabilistic algorithm A_1, whose running time is essentially the same as that of A, such that

$$|\Pr[T_2] - \Pr[T_1]| \leq \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + 3/q.$$

Game $\mathbf{G}_3$. In this game, we modify the decryption oracle in game $\mathbf{G}_2$, replacing steps **D4** and **D5** with:

D4′: Test if $\hat{a} = a^w$ and $d = a^{x+\sum_{i=1}^{N} y^{(i)} v_i}$; output reject and halt if this is not the case.

D5′: $b \leftarrow a^z$.

Let R_3 be the event that in game $\mathbf{G}_3$, some ciphertext ψ is submitted to the decryption oracle that is rejected in step **D4′** but that would have passed the test in step **D4**.

As in the proof of Theorem 1, we have

$$|\Pr[T_3] - \Pr[T_2]| \leq \Pr[R_3].$$

We claim that

$$\Pr[R_3] \leq Q_{\mathsf{A}}(\lambda)/q.$$

We can prove the analog of Lemma 8 (in game $\mathbf{G}_5$ in the proof of Theorem 1) by considering an $(N+3) \times (2N+2)$ matrix M over $\mathbf{Z}_q$ defined as

$$M := \begin{pmatrix} 1 & w & & & & & \\ & & 1 & w & & & \\ & & & & \ddots & & \\ & & & & & 1 & w \\ u^* & \hat{u}^* w & u^* v_1^* & \hat{u}^* v_1^* w & \cdots & u^* v_N^* & \hat{u}^* v_N^* w \\ u & \hat{u} w & u v_1 & \hat{u} v_1 w & \cdots & u v_N & \hat{u} v_N w \end{pmatrix},$$

where $w \neq 0$, $\hat{u} \neq u$, $\hat{u}^* \neq u^*$, and $v_i \neq v_i^*$ for some $i \in \{1, \ldots, N\}$. It will suffice to show that the rows of M are linearly independent.

If we choose i such that $v_i \neq v_i^*$, and consider the 4×4 sub-matrix M' of M consisting of the intersection of columns 1, 2, $2i+1$, $2i+2$ of M, and rows 1, $i+1$, $N+2$, $N+3$ of M, we see that matrix M' has the same form as the matrix considered in Lemma 11, and hence is non-singular. It follows that the rows of M are linearly independent, since any non-trivial linear relation among the rows of M implies a non-trivial linear relation among the rows of M'.

Game $\mathbf{G}_4$. We again modify the algorithm used by the encryption oracle, replacing step **E5** by

E5′: $r \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ c \leftarrow g^r.$

By reasoning analogous to that in game $\mathbf{G}_4$ in the proof of Theorem 1, one can show that

$$\Pr[T_4] = \Pr[T_3].$$

Moreover, by construction it is evident that

$$\Pr[T_4] = 1/2.$$

That completes the proof sketch of Theorem 3. We leave it to the reader to work out the details of the design and analysis of variants CS2a and CS2b of scheme CS2, corresponding to the variants CS1a and CS1b of scheme CS1, which were discussed in §6.3.

Remark 11 Note that the high-level structure of the proof of Theorem 3 is significantly simpler than that of Theorem 1. In particular, in the analysis of game $\mathbf{G}_3$ in the proof of Theorem 3, we were able to bound the quantity $\Pr[R_3]$ directly, without deferring the analysis to a later game, as in the proof of Theorem 1. This simplification comes from the fact that we do not have to deal with a target collision resistant hash function in Theorem 3, as we did in Theorem 1. Indeed, if in the scheme CS1 we use a collision resistant hash function, we could prove the security of CS1 using a proof with essentially the same line of reasoning as that of the proof of Theorem 3, with one extra game between $\mathbf{G}_0$ and $\mathbf{G}_1$ to effectively ban hash function collisions.

7 Hybrid Encryption

The encryption schemes presented in the previous section all had restricted message spaces. In some settings, an encryption scheme with an unrestricted message space is more desirable. A simple and efficient way to build an encryption scheme that has an unrestricted message is to build a *hybrid* encryption scheme. Loosely speaking, such a scheme uses public-key encryption techniques to encrypt a key K that is then used to encrypt the actual message using symmetric-key encryption techniques. In this section, we develop the necessary tools for building a hybrid public-key encryption scheme.

One key ingredient in any hybrid scheme is a *key encapsulation mechanism.* This is like a public-key encryption scheme, except that the job of the encryption algorithm is to generate the encryption of a random key K. Of course, one can always use a general-purpose public-key encryption scheme to do this, by simply generating K at random, and then encrypting it. However, there are typically more efficient ways to do this.

As a quick example of a key encapsulation mechanism, consider the following variation of the ElGamal encryption scheme. Let G be a group of prime order q generated by an element g. Let H be a cryptographic hash function, such as SHA-1. The public key consists of a group element $h = g^z$, where $z \in \mathbf{Z}_q$ is chosen at random; the secret key is z. To generate an encryption of a symmetric key K, we compute

$$u \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ a \leftarrow g^u;\ b \leftarrow h^u;\ K \leftarrow H(b);$$

to form a ciphertext $\psi = a$. To decrypt a ciphertext $\psi = a$ using the secret key, one computes

$$b \leftarrow a^z;\ K \leftarrow H(b);$$

obtaining a symmetric key K.

To build a complete hybrid encryption scheme, we combine a key encapsulation mechanism with a symmetric-key encryption scheme.

7.1 Key encapsulation

A key encapsulation mechanism KEM consists of the following algorithms:

- A probabilistic, polynomial-time *key generation algorithm* $\mathsf{KEM.KeyGen}$ that on input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, outputs a public key/secret key pair $(\mathsf{PK}, \mathsf{SK})$. The structure of PK and SK depends on the particular scheme.

  For $\lambda \in \mathbf{Z}_{\geq 0}$, we define the probability spaces

  $$\mathsf{KEM.PKSpace}_\lambda := \{\mathsf{PK} : (\mathsf{PK}, \mathsf{SK}) \stackrel{R}{\leftarrow} \mathsf{KEM.KeyGen}(1^\lambda)\},$$

  and

  $$\mathsf{KEM.SKSpace}_\lambda := \{\mathsf{SK} : (\mathsf{PK}, \mathsf{SK}) \stackrel{R}{\leftarrow} \mathsf{KEM.KeyGen}(1^\lambda)\}.$$

- A probabilistic, polynomial-time encryption algorithm $\mathsf{KEM.Encrypt}$ that takes as input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, and a public key $\mathsf{PK} \in [\mathsf{KEM.PKSpace}_\lambda]$, and outputs a pair (K, ψ), where K is a *key* and ψ is a ciphertext.

  A key K is a bit string of length $\mathsf{KEM.KeyLen}(\lambda)$, where $\mathsf{KEM.KeyLen}(\lambda)$ is another parameter of the key encapsulation mechanism.

  A ciphertext is a bit string.

- A deterministic, polynomial-time decryption algorithm $\mathsf{KEM.Decrypt}$ that takes as input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key $\mathsf{SK} \in [\mathsf{KEM.SKSpace}_\lambda]$, a ciphertext ψ, and outputs either a key K or the special symbol reject.

7.1.1 Soundness

As for public key encryption, we need an appropriate notion of soundness. A definition of soundness that is adequate for our purposes runs as follows. Let us say a public key/secret key pair $(\mathsf{PK}, \mathsf{SK}) \in [\mathsf{KEM.KeyGen}(1^\lambda)]$ is *bad* if for some $(K, \psi) \in [\mathsf{KEM.Encrypt}(1^\lambda, \mathsf{PK})]$, we have $\mathsf{KEM.Decrypt}(1^\lambda, \mathsf{SK}, \psi) \neq K$. Let $\mathsf{BadKeyPair}_{\mathsf{KEM}}(\lambda)$ denote the probability that the key generation algorithm generates a bad key pair for a given value of λ. Then our requirement is that $\mathsf{BadKeyPair}_{\mathsf{KEM}}(\lambda)$ grows negligibly in λ.

7.1.2 Security against adaptive chosen ciphertext attack

For a key encapsulation mechanism, an adversary A in an adaptive chosen ciphertext attack is a probabilistic, polynomial-time oracle query machine that takes as input 1^λ, where $\lambda \in \mathbf{Z}_{\geq 0}$ is the security parameter. We now describe the attack game used to define security against adaptive chosen ciphertext attack, which is quite similar to that used to define the corresponding notion of security for a public-key encryption scheme.

Stage 1: The adversary queries a *key generation oracle*. The key generation oracle computes $(\mathsf{PK}, \mathsf{SK}) \stackrel{R}{\leftarrow} \mathsf{KEM.KeyGen}(1^\lambda)$ and responds with PK.

Stage 2: The adversary makes a sequence of calls to a *decryption oracle*.

For each decryption oracle query, the adversary submits a ciphertext ψ, and the decryption oracle responds with $\mathsf{KEM.Decrypt}(1^\lambda, \mathsf{SK}, \psi)$.

Stage 3: The adversary queries an *encryption oracle*.

The encryption oracle computes:

$$\begin{array}{l} (K^*, \psi^*) \stackrel{R}{\leftarrow} \mathsf{KEM.Encrypt}(1^\lambda, \mathsf{PK});\ K^+ \stackrel{R}{\leftarrow} \{0,1\}^\ell;\ \tau \stackrel{R}{\leftarrow} \{0,1\}; \\ \text{if } \tau = 0 \text{ then } K^\dagger \leftarrow K^* \text{ else } K^\dagger \leftarrow K^+; \end{array}$$

where $\ell := \mathsf{KEM.KeyLen}(\lambda)$, and responds with the pair $(K^\dagger, \psi^*)$.

Stage 4: The adversary continues to make calls to the decryption oracle, subject only to the restriction that a submitted ciphertext ψ is not *identical* to ψ^*.

Stage 5: The adversary outputs $\hat{\tau} \in \{0,1\}$.

We define $\mathsf{AdvCCA}_{\mathsf{KEM},\mathsf{A}}(\lambda)$ to be $|\Pr[\tau = \hat{\tau}] - 1/2|$ in the above attack game.
We say that KEM is *secure against adaptive chosen ciphertext attack* if

for all probabilistic, polynomial-time oracle query machines A*, the function* $\mathsf{AdvCCA}_{\mathsf{KEM},\mathsf{A}}(\lambda)$ *grows negligibly in* λ*.*

In applying the above definition of security, one typically works directly with the quantity

$$\mathsf{AdvCCA}'_{\mathsf{KEM},\mathsf{A}}(\lambda) := |\Pr[\hat{\tau} = 1 \mid \tau = 0] - \Pr[\hat{\tau} = 1 \mid \tau = 1]|\,.$$

It is easy to verify that

$$\mathsf{AdvCCA}'_{\mathsf{KEM},\mathsf{A}}(\lambda) = 2 \cdot \mathsf{AdvCCA}_{\mathsf{KEM},\mathsf{A}}(\lambda).$$

7.2 One-time symmetric-key encryption

A *one-time symmetric-key encryption scheme* SKE consists of two algorithms:

- A deterministic, polynomial-time encryption algorithm $\mathsf{SKE.Encrypt}$ that takes as input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a key K, and a message m, and outputs a ciphertext χ.

  The key K is a bit string of length $\mathsf{SKE.KeyLen}(\lambda)$.

  Here, $\mathsf{SKE.KeyLen}(\lambda)$ is a parameter of the encryption scheme, which we assume can be computed in deterministic polynomial time given 1^λ.

  The message m is a bit string of arbitrary, unbounded length.

  The ciphertext χ is a bit string.

  We denote by $\mathsf{SKE.CTLen}(\lambda, \ell)$ the maximum length of any encryption of a message of length at most ℓ, when using security parameter λ.

- A deterministic, polynomial-time decryption algorithm $\mathsf{SKE.Decrypt}$ that takes as input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a key K, and a ciphertext χ and outputs a message m or the special symbol reject.

  The key K is a bit string of length $\mathsf{SKE.KeyLen}(\lambda)$.

  The ciphertext χ is a bit string of arbitrary length.

We require that SKE satisfy the following *soundness* condition: for all $\lambda \in \mathbf{Z}_{\geq 0}$, for all $K \in \{0,1\}^{\mathsf{SKE.KeyLen}(\lambda)}$, for all $m \in \{0,1\}^*$, we have:

$$\mathsf{SKE.Decrypt}(1^\lambda, K, \mathsf{SKE.Encrypt}(1^\lambda, K, m)) = m.$$

7.2.1 Two definitions of security

We define two notions of security for a one-time symmetric-key encryption scheme: security against passive attacks, and security against adaptive chosen ciphertext attacks.

As usual, an adversary A is a probabilistic, polynomial-time oracle query machine that takes as input 1^λ, where $\lambda \in \mathbf{Z}_{\geq 0}$ is the security parameter.

A *passive* attack runs as follows. The adversary A chooses two messages, m_0 and m_1, of equal length, and gives these to an *encryption oracle*. The encryption oracle generates a random key K of length $\mathsf{SKE.KeyLen}(\lambda)$, along with random $\sigma \in \{0,1\}$, and encrypts the message m_σ using the key K. The adversary A is then given the resulting ciphertext χ^*. Finally, the adversary outputs $\hat{\sigma} \in \{0,1\}$.

We define $\mathsf{AdvPA}_{\mathsf{SKE},\mathsf{A}}(\lambda)$ to be $|\Pr[\sigma = \hat{\sigma}] - 1/2|$ in the above attack game.

We say that SKE is *secure against passive attacks* if

> *for all probabilistic, polynomial-time oracle query machines* A*, the function* $\mathsf{AdvPA}_{\mathsf{SKE},\mathsf{A}}(\lambda)$ *grows negligibly in* λ*.*

An *adaptive chosen ciphertext attack* is exactly the same as a passive attack, except that *after* the adversary A obtains the target ciphertext χ^* from the encryption oracle, the adversary may then query a *decryption oracle* any number of times. In each decryption oracle query, A submits a ciphertext $\chi \neq \chi^*$, and obtains the decryption of χ under the key K. As in the passive attack, A outputs $\hat{\sigma} \in \{0,1\}$.

We define $\mathsf{AdvCCA}_{\mathsf{SKE},\mathsf{A}}(\lambda)$ to be $|\Pr[\sigma = \hat{\sigma}] - 1/2|$ in the above attack game.

We say that SKE is *secure against adaptive chosen ciphertext attacks* if

> *for all probabilistic, polynomial-time oracle query machines* A*, the function* $\mathsf{AdvCCA}_{\mathsf{SKE},\mathsf{A}}(\lambda)$ *grows negligibly in* λ*.*

7.2.2 Constructions

Our definition of a symmetric-key encryption scheme and the corresponding notions of security are tailored to the application of building a hybrid public-key encryption scheme. These definitions may not be appropriate for other settings. In particular, our definitions of security do not imply protection against chosen plaintext attack; however, this protection is not needed for hybrid public-key encryption schemes, since a symmetric key is only used to encrypt a single message.

It is easy to build a symmetric key encryption scheme that achieves security against passive attacks using standard symmetric-key techniques. For example, to encrypt a message m, one can expand the key K using a pseudo-random bit generator to obtain a "one time pad" α of length $|m|$, and then compute $\chi \leftarrow m \oplus \alpha$.

A pseudo-random bit generator can be built from an arbitrary one-way permutation [GL89], or even from an arbitrary one-way function [ILL89, HILL99]. These constructions, however, are not very practical. In a practical implementation, it is perfectly reasonable to stretch the key K by using it as the key to a dedicated block cipher, and then evaluate the block cipher at successive points (so-called "counter mode") to obtain a sequence of pseudo-random bits (c.f. [MvOV97, Chapter 7]).

Note that the above construction yields a scheme that is completely insecure against adaptive chosen ciphertext attack. However, it is also easy to build a symmetric key encryption scheme $\mathsf{SKE2}$ that achieves security against adaptive chosen ciphertext attack, given an arbitrary scheme $\mathsf{SKE1}$ that is only secure against passive attacks.

One technique is to simply build an $\mathsf{SKE2}$ ciphertext by attaching a *message authentication code* to the $\mathsf{SKE1}$ ciphertext. Although this technique seems to be "folklore," for completeness, we develop the details here.

A *one-time message authentication code* MAC specifies the following items:

- For $\lambda \in \mathbf{Z}_{\geq 0}$, a *key length* parameter $\mathsf{MAC.KeyLen}(\lambda)$ and an *output length* parameter $\mathsf{MAC.OutLen}(\lambda)$.

  We assume that $\mathsf{MAC.KeyLen}(\lambda)$ can be computed in deterministic polynomial time given 1^λ.

- A family of functions indexed by $\lambda \in \mathbf{Z}_{\geq 0}$ and $\mathsf{mk} \in \{0,1\}^{\mathsf{MAC.KeyLen}(\lambda)}$, where each function $\mathsf{MAC}^\lambda_{\mathsf{mk}}$ maps arbitrary bit strings to bit strings of length exactly $\mathsf{MAC.OutLen}(\lambda)$.

  There must be a deterministic, polynomial-time algorithm that on input 1^λ, $\mathsf{mk} \in \{0,1\}^{\mathsf{MAC.KeyLen}(\lambda)}$, and $\alpha \in \{0,1\}^*$, outputs $\mathsf{MAC}^\lambda_{\mathsf{mk}}(\alpha)$.

To define security for MAC, we define an attack game as follows. As usual, an adversary A is a probabilistic, polynomial-time oracle query machine that takes as input 1^λ, where $\lambda \in \mathbf{Z}_{\geq 0}$ is the security parameter. The adversary A first chooses a bit string α, and submits this to an oracle. The oracle generates a random key mk of length $\mathsf{MAC.KeyLen}(\lambda)$, computes $\beta \leftarrow \mathsf{MAC}^\lambda_{\mathsf{mk}}(\alpha)$, and returns β to the adversary. The adversary A then outputs a list

$$((\alpha_1, \beta_1), \ldots, (\alpha_k, \beta_k))$$

of pairs of bit strings. We say that A has *produced a forgery* if for some $1 \leq i \leq k$, we have $\alpha_i \neq \alpha$ and $\mathsf{MAC}^\lambda_{\mathsf{mk}}(\alpha_i) = \beta_i$.

We say that A is a $(L_1(\lambda), L_2(\lambda), N(\lambda))$ forging adversary if $|\alpha| \leq L_1(\lambda)$, $k \leq N(\lambda)$, and $|\alpha_i| \leq L_2(\lambda)$ for all $1 \leq i \leq k$.

Define $\mathsf{AdvForge}_{\mathsf{MAC},\mathsf{A}}(\lambda)$ to be the probability that A produces a forgery in the above game. We say that MAC is secure if

> *for all probabilistic, polynomial-time oracle query machines* A*, the function* $\mathsf{AdvForge}_{\mathsf{MAC},\mathsf{A}}(\lambda)$ *grows negligibly in* λ.

Message authentication codes have been extensively studied (c.f. [MvOV97, Chapter 9]). Once can easily build secure one-time message authentication codes using an appropriate family of universal hash functions, without relying on any intractability assumptions. There are also other ways to build message authentication codes which may be preferable in practice, even though the security of these schemes is not fully proven.

Now we show how to use $\mathsf{SKE1}$ and MAC to build $\mathsf{SKE2}$. The key length $\mathsf{SKE2.KeyLen}(\lambda)$ of $\mathsf{SKE2}$ will be equal to

$$\mathsf{SKE1.KeyLen}(\lambda) + \mathsf{MAC.KeyLen}(\lambda).$$

We will write such a key as (K, mk), where K is a bit string of length $\mathsf{SKE1.KeyLen}(\lambda)$, and mk is a bit string of length $\mathsf{MAC.KeyLen}(\lambda)$.

To encrypt a message m under a key (K, mk) as above, algorithm $\mathsf{SKE2.Encrypt}$ computes

$$\chi \leftarrow \mathsf{SKE1.Encrypt}(1^\lambda, K, m);\ \mathsf{tag} \leftarrow \mathsf{MAC}^\lambda_{\mathsf{mk}}(\chi);\ \chi' \leftarrow \chi \,\|\, \mathsf{tag};$$

and outputs the ciphertext χ'.

To decrypt a ciphertext χ' under a key (K, mk) as above, algorithm $\mathsf{SKE2.Decrypt}$ first parses χ' as $\chi' = \chi \,\|\, \mathsf{tag}$, where tag is a bit string of length $\mathsf{MAC.OutLen}(\lambda)$. If this parsing step fails (because χ' is too short), then the algorithm outputs reject; otherwise, it computes

$$\mathsf{tag}' \leftarrow \mathsf{MAC}^\lambda_{\mathsf{mk}}(\chi).$$

If $\mathsf{tag} \neq \mathsf{tag}'$, the algorithm outputs reject; otherwise, it computes

$$m \leftarrow \mathsf{SKE1.Decrypt}(1^\lambda, K, \chi);$$

and outputs m.

To analyze the security of $\mathsf{SKE2}$, we recall that for all probabilistic, polynomial-time oracle query machines A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, we denote by $Q_\mathsf{A}(\lambda)$ an upper bound on the number of decryption oracle queries made by A on input 1^λ. Although we introduced this notation in the context of public-key encryption, we can adopt it here in the context of symmetric-key encryption as well. We remind the reader that $Q_\mathsf{A}(\lambda)$ should be a strict bound that holds for any environment.

For all probabilistic, polynomial-time oracle query machines A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, we define $B_\mathsf{A}(\lambda)$ to be an upper bound on the length of the messages submitted by A to the encryption oracle, and $B'_\mathsf{A}(\lambda)$ to be an upper bound on the ciphertexts submitted by A to the decryption oracle. As usual, these upper bounds should hold regardless of the environment of A.

Theorem 4 *If* $\mathsf{SKE1}$ *is secure against passive attacks, and* MAC *is a secure one-time message authentication code, then* $\mathsf{SKE2}$ *is secure against adaptive chosen ciphertext attacks.*

In particular, for every probabilistic, polynomial-time oracle query machine A*, there exist probabilistic oracle query machine* A_1 *and* A_2*, whose running times are essentially the same as that of* A*, such that for all* $\lambda \in \mathbf{Z}_{\geq 0}$*,*

$$\mathsf{AdvCCA}_{\mathsf{SKE2},\mathsf{A}}(\lambda) \leq \mathsf{AdvPA}_{\mathsf{SKE1},\mathsf{A}_1}(\lambda) + \mathsf{AdvForge}_{\mathsf{MAC},\mathsf{A}_2}(\lambda).$$

Moreover, A_2 *is a*

$$(\mathsf{SKE1.CTLen}(\lambda, B(\lambda)),\ B'(\lambda) - \mathsf{MAC.OutLen}(\lambda),\ Q_\mathsf{A}(\lambda))$$

forging adversary.

Proof. Fix A and λ, and let $\mathbf{G}_0$ denote the original chosen ciphertext attack game. Let T_0 be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_0$.

We next define a modified attack game $\mathbf{G}_1$, in which all ciphertexts submitted to the decryption oracle by A in game $\mathbf{G}_1$ are simply rejected.

Let T_1 be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_1$. Let R_1 be the event in game $\mathbf{G}_1$ that some ciphertext is rejected in game $\mathbf{G}_1$ that would not have been rejected under the rules of game $\mathbf{G}_0$.

Since games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically until event R_1 occurs, the events $T_0 \wedge \neg R_1$ and $T_1 \wedge \neg R_1$ are identical, and so by Lemma 4, we have $|\Pr[T_0] - \Pr[T_1]| \leq \Pr[R_1]$.

It is straightforward to verify that

$$\Pr[R_1] \leq \mathsf{AdvForge}_{\mathsf{MAC},\mathsf{A}_2}(\lambda) \tag{16}$$

for an adversary A_2 as described above.

The theorem now follows by observing that the attack by A in game $\mathbf{G}_1$ is now a passive attack. That is, the adversary A_1 in the theorem simply runs the adversary A, and whenever A makes a decryption oracle query, adversary A_1 simply lets A continue *as if* the decryption oracle rejected the ciphertext. □

Remark 12 Although the keys for $\mathsf{SKE2}$ are longer than those for $\mathsf{SKE1}$, this need not be the case if we use a pseudo-random bit generator to stretch a short key into a suitably long key. Indeed, the key length of any symmetric key encryption scheme need be no longer than the key length of a secure a pseudo-random bit generator.

7.3 A hybrid construction

Let KEM be a key encapsulation mechanism (as defined in §7.1) and let SKE be a one-time symmetric key encryption scheme (as defined in §7.2). Further, let us assume that the two schemes are *compatible* in the sense that for all $\lambda \in \mathbf{Z}_{\geq 0}$, we have $\mathsf{KEM.KeyLen}(\lambda) = \mathsf{SKE.KeyLen}(\lambda)$. We now describe a hybrid public-key encryption scheme HPKE.

The key generation algorithm for HPKE is the same as that of KEM, and the public and secret keys are the same as those of KEM.

To encrypt a message m in the hybrid scheme, we run $\mathsf{KEM.Encrypt}$ to generate a symmetric key K and a ciphertext ψ encrypting K. We then encrypt m under the key K using $\mathsf{SKE.Encrypt}$, obtaining a ciphertext χ. The output of the encryption algorithm is $\hat{\psi} = (\psi, \chi)$, encoded in a canonical fashion as a bit string.

The decryption algorithm for the hybrid scheme runs as follows. Given a ciphertext $\hat{\psi}$, we first verify that $\hat{\psi}$ properly encodes a pair (ψ, χ). If not, we output reject and halt. Next, we decrypt ψ using $\mathsf{KEM.Decrypt}$; if this yields reject, then we output reject and halt. Otherwise, we obtain a symmetric key K and decrypt χ under K using $\mathsf{SKE.Decrypt}$, and output the resulting decryption (which may be reject).

Theorem 5 *If* KEM *and* SKE *are secure against adaptive chosen ciphertext attacks, then so is* HPKE.

In particular, if A *is a probabilistic, polynomial-time oracle query machine, then there exist probabilistic oracle query machines* A_1 *and* A_2*, whose running times are essentially the same as that of* A*, such that for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, we have*

$$\mathsf{AdvCCA}_{\mathsf{HPKE},\mathsf{A}}(\lambda) \leq \mathsf{BadKeyPair}_{\mathsf{KEM}}(\lambda) + \mathsf{AdvCCA}'_{\mathsf{KEM},\mathsf{A}_1}(\lambda) + \mathsf{AdvCCA}_{\mathsf{SKE},\mathsf{A}_2}(\lambda).$$

Proof. Fix A and λ, and let $\mathbf{G}_0$ be the original chosen ciphertext attack game played by A against HPKE. We let $\hat{\psi}^* = (\psi^*, \chi^*)$ denote the target ciphertext; σ is the hidden bit generated by the encryption oracle and $\hat{\sigma}$ is the bit output by A. Let T_0 be the event that $\sigma = \hat{\sigma}$. Also, let K^* denote the symmetric key output by the algorithm $\mathsf{KEM.Encrypt}$ during the encryption process within the encryption oracle.

We now define a modified game $\mathbf{G}_1$. In this game, whenever a ciphertext (ψ, χ) is submitted to the decryption oracle after the invocation of the encryption oracle, if $\psi = \psi^*$ (but $\chi \neq \chi^*$ of course), then the decryption oracle does not apply algorithm $\mathsf{KEM.Decrypt}$ to obtain the symmetric key, but instead just uses the key K^* produced by the encryption oracle. Let T_1 be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_1$.

This change is slightly more than just conceptual, since $\mathsf{KEM.KeyGen}$ may generate a bad key pair with probability $\mathsf{BadKeyPair}_{\mathsf{KEM}}(\lambda)$. However, unless this occurs, games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically, and so by Lemma 4, we have

$$|\Pr[T_1] - \Pr[T_0]| \leq \mathsf{BadKeyPair}_{\mathsf{KEM}}(\lambda).$$

Now we define a modified game $\mathbf{G}_2$. This game behaves just like game $\mathbf{G}_1$, except that we use a completely random symmetric key K^+ in place of the key K^* in both the encryption and decryption oracles. Let T_2 be the event that $\sigma = \hat{\sigma}$ in game $\mathbf{G}_2$.

It is straightforward to see that there is an oracle query machine A_1, whose running time is essentially the same as that of A, such that

$$|\Pr[T_2] - \Pr[T_1]| = \mathsf{AdvCCA}'_{\mathsf{KEM},\mathsf{A}_1}(\lambda).$$

The adversary A_1 basically just runs the adversary A. In the attack game that A_1 is playing against KEM, the value $K^\dagger$ is equal to K^* in game $\mathbf{G}_1$, and is equal to K^+ in game $\mathbf{G}_2$. Note that in games $\mathbf{G}_1$ and $\mathbf{G}_2$, the ciphertext ψ^* is never explicitly decrypted, and so A_1 need not submit this for decryption either.

Lastly, we observe that there is an oracle query machine A_2, whose running time is essentially the same as that of A, such that

$$|\Pr[T_2] - 1/2| = \mathsf{AdvCCA}_{\mathsf{SKE},\mathsf{A}_2}(\lambda).$$

To see this, note that in game $\mathbf{G}_2$, the ciphertext χ^* is produced using the random symmetric encryption key K^+, and also that some other ciphertexts $\chi \neq \chi^*$ are decrypted using K^+, but that the key K^+ plays no other role in game $\mathbf{G}_2$. Thus, in game $\mathbf{G}_2$, the adversary A is essentially just carrying out an adaptive chosen ciphertext attack against SKE. □

Remark 13 We stress that it is essential for both KEM and SKE to be secure against adaptive chosen ciphertext attack in order to prove that HPKE is as well. One cannot start with a "weak" KEM and hope to "repair" it with a hybrid construction: doing this may indeed foil some specific attacks, but we know of no way to formally reason about the security of such a scheme. It is also important not to waste the chosen ciphertext security of KEM by using a "weak" SKE. Indeed, some popular methods of constructing a "digital envelope" use a SKE that may only be secure against passive attacks; even if the resulting composite ciphertext is digitally signed, this does not necessarily provide security against chosen ciphertext attack.

Remark 14 In Remarks 9 and 10, we cautioned that revealing the reason for rejecting a ciphertext may invalidate the proof of security. This is not the case for the above hybrid construction. There are two reasons a ciphertext may be rejected in this construction: either $\mathsf{KEM.Decrypt}$ rejects ψ, or $\mathsf{SKE.Decrypt}$ rejects χ. We leave it to the reader to verify that the above security proof goes through, essentially unchanged, even if the adversary is told the reason for rejecting. This is a nice feature, since in most practical implementations, it would be easy to distinguish these two rejection cases, assuming that χ was very long, and that the decryption algorithm halted immediately when $\mathsf{KEM.Decrypt}$ rejects.

8 Key Derivation Functions

In the next section, we will present and analyze a key encapsulation mechanism. The key will be derived by hashing a pair of group elements. In order not to clutter that section, we develop here the notion of such a key derivation function.

Let $\mathcal{G}$ be a computational group scheme, specifying a sequence $(\mathbf{S}_\lambda)_{\lambda\in\mathbf{Z}_{\geq 0}}$ of group distributions. A *key derivation scheme* KDF *associated with* $\mathcal{G}$ specifies two items:

- A family of *key spaces* indexed by $\lambda \in \mathbf{Z}_{\geq 0}$ and $\Gamma \in [\mathbf{S}_\lambda]$. Each such key space is a probability space, denoted $\mathsf{KDF.KeySpace}_{\lambda,\Gamma}$, on bit strings, called *derivation keys.*

  There must exist a probabilistic, polynomial-time algorithm whose output distribution on input 1^λ and Γ is equal to $\mathsf{KDF.KeySpace}_{\lambda,\Gamma}$.

- A family of *key derivation functions* indexed by $\lambda \in \mathbf{Z}_{\geq 0}$, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, and $\mathsf{dk} \in [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}]$, where each such function $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}$ maps a pair $(a, b) \in G^2$ of group elements to a key K.

  A key K is a bit string of length $\mathsf{KDF.OutLen}(\lambda)$. The parameter $\mathsf{KDF.OutLen}(\lambda)$ should be computable in deterministic polynomial time given 1^λ.

  There must exist a deterministic, polynomial-time algorithm that on input 1^λ, $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, $\mathsf{dk} \in [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}]$, and $(a, b) \in G^2$, outputs $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$.

We now define the security property that we shall require of KDF.

For all 0/1-valued, probabilistic, polynomial-time algorithms A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, define

$$
\begin{aligned}
&\mathsf{AdvDist}_{\mathsf{KDF,A}}(\lambda) := \\
&\quad\Big|\Pr[\ \tau = 1 : \Gamma \xleftarrow{R} \mathbf{S}_\lambda;\ \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\ a, b \xleftarrow{R} G; \\
&\qquad\qquad \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \mathsf{dk}, a, \mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b))\ ] - \\
&\quad\ \Pr[\ \tau = 1 : \Gamma \xleftarrow{R} \mathbf{S}_\lambda;\ \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\ a \xleftarrow{R} G;\ K \xleftarrow{R} \{0,1\}^{\mathsf{KDF.OutLen}(\lambda)}; \\
&\qquad\qquad \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \mathsf{dk}, a, K)\ ]\Big|
\end{aligned}
$$

That is, $\mathsf{AdvDist}_{\mathsf{KDF,A}}(\lambda)$ measures the advantage that A has in distinguishing two distributions: in the first it is given $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$ and in the second it is given a random key K; in both distributions it is given the derivation key dk as well as the auxiliary group element a, so in effect, both dk and a are to be regarded as public data.

We shall say that KDF is *secure* if this distinguishing advantage is negligible, i.e., *for all 0/1-valued, probabilistic, polynomial-time algorithms* A*, the function* $\mathsf{AdvDist}_{\mathsf{KDF},\mathsf{A}}(\lambda)$ *grows negligibly in* λ*.*

It is also convenient to define a quantity analogous to $\mathsf{AdvDist}_{\mathsf{KDF},\mathsf{A}}(\lambda)$, but conditioned on a fixed group description. For all 0/1-valued, probabilistic, polynomial-time algorithms A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$,

$$
\begin{aligned}
&\mathsf{AdvDist}_{\mathsf{KDF},\mathsf{A}}(\lambda \mid \Gamma) := \\
&\quad \Big| \Pr[\ \tau = 1 : \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\ a, b \xleftarrow{R} G; \\
&\qquad\qquad \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \mathsf{dk}, a, \mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b))\ ] - \\
&\quad \Pr[\ \tau = 1 : \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\ a \xleftarrow{R} G;\ K \xleftarrow{R} \{0,1\}^{\mathsf{KDF.OutLen}(\lambda)}; \\
&\qquad\qquad \tau \xleftarrow{R} \mathsf{A}(1^\lambda, \Gamma, \mathsf{dk}, a, K)\ ] \Big|
\end{aligned}
$$

8.1 Constructions

8.1.1 Unconditionally secure constructions

One can build a secure KDF for $\mathcal{G}$ without any assumptions, provided the groups defined by $\mathcal{G}$ are sufficiently large, which they certainly will be in our applications. Indeed, all we need is that KDF is *pair-wise independent*.

In our context, we shall say that a KDF is pair-wise independent if for all $\lambda \in \mathbf{Z}_{\geq 0}$, for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, for all $a, b, b' \in G$ with $b \neq b'$, the distribution

$$\{(\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b), \mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b')) : \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma}\}$$

is the uniform distribution over all pairs of bits strings of length $\mathsf{KDF.OutLen}(\lambda)$.

By the Leftover Hash Lemma [ILL89, IZ89], it follows that if KDF is pair-wise independent, then for all 0/1-valued, probabilistic, polynomial-time algorithms A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$,

$$\mathsf{AdvDist}_{\mathsf{KDF},\mathsf{A}}(\lambda \mid \Gamma) \leq 2^{-k},$$

where

$$k = \lfloor \frac{\lfloor \log_2 q \rfloor - \mathsf{KDF.OutLen}(\lambda)}{2} \rfloor.$$

We also point out that fairly efficient pair-wise independent functions can be constructed without relying on any intractability assumptions.

8.1.2 Conditionally secure constructions

In practice, to build a key derivation function, one might simply use a dedicated cryptographic hash function, like SHA-1.

In this situation, we will simply be forced to assume that such a KDF is secure. However, such an intractability assumption is not entirely unreasonable. Moreover, a dedicated cryptographic hash function has several potential advantages over a pair-wise independent hash function:

- it may not use a key, or it may use a very short key, which may lead to a significant space savings;

- it can usually be evaluated more quickly than a typical pair-wise independent hash function can;
- it can be safely used to derive output keys that are significantly longer than would be safe to derive with a typical pair-wise independent hash function;
- it may, at least heuristically, provide even more security in applications than a typical pair-wise independent hash function.

9 The New Encryption Scheme: Hybrid Version

9.1 Description of the Scheme

In this section, we present a hybrid version of our new encryption scheme. Specifically, we present a key encapsulation mechanism $\mathsf{CS3}$, out of which one can easily construct a hybrid encryption scheme, as described in §7.

The scheme makes use of a computational group scheme $\mathcal{G}$ as described in §4.1, defining a sequence $(\mathbf{S}_\lambda)_{\lambda\in\mathbf{Z}_{\geq 0}}$ of distributions of group descriptions, and providing a sampling algorithm $\hat{S}$, where the output distribution $\hat{S}(1^\lambda)$ closely approximates $\mathbf{S}_\lambda$.

The scheme also makes use of a binary group hashing scheme HF associated with $\mathcal{G}$, as described in §5.

Finally, the scheme makes use of a key derivation scheme KDF, associated with $\mathcal{G}$, as described in §8. Note that output key length $\mathsf{CS3.KeyLen}(\lambda)$ of the scheme is equal to $\mathsf{KDF.OutLen}(\lambda)$.

The scheme is described in detail in Figure 5.

9.2 Security analysis of the scheme

We shall prove that $\mathsf{CS3}$ is secure against adaptive chosen ciphertext attack if the DDH assumption holds for $\mathcal{G}$, and the TCR assumption holds for HF, and assuming that KDF is a secure key derivation scheme.

As we have done before, for all probabilistic, polynomial-time oracle query machines A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, we let $Q_\mathsf{A}(\lambda)$ be an upper bound on the number of decryption oracle queries made by A on input 1^λ. We assume that $Q_\mathsf{A}(\lambda)$ is a strict bound in the sense that it holds regardless of the probabilistic choices of A, and regardless of the responses to its oracle queries from its environment.

Theorem 6 *If the DDH assumption holds for $\mathcal{G}$ and the TCR assumption holds for HF, and assuming that KDF is a secure key derivation scheme, then $\mathsf{CS3}$ is secure against adaptive chosen ciphertext attack*

In particular, for all probabilistic, polynomial-time oracle query machines A, there exist probabilistic algorithms A_1, A_2, and A_3 whose running times are essentially the same as that of A, such that the following holds. For all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we have

$$\begin{aligned} \mathsf{AdvCCA}_{\mathsf{CS3},\mathsf{A}}(\lambda \mid \Gamma) \quad \leq \quad & \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + \mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}_2}(\lambda \mid \Gamma) + \\ & \mathsf{AdvDist}_{\mathsf{KDF},\mathsf{A}_3}(\lambda \mid \Gamma) + (Q_\mathsf{A}(\lambda) + 3)/q. \end{aligned} \tag{17}$$

To prove Theorem 6, let us fix a probabilistic, polynomial-time oracle query machine A, the value of the security parameter $\lambda \in \mathbf{Z}_{\geq 0}$, and the group description $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$.

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\begin{aligned}&\Gamma[\hat{G}, G, g, q] \xleftarrow{R} \hat{S}(1^\lambda);\ \mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\\ &w \xleftarrow{R} \mathbf{Z}_q^*;\ x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{R} \mathbf{Z}_q;\\ &\hat{g} \leftarrow g^w;\ e \leftarrow g^{x_1}\hat{g}^{x_2};\ f \leftarrow g^{y_1}\hat{g}^{y_2};\ h \leftarrow g^{z_1}\hat{g}^{z_2};\end{aligned}$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \mathsf{dk}, \hat{g}, e, f, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, \mathsf{dk}, x_1, x_2, y_1, y_2, z_1, z_2)$.

Encryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a public key

$$\mathsf{PK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, \mathsf{dk}, \hat{g}, e, f, h) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}] \times G^4,$$

compute

E1: $u \xleftarrow{R} \mathbf{Z}_q$;
E2: $a \leftarrow g^u$;
E3: $\hat{a} \leftarrow \hat{g}^u$;
E4: $b \leftarrow h^u$;
E5: $K \leftarrow \mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$;
E6: $v \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(a, \hat{a})$;
E7: $d \leftarrow e^u f^{uv}$;

and output the symmetric key K and the ciphertext $\psi = (a, \hat{a}, d)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, \mathsf{dk}, x_1, x_2, y_1, y_2, z_1, z_2) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q^6,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 3-tuple $(a, \hat{a}, d) \in \hat{G}^3$; output reject and halt if ψ is not of this form.
D2: Test if a and $\hat{a}$ belong to G; output reject and halt if this is not the case.
D3: Compute $v \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(a, \hat{a})$.
D4: Test if $d = a^{x_1+y_1v}\hat{a}^{x_2+y_2v}$; output reject and halt if this is not the case.
D5: Compute $b \leftarrow a^{z_1}\hat{a}^{z_2}$.
D6: Compute $K \leftarrow \mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$, and output the symmetric key K.

Figure 5: The key encapsulation mechanism CS3

The proof follows the same line of argument as the proof of Theorem 1, and we will at several places appeal to argument in that proof, so as to avoid unnecessary repetition.

The attack game is as described in §7.1.2. We now discuss the relevant random variables in this game.

Suppose that the public key is $(\Gamma, \mathsf{hk}, \mathsf{dk}, \hat{g}, e, f, h)$ and that the secret key is $(\Gamma, \mathsf{hk}, \mathsf{dk}, x_1, x_2, y_1, y_2, z_1, z_2)$. Let $w := \log_g \hat{g}$, and define $x, y, z \in \mathbf{Z}_q$ as follows:

$$x := x_1 + x_2 w,\ \ y := y_1 + y_2 w,\ \ z := z_1 + z_2 w.$$

As a notational convention, whenever a particular ciphertext ψ is under consideration in some context, the following values are also implicitly defined in that context:

- $a, \hat{a}, d \in G$, where $\psi = (a, \hat{a}, d)$;
- $u, \hat{u}, v, s \in \mathbf{Z}_q$, where

$$u := \log_g a,\ \ \hat{u} := \log_{\hat{g}} \hat{a},\ \ v := \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a, \hat{a}),\ \ \ s := \log_g d.$$

For the target ciphertext ψ^*, we also denote by $a^*, \hat{a}^*, d^* \in G$, and $u^*, \hat{u}^*, v^*, s^* \in \mathbf{Z}_q$ the corresponding values.

The probability space defining the attack game is then determined by the following, mutually independent, random variables:

- the coin tosses of A;
- the values $\mathsf{hk}, \mathsf{dk}, w, x_1, x_2, y_1, y_2, z_1, z_2$ generated by the key generation algorithm;
- the values $\tau \in \{0,1\}$, $K^+ \in \{0,1\}^{\mathsf{KDF.OutLen}(\lambda)}$, and $u^* \in \mathbf{Z}_q$ generated by the encryption oracle in the attack game.

Let $\mathbf{G}_0$ be the original attack game, let $\hat{\tau} \in \{0,1\}$ denote the output of A, and let T_0 be the event that $\tau = \hat{\tau}$ in $\mathbf{G}_0$, so that $\mathsf{AdvCCA}_{\mathsf{CS3},\mathsf{A}}(\lambda \mid \Gamma) = |\Pr[T_0] - 1/2|$.

As in the proof of Theorem 1, we shall define a sequence of modified games $\mathbf{G}_i$, for $i = 1, 2, \ldots$, where in game $\mathbf{G}_i$, the event T_i will be the event corresponding to event T_0, but in game $\mathbf{G}_i$. The overall structure of the proof will differ a bit from that of Theorem 1, even though many of the low level details will be very similar. Indeed, the proof of this theorem is conceptually a bit simpler (even though there are more steps) than that of Theorem 1, since the inputs to $\mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}$ in the encryption oracle are independent of any quantities computed by the adversary; we also save a term of $1/q$ in (17) because of this (compare with (3) in Theorem 1).

Game $\mathbf{G}_1$. We now modify game $\mathbf{G}_0$ to obtain a new game $\mathbf{G}_1$. These two games are identical, except that instead of using the encryption algorithm as given to compute the target ciphertext ψ^*, we use a modified encryption algorithm, in which steps **E4** and **E7** are replaced by:

E4′: $b \leftarrow a^{z_1} \hat{a}^{z_2}$;
E7′: $d \leftarrow a^{x_1 + y_1 v} \hat{a}^{x_2 + y_2 v}$.

By the same reasoning as in the proof of Theorem 1, we have

$$\Pr[T_1] = \Pr[T_0].$$

Game $\mathbf{G}_2$. We again modify the encryption oracle, replacing step **E3** by

E3′: $\hat{u} \xleftarrow{R} \mathbf{Z}_q$; $\hat{a} \leftarrow \hat{g}^{\hat{u}}$.

By the same reasoning as in the proof of Theorem 1, one sees that there exists a probabilistic algorithm A_1, whose running time is essentially the same as that of A, such that

$$|\Pr[T_2] - \Pr[T_1]| \leq \mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + 2/q.$$

Note that unlike game $\mathbf{G}_2$ in the proof of Theorem 1, we do not impose the restriction $u^* \neq \hat{u}^*$ just yet; it is technically convenient to defer this until later. This is why the term $2/q$ appears in the above bound, rather than $3/q$.

Game $\mathbf{G}_3$. This game is the same as game $\mathbf{G}_2$, except for the following modification.

We modify the decryption oracle so that it applies the following *special rejection rule*: if the adversary submits a ciphertext ψ for decryption at a point in time after the encryption oracle has been invoked, such that $(a, \hat{a}) \neq (a^*, \hat{a}^*)$ but $v = v^*$, then the decryption oracle immediately outputs reject and halts (before executing step **D4′**).

We claim that there exists a probabilistic algorithm A_2, whose running time is essentially the same as that of A, such that

$$|\Pr[T_3] - \Pr[T_2]| \leq \mathsf{AdvTCR}_{\mathsf{HF},\mathsf{A}_2}(\lambda \mid \Gamma).$$

This follows from reasoning very similar to the proof of Lemma 7 in the analysis of game $\mathbf{G}_5$ in the proof of Theorem 1. Observe that we can impose the special rejection rule already in this game, rather than deferring to a later game as in the proof of Theorem 1, because, as we mentioned above, the inputs to $\mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}$ in the encryption oracle are independent of any quantities computed by the adversary.

Game $\mathbf{G}_4$. We again modify the encryption oracle, replacing step **E3′** by

E3″: $\hat{u} \xleftarrow{R} \mathbf{Z}_q \setminus \{u\}$; $\hat{a} \leftarrow \hat{g}^{\hat{u}}$.

It is easy to verify that

$$|\Pr[T_4] - \Pr[T_3]| \leq 1/q.$$

Game $\mathbf{G}_5$. In this game, we modify the decryption oracle in game $\mathbf{G}_4$, replacing steps **D4** and **D5** with:

D4′: Test if $\hat{a} = a^w$ and $d = a^{x+yv}$; output reject and halt if this is not the case.

D5′: $b \leftarrow a^z$.

Let R_5 be the event that in game $\mathbf{G}_5$, some ciphertext ψ is submitted to the decryption oracle that is rejected in step **D4′** but that would have passed the test in step **D4**.

It is clear that

$$|\Pr[T_5] - \Pr[T_4]| \leq \Pr[R_5].$$

We also claim that

$$\Pr[R_5] \leq Q_{\mathsf{A}}(\lambda)/q.$$

This follows from reasoning analogous to that in Lemma 8 (in game $\mathbf{G}_5$ in the proof of Theorem 1).

Game $\mathbf{G}_6$. We again modify the algorithm used by the encryption oracle, replacing step **E4′** by

E4″: $r \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ b \leftarrow g^r$.

By reasoning analogous to that in the analysis of game $\mathbf{G}_4$ in the proof of Theorem 1, one can easily show that

$$\Pr[T_6] = \Pr[T_5].$$

Game $\mathbf{G}_7$. In this game, we modify the encryption oracle, replacing step **E5** of the encryption algorithm by

E5′: $K \stackrel{R}{\leftarrow} \{0,1\}^{\mathsf{KDF.OutLen}(\lambda)}$.

It is straightforward to see that there exists a probabilistic algorithm A_3, whose running time is essentially the same as that of A, such that

$$|\Pr[T_7] - \Pr[T_6]| \le \mathsf{AdvDist}_{\mathsf{KDF},\mathsf{A}_3}(\lambda \mid \Gamma).$$

Furthermore, it is clear that by construction that

$$\Pr[T_7] = 1/2.$$

That completes the proof of Theorem 6.

9.3 Two variations

One can easily modify scheme $\mathsf{CS3}$ to obtain two variants, which we call $\mathsf{CS3a}$ and $\mathsf{CS3b}$, that are analogous to the variations $\mathsf{CS1a}$ and $\mathsf{CS1b}$ of $\mathsf{CS1}$, discussed in §6.3. Only the key generation and decryption algorithms differ. The details are are presented in Figures 6 and 7.

Remark 15 Scheme $\mathsf{CS3b}$ is essentially the same scheme that was originally presented in [Sho00b]. This scheme is the most efficient scheme among all those presented in this paper. It is also attractive in that it yields a public-key encryption scheme with an unrestricted message space. Moreover, this scheme has some other attractive security properties that will be examined in §10.

Remark 16 Analogous to Remark 7, we do not have to separately test if $\hat{a}$ belongs to the subgroup G in step **D2′** of the decryption algorithm of $\mathsf{CS3b}$, and we may test if a belongs to G in some cases by testing if $a^q = 1_G$.

Remark 17 Analogous to Remark 8, in scheme $\mathsf{CS3b}$, the decryption algorithm has to compute either three or four (if we test if $a^q = 1_G$) powers of a, and special algorithmic techniques can be exploited to do this.

Remark 18 Analogous to Remarks 9 and 10, it is strongly recommended to *always* compute both exponentiations in step **D4′** of $\mathsf{CS3b}$ before rejecting the ciphertext, and to not reveal the precise reason why any ciphertext was rejected. Again, we know of no actual attack given this information, and in fact, it seems very unlikely — see §10.7 below.

The following theorem can proved using an argument almost identical to the argument that was used to prove Theorem 2. We leave it to the reader to verify this.

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\begin{aligned}&\Gamma[\hat{G}, G, g, q] \xleftarrow{R} \hat{S}(1^\lambda);\ \mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\\ &w \xleftarrow{R} \mathbf{Z}_q^*;\ x_1, x_2, y_1, y_2, z \xleftarrow{R} \mathbf{Z}_q;\\ &\hat{g} \leftarrow g^w;\ e \leftarrow g^{x_1}\hat{g}^{x_2};\ f \leftarrow g^{y_1}\hat{g}^{y_2};\ h \leftarrow g^z;\end{aligned}$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \mathsf{dk}, \hat{g}, e, f, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, \mathsf{dk}, x_1, x_2, y_1, y_2, z)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, \mathsf{dk}, x_1, x_2, y_1, y_2, z) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q^5,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 3-tuple $(a, \hat{a}, d) \in \hat{G}^3$; output reject and halt if ψ is not of this form.
D2: Test if a and $\hat{a}$ belong to G; output reject and halt if this is not the case.
D3: Compute $v \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(a, \hat{a})$.
D4: Test if $d = a^{x_1+y_1v}\hat{a}^{x_2+y_2v}$; output reject and halt if this is not the case.
D5′: Compute $b \leftarrow a^z$.
D6: Compute $K \leftarrow \mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$, and output the symmetric key K.

Figure 6: Key generation and decryption algorithms for CS3a

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\begin{aligned}&\Gamma[\hat{G}, G, g, q] \xleftarrow{R} \hat{S}(1^\lambda);\ \mathsf{hk} \xleftarrow{R} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\\ &w \xleftarrow{R} \mathbf{Z}_q^*;\ x, y, z \xleftarrow{R} \mathbf{Z}_q;\\ &\hat{g} \leftarrow g^w;\ e \leftarrow g^x;\ f \leftarrow g^y;\ h \leftarrow g^z;\end{aligned}$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, \mathsf{dk}, \hat{g}, e, f, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, \mathsf{dk}, x, y, z)$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{hk}, \mathsf{dk}, x, y, z) \in [\mathbf{S}_\lambda] \times [\mathsf{HF.KeySpace}_{\lambda,\Gamma}] \times [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q^3,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a 3-tuple $(a, \hat{a}, d) \in \hat{G}^3$; output reject and halt if ψ is not of this form.
D2′: Test if a belongs to G; output reject and halt if this is not the case.
D3: Compute $v \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(a, \hat{a})$.
D4′: Test if $\hat{a} = a^w$ and $d = a^{x+yv}$; output reject and halt if this is not the case.
D5′: Compute $b \leftarrow a^z$.
D6: Compute $K \leftarrow \mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$, and output the symmetric key K.

Figure 7: Key generation and decryption algorithms for CS3b

Theorem 7 *If the DDH assumption holds for $\mathcal{G}$ and the TCR assumption holds for* HF*, and assuming that* KDF *is a secure key derivation scheme, then* $\mathsf{CS3a}$ *and* $\mathsf{CS3b}$ *are secure against adaptive chosen ciphertext attack.*

In particular, for all probabilistic, polynomial-time oracle query machines A*, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we have*

$$|\mathsf{AdvCCA}_{\mathsf{CS3a},\mathsf{A}}(\lambda \mid \Gamma) - \mathsf{AdvCCA}_{\mathsf{CS3},\mathsf{A}}(\lambda \mid \Gamma)| \leq Q_{\mathsf{A}}(\lambda)/q$$

and

$$|\mathsf{AdvCCA}_{\mathsf{CS3b},\mathsf{A}}(\lambda \mid \Gamma) - \mathsf{AdvCCA}_{\mathsf{CS3},\mathsf{A}}(\lambda \mid \Gamma)| \leq Q_{\mathsf{A}}(\lambda)/q$$

10 Further Security Considerations of Scheme CS3b

The key encapsulation mechanism CS3b, which was described and analyzed in §9.3, has some other interesting security properties, which we discuss in this section.

The main results we present here are the following. First, we show that CS3b is no less secure than a more traditional key encapsulation mechanism that is a hashed variant of ElGamal encryption, which we call HEG. Second, we also show that CS3b is secure in the random oracle model (viewing KDF as a random oracle) if the CDH and TCR assumptions hold. Along the way, we also give a security analysis of HEG in the random oracle model, based on a rather non-standard intractability assumption.

10.1 Hashed ElGamal key encapsulation

We begin by presenting a fairly traditional version of ElGamal key encapsulation, which we call HEG.

The scheme makes use of a computational group scheme $\mathcal{G}$ as described in §4.1, defining a sequence $(\mathbf{S}_\lambda)_{\lambda \in \mathbf{Z}_{\geq 0}}$ of distributions of group descriptions, and providing a sampling algorithm $\hat{S}$, where the output distribution $\hat{S}(1^\lambda)$ closely approximates $\mathbf{S}_\lambda$.

Also, the scheme makes use of a key derivation scheme KDF, associated with $\mathcal{G}$, as described in §8. Note that output key length $\mathsf{EG.KeyLen}(\lambda)$ of the scheme is equal to $\mathsf{KDF.OutLen}(\lambda)$.

The scheme is described in detail in Figure 8.

10.2 The random oracle model

We will analyze the security of both schemes HEG and CS3b in the random oracle model. In this approach, a cryptographic hash function — in this case KDF — is modeled for the purposes of analysis as a "black box" containing a random function to which the adversary and the algorithms implementing the cryptosystem have "oracle access." This approach has been used implicitly and informally for some time; however, it was formalized by Bellare and Rogaway [BR93], and has subsequently been used quite a bit in the cryptographic research community.

More precisely, we shall analyze the security the scheme HEG and later CS3b in an idealized model of computation where for all $\lambda \in \mathbf{Z}_{\geq 0}$, all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, all $\mathsf{dk} \in [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}]$, and all $a, b \in G$, we treat the values $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$ as mutually independent, random bit strings of length $\mathsf{KDF.OutLen}(\lambda)$; moreover, the only way to obtain the value of $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$ is to explicitly query an oracle with input $(\lambda, \Gamma, \mathsf{dk}, a, b)$. Actually, to be complete, we allow Γ, dk, a, and b to range

Key Generation: On input 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, compute

$$\Gamma[\hat{G}, G, g, q] \xleftarrow{R} \hat{S}(1^\lambda);\ \mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma};\ z \xleftarrow{R} \mathbf{Z}_q;\ h \leftarrow g^z;$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{dk}, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{dk}, z)$.

Encryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a public key

$$\mathsf{PK} = (\Gamma[\hat{G}, G, g, q], \mathsf{dk}, h) \in [\mathbf{S}_\lambda] \times [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}] \times G,$$

compute

E1: $u \xleftarrow{R} \mathbf{Z}_q$;
E2: $a \leftarrow g^u$;
E3: $b \leftarrow h^u$;
E4: $K \leftarrow \mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$;

and output the symmetric key K and the ciphertext $\psi = a$.

Decryption: Given 1^λ for $\lambda \in \mathbf{Z}_{\geq 0}$, a secret key

$$\mathsf{SK} = (\Gamma[\hat{G}, G, g, q], \mathsf{dk}, z) \in [\mathbf{S}_\lambda] \times [\mathsf{KDF.KeySpace}_{\lambda,\Gamma}] \times \mathbf{Z}_q,$$

along with a ciphertext ψ, do the following.

D1: Parse ψ as a group element $a \in \hat{G}$; output reject and halt if ψ is not of this form.
D2: Test if a belongs to G; output reject and halt if this is not the case.
D3: Compute $b \leftarrow a^z$.
D4: Compute $K \leftarrow \mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$, and output the symmetric key K.

Figure 8: The key encapsulation mechanism HEG

over arbitrary bit strings, regardless of whether these are valid encodings of appropriate objects. Since in any of our applications, only a finite number of the values $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a,b)$ will be relevant, experiments based on these values can be modeled using finite probability spaces.

When considering an adversary A that is carrying out an adaptive chosen ciphertext attack in the random oracle model, in addition to the usual types of oracle queries that A makes, the adversary A is also allowed to query the random oracle representing KDF. We shall denote by $Q'_{\mathsf{A}}(\lambda)$ a strict upper bound on the number of random oracle queries that A makes for a given value of the security parameter λ; as usual, this bound should hold regardless of the environment in which A actually runs.

10.3 CS3b is at least as secure as HEG

We now show that the scheme $\mathsf{CS3b}$ is at least as secure as HEG.

Theorem 8 *If scheme* HEG *is secure against adaptive chosen ciphertext attack, then so is* $\mathsf{CS3b}$*; moreover, this implication holds in either the standard or random oracle models.*

In particular, for all probabilistic, polynomial-time oracle query machines A*, there exists another oracle query machine* A_1*, whose running time is essentially the same as that of* A*, such that for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, and all* $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$*, we have*

$$\mathsf{AdvCCA}_{\mathsf{CS3b},\mathsf{A}}(\lambda \mid \Gamma) \leq \mathsf{AdvCCA}_{\mathsf{HEG},\mathsf{A}_1}(\lambda \mid \Gamma);$$

moreover, $Q_{\mathsf{A}_1}(\lambda) \leq Q_{\mathsf{A}}(\lambda)$ *and (in the random oracle model)* $Q'_{\mathsf{A}_1}(\lambda) \leq Q'_{\mathsf{A}}(\lambda)$*.*

Proof. Fix A, λ, and $\Gamma[\hat{G}, G, g, q]$ as above. We construct an adversary A_1 that attacks HEG. The adversary A_1 makes use of A by providing an environment for A, as follows.

First, suppose that A_1 is given a public key (Γ, dk, h) for scheme HEG, where Γ is fixed as above. Adversary A_1 then "dresses up" the HEG public key to look like a $\mathsf{CS3b}$ public key; namely, A_1 computes

$$\mathsf{hk} \stackrel{R}{\leftarrow} \mathsf{HF.KeySpace}_{\lambda,\Gamma};\ w \stackrel{R}{\leftarrow} \mathbf{Z}_q^*;\ x, y \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ \hat{g} \leftarrow g^w;\ e \leftarrow g^x;\ f \leftarrow g^y;$$

and presents A with the $\mathsf{CS3b}$ public key

$$(\Gamma, \mathsf{hk}, \mathsf{dk}, \hat{g}, e, f, h).$$

Second, whenever A submits a $\mathsf{CS3b}$ ciphertext $(a, \hat{a}, d) \in \hat{G}^3$ to the decryption oracle, adversary A_1 first performs the validity tests of the decryption algorithm of $\mathsf{CS3b}$, making use of the values hk, w, x, y generated above; if these tests pass, then A_1 invokes the decryption oracle of HEG with input a.

Third, when A invokes the encryption oracle of $\mathsf{CS3b}$, adversary A_1 does the following. It invokes the encryption oracle of HEG, obtaining a ciphertext $a^* \in G$ and a key $K^\dagger$. It then "dresses up" a^* to look like a $\mathsf{CS3b}$ ciphertext; namely, it computes

$$\hat{a}^* \leftarrow (a^*)^w;\ v^* \leftarrow \mathsf{HF}^{\lambda,\Gamma}_{\mathsf{hk}}(a^*, \hat{a}^*);\ d^* \leftarrow (a^*)^{x+yv^*};$$

and presents A with the $\mathsf{CS3b}$ ciphertext $(a^*, \hat{a}^*, d^*)$ along with the key $K^\dagger$.

Fourth, when A terminates and outputs a value, A_1 also terminates and outputs the same value.

That completes the description of the adversary A_1.

One has to check that A_1 carries out a legal adaptive chosen ciphertext attack, in the sense that it should not attempt to submit the target ciphertext itself to the decryption oracle, subsequent to the invocation of the encryption oracle. Consider a ciphertext a submitted by A_1 to the decryption oracle. This was derived from a valid CS3b ciphertext $(a, \hat{a}, d)$ submitted by A to the decryption oracle. By the construction, it is easy to see that if $a = a^*$, then in fact, $(a, \hat{a}, d) = (a^*, \hat{a}^*, d^*)$, which cannot happen if A itself carries out a legal attack.

Since the simulation by A_1 above is perfect, it is clear that whatever advantage A has in guessing the hidden bit, adversary A_1 has precisely the same advantage. It is also clear by construction that $Q_{\mathsf{A}_1}(\lambda) \leq Q_{\mathsf{A}}(\lambda)$, and in the random oracle model that $Q'_{\mathsf{A}_1}(\lambda) \leq Q'_{\mathsf{A}}(\lambda)$. $\square$

10.4 The security of HEG in the random oracle model

As for the security of HEG, even in the random oracle model, we do not know how to prove a very strong result. We content ourselves with a proof that the scheme HEG is secure against adaptive chosen ciphertext attack in the random oracle model, provided the CDH assumption holds *relative to an oracle for the DDH problem.*

More precisely, for all probabilistic, polynomial-time oracle query machines A, and for all $\lambda \in \mathbf{Z}_{\geq 0}$, we define

$$\mathsf{AdvCDH}^*_{\mathcal{G},\mathsf{A}}(\lambda) := \Pr[\ c = g^{xy} : \Gamma[\hat{G}, G, g, q] \stackrel{R}{\leftarrow} \mathbf{S}_\lambda;\ x \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ y \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ c \stackrel{R}{\leftarrow} \mathsf{A}^{\mathsf{DHP}_{\lambda,\Gamma}}(1^\lambda, \Gamma, g^x, g^y)\ ],$$

where the notation $\mathsf{A}^{\mathsf{DHP}_{\lambda,\Gamma}}(\cdots)$ signifies that A runs with access to an oracle for the Diffie-Hellman predicate $\mathsf{DHP}_{\lambda,\Gamma}$ defined in §4.3.3.

We say that the CDH assumption for $\mathcal{G}$ holds relative to an oracle for the DDH problem if:

> *for all probabilistic, polynomial-time oracle query machines* A*, the function* $\mathsf{AdvCDH}^*_{\mathcal{G},\mathsf{A}}(\lambda)$ *is negligible in* λ.

For all probabilistic, polynomial-time oracle query machines A, for all $\lambda \in \mathbf{Z}_{\geq 0}$, and for all $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$, we also define

$$\mathsf{AdvCDH}^*_{\mathcal{G},\mathsf{A}}(\lambda \mid \Gamma) := \Pr[\ c = g^{xy} : x \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ y \stackrel{R}{\leftarrow} \mathbf{Z}_q;\ c \stackrel{R}{\leftarrow} \mathsf{A}^{\mathsf{DHP}_{\lambda,\Gamma}}(1^\lambda, \Gamma, g^x, g^y)\ ].$$

On the one hand, it has been proven (see [MW98]) that in a "generic" model of computation, the CDH problem remains hard even in the presence of a DDH oracle, thus giving some justification for this assumption; on the other hand, if one works with the special elliptic curves discussed in the paper [JN01] already mentioned in §4.4, then the DDH oracle actually has an efficient implementation, even though the CDH problem still appears hard.

Theorem 9 *The scheme* HEG *is secure in the random oracle model if the CDH assumption for* $\mathcal{G}$ *holds relative to an oracle for the DDH problem.*

In particular, for all probabilistic, polynomial-time oracle query machines A*, there exists an oracle query machine* A_1*, whose running time is essentially the same as that of* A*, such that for all* $\lambda \in \mathbf{Z}_{\geq 0}$*, and for all* $\Gamma[\hat{G}, G, g, q] \in [\mathbf{S}_\lambda]$*, we have*

$$\mathsf{AdvCCA}_{\mathsf{HEG},\mathsf{A}}(\lambda \mid \Gamma) \leq \mathsf{AdvCDH}^*_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) + Q_{\mathsf{A}}(\lambda)/q;$$

moreover, the number or DDH-oracle queries made by A_1 *is bounded by* $Q'_{\mathsf{A}}(\lambda)$.

To prove Theorem 9, let us fix A, λ, and $\Gamma[\hat{G}, G, g, q]$. The attack game is as described in §7.1.2.

We begin by describing the relevant random variables in the attack game. The public key is (Γ, dk, h) and the secret key is (Γ, dk, z).

For a given ciphertext ψ, we let $a \in G$ denote the corresponding group element, we let $b := a^z$, $u := \log_g a$, and $K := \mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a, b)$. Note also that $b = a^u$. For the target ciphertext ψ^*, we let a^*, b^*, u^*, and K^* denote the corresponding values.

The encryption oracle also generates values $\tau \in \{0, 1\}$ and $K^+ \in \{0, 1\}^{\mathsf{KDF.OutLen}(\lambda)}$.

Let $\mathbf{G}_0$ be the original attack game, let $\hat{\tau}$ denote the output of A, and let T_0 be the event that $\tau = \hat{\tau}$, so that $\mathsf{AdvCCA}_{\mathsf{HEG},\mathsf{A}}(\lambda \mid \Gamma) = |\Pr[T_0] - 1/2|$.

As usual, we define a sequence of games $\mathbf{G}_1$, $\mathbf{G}_2$, etc., and in game $\mathbf{G}_i$ for $i \geq 1$ we define T_i to be the event in game $\mathbf{G}_i$ corresponding to event T_0 in game $\mathbf{G}_0$.

Game $\mathbf{G}_1$. We modify game $\mathbf{G}_0$ as follows. First, we run the encryption oracle at the beginning of the attack game, but we give the results of this to the adversary only when it actually invokes the encryption oracle. This is a purely conceptual change, since the adversary provides no input to the encryption oracle. Second, if the adversary ever submits a ciphertext $\psi = \psi^*$ to the decryption oracle *before* the encryption algorithm is invoked, we abort the game immediately, before responding to this decryption oracle invocation (the environment, say, goes silent at this point).

Let F_1 be the event that game $\mathbf{G}_1$ is aborted as above. It is clear that $\Pr[F_1] \leq Q_{\mathsf{A}}(\lambda)/q$. It is also clear that games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically until event F_1 occurs, and so by Lemma 4, we have $|\Pr[T_1] - \Pr[T_0]| \leq \Pr[F_1]$.

Game $\mathbf{G}_2$. We next modify game $\mathbf{G}_1$ as follows. If the adversary ever queries the random oracle to obtain the value of $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a^*, b^*)$, we immediately abort the game, before responding to this random oracle invocation.

It is easy to see that $\Pr[T_2] = 1/2$. This follows directly from the fact that in game $\mathbf{G}_2$, the value of $\mathsf{KDF}^{\lambda,\Gamma}_{\mathsf{dk}}(a^*, b^*)$ is obtained from the random oracle only by the encryption oracle: the adversary never queries the random oracle directly at this point, nor does the decryption oracle.

Let F_2 be the event that game $\mathbf{G}_2$ is aborted as above. It is clear that $|\Pr[T_2] - \Pr[T_1]| \leq \Pr[F_2]$, so it suffices to bound $\Pr[F_2]$.

We claim that $\Pr[F_2] = \mathsf{AdvCDH}^*_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma)$ for an oracle query machine A_1 whose running time and number of oracle queries are bounded as in the statement of the theorem.

We now describe A_1. It takes as input 1^λ, $\Gamma[\hat{G}, G, g, q]$, along with group elements $a^*, h \in G$, and attempts to compute $b^* \in G$ such that $\mathsf{DHP}_{\lambda,\Gamma}(h, a^*, b^*) = 1$. The machine A_1 has access to an oracle for the function $\mathsf{DHP}_{\lambda,\Gamma}$.

Machine A_1 simulates the environment of game $\mathbf{G}_2$ for A as follows. It first computes $\mathsf{dk} \xleftarrow{R} \mathsf{KDF.KeySpace}_{\lambda,\Gamma}$ and gives A the public key (Γ, dk, h). For the target ciphertext, it of course sets $\psi^* := a^*$. For the other output $K^\dagger$ of the encryption oracle, A_1 simply generates this as a random bit string of length $\mathsf{KDF.OutLen}(\lambda)$.

Machine A_1 also needs to simulate the responses to the random oracle and decryption oracle queries. For the random oracle queries, the only values that are relevant are those corresponding to the given values of λ, Γ, and dk.

For the irrelevant random oracle queries, A_1 simply maintains a set of input/output pairs, generating outputs at random as necessary.

Machine A_1 processes relevant random oracle queries using the following data structures:

- a set $\mathcal{V}_1$ of triples (a, b, K), with $a, b \in G$ and $K \in \{0,1\}^{\mathsf{KDF.OutLen}(\lambda)}$, initially empty; this will contain those triples (a, b, K) for which A_1 has assigned the value K to $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$;
- a set $\mathcal{V}_2$ of pairs (a, b), with $a, b \in G$, initially empty; this will contain precisely those pairs (a, b) such that $(a, b, K) \in \mathcal{V}_1$ for some K, and $\mathsf{DHP}_{\lambda,\Gamma}(h, a, b) = 1$;
- a set $\mathcal{V}_3$ of pairs (a, K), with $a \in G$ and $K \in \{0,1\}^{\mathsf{KDF.OutLen}(\lambda)}$, initially empty; this will contain pairs (a, K) for which A_1 has assigned the value K to $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$ for $b \in G$ with $\mathsf{DHP}_{\lambda,\Gamma}(h, a, b) = 1$, even though A_1 does not actually know the value of b.

Given a request for the value $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$, machine A_1 does the following:

- It tests if $(a, b, K) \in \mathcal{V}_1$ for some K. If so (which means that A has queried the value $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$ before), it returns K as the value of $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$; otherwise, it continues.
- It invokes its own DDH-oracle to determine if $\mathsf{DHP}_{\lambda,\Gamma}(h, a, b) = 1$.
- If $\mathsf{DHP}_{\lambda,\Gamma}(h, a, b) = 1$, then:
  - If $a = a^*$, it halts and outputs the solution $b^* := b$ to the given problem instance (this corresponds to the early-abort rule introduced in game $\mathbf{G}_2$); otherwise, it continues.
  - It adds the pair (a, b) to the set $\mathcal{V}_2$.
  - If $(a, K) \in \mathcal{V}_3$ for some K, then it adds the triple (a, b, K) to $\mathcal{V}_1$, and returns K as the value of $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$; otherwise, it continues.
- It generates K as a random bit string of length $\mathsf{KDF.OutLen}(\lambda)$, adds the triple (a, b, K) to $\mathcal{V}_1$, and returns K as the value of $\mathsf{KDF}_{\mathsf{dk}}^{\lambda,\Gamma}(a, b)$.

Machine A_1 processes decryption oracle queries as follows. Suppose it is given a ciphertext ψ, with $a \in G$ the corresponding group element. Then it does the following:

- If $\psi = \psi^*$ (which can only happen if the encryption oracle has not yet been invoked), then it simply halts (this corresponds to the early-abort rule introduced in game $\mathbf{G}_1$); otherwise, continues.
- It tests if $(a, b) \in \mathcal{V}_2$ for some $b \in G$.
- If this is so, then it finds the (unique) triple in $\mathcal{V}_1$ of the form (a, b, K) for some K, and returns this value of K as the result of the decryption oracle invocation; otherwise, it continues.
- It tests if $(a, K) \in \mathcal{V}_3$ for some K.
- If this is so, then it returns this value of K as the result of the decryption oracle; otherwise, it generates a random bit string K of length $\mathsf{KDF.OutLen}(\lambda)$, adds the pair (a, K) to $\mathcal{V}_3$, and returns this value of K as the result of the decryption oracle invocation.

It is straightforward to verify by inspection that A_1 as above does the job.

That completes the proof of Theorem 9.

10.5 The security of CS3b in the random oracle model

We can now prove the following security theorem for CS3b in the random oracle model.

Theorem 10 *The scheme* CS3b *is secure in the random oracle model if the CDH assumption holds for* $\mathcal{G}$*, and the TCR assumption holds for* HF.

Proof. To prove this, let us assume by way of contradiction that the CDH assumption holds for $\mathcal{G}$ and the TCR assumption holds for HF, but CS3b is not secure in the random oracle model.

Now, the CDH assumption implies that for any polynomials P_1 and P_2 (with integer coefficients, taking positive values on $\mathbf{Z}_{\geq 0}$), there exists a $\lambda_0 \in \mathbf{Z}_{\geq 0}$, such that for all $\lambda \geq \lambda_0$,

$$\Pr[q \leq P_1(\lambda) : \Gamma[\hat{G}, G, g, q] \stackrel{R}{\leftarrow} \mathbf{S}_\lambda] \leq 1/P_2(\lambda),$$

since otherwise, a trivial, brute-force algorithm would have a CDH advantage that was not negligible. This implies in particular that when we model KDF as a random oracle, it acts as a secure key derivation scheme. From this it follows from Theorems 6 and 7 that CS3b is secure in the random oracle model if the DDH assumption holds; actually, since these two theorems do not deal with the random oracle model, one must make a cursory inspection of the proofs of these theorems to draw this conclusion, but this is very straightforward.

Let A be a polynomial-time adversary that breaks the security of CS3b in the random oracle model. This means that there exist polynomials P_1, P_2 (with integer coefficients, taking positive values on $\mathbf{Z}_{\geq 0}$), an infinite set $\Lambda \subset \mathbf{Z}_{\geq 0}$, and sets $\mathcal{Z}_\lambda \subset [\mathbf{S}_\lambda]$ for each $\lambda \in \Lambda$, such that

- for all $\lambda \in \Lambda$ and $\Gamma \in \mathcal{Z}_\lambda$, $\mathsf{AdvCCA}_{\mathsf{CS3b},\mathsf{A}}(\lambda \mid \Gamma) \geq 1/P_1(\lambda)$,
- for all $\lambda \in \Lambda$, $\Pr_{\mathbf{S}_\lambda}[\mathcal{Z}_\lambda] \geq 1/P_2(\lambda)$.

Theorems 6 and 7 (adapted to the random oracle model), together with our TCR assumption, imply that there exists a polynomial-time algorithm A_1, such that for all sufficiently large $\lambda \in \Lambda$, and for all but a negligible fraction of Γ in $\mathcal{Z}_\lambda$, we have

$$\mathsf{AdvDDH}_{\mathcal{G},\mathsf{A}_1}(\lambda \mid \Gamma) \geq 1/(2P_1(\lambda)).$$

We now apply Lemma 3 using the above algorithm A_1, and choosing the value of κ in that lemma so that $2^{-\kappa} \cdot Q'_{\mathsf{A}}(\lambda) \leq 1/2$, yielding a polynomial-time algorithm A_2, such that for all sufficiently large $\lambda \in \Lambda$, and for all but a negligible fraction of $\Gamma \in \mathcal{Z}_\lambda$, and for all $\rho \in \mathcal{T}_{\lambda,\Gamma}$,

$$\Pr[\mathsf{A}_2(1^\lambda, \Gamma, \rho) \neq \mathsf{DHP}_{\lambda,\Gamma}(\rho)] \leq 1/(2Q'_{\mathsf{A}}(\lambda)).$$

Applying Theorem 8 with the adversary A yields a polynomial-time adversary A_3 such that for all $\lambda \in \Lambda$ and $\Gamma \in \mathcal{Z}_\lambda$, $\mathsf{AdvCCA}_{\mathsf{HEG},\mathsf{A}_3}(\lambda \mid \Gamma) \geq 1/P_1(\lambda)$. Applying Theorem 9 with the adversary A_3 yields a polynomial-time oracle machine A_4 such that

$$\mathsf{AdvCDH}^*_{\mathcal{G},\mathsf{A}_4}(\lambda \mid \Gamma) \geq 1/(2P_1(\lambda))$$

for all sufficiently large $\lambda \in \Lambda$, and for all but a negligible fraction of $\Gamma \in \mathcal{Z}_\lambda$. Since for a given value of λ, algorithm A_4 makes no more than $Q'_{\mathsf{A}}(\lambda)$ DDH-oracle queries, if we replace the DDH-oracle used by A_4 with algorithm A_2 above, we obtain a polynomial-time algorithm A_5 such that for all sufficiently large $\lambda \in \Lambda$, and for all but a negligible fraction of Γ in $\mathcal{Z}_\lambda$, we have $\mathsf{AdvCDH}_{\mathcal{G},\mathsf{A}_5}(\lambda \mid \Gamma) \geq 1/(4P_1(\lambda))$. But this contradicts the CDH assumption. □

10.6 Random oracles and pair-wise independent key derivation functions: getting the best of both

If we want to prove the security of CS3b in the standard model without making any intractability assumptions about KDF, then we may choose KDF to be pair-wise independent. On the one hand, standard constructions for pair-wise independent hash functions typically exhibit a lot of algebraic structure, and it is not very reasonable to assume that such a KDF can be safely modeled as a random oracle. On the other hand, typical dedicated cryptographic hash functions, like SHA-1, may be modeled as random oracles, but they are certainly not pair-wise independent.

We shall sketch here how to get the best of both worlds, i.e., how to implement the KDF so that we get a proof of security of CS3b in the standard model just under the DDH and TCR assumptions, and in the random oracle model under the CDH and TCR assumptions.

The idea is this: compute KDF as the XOR of a pair-wise independent hash KDF1 and a cryptographic hash KDF2.

It is clear that if KDF1 is pair-wise independent, then so is KDF, and so the security of CS3b in the standard model under the DDH and TCR assumptions now follows directly from Theorem 7.

Now suppose we model the cryptographic hash KDF2 as a random oracle. It is easy to see that for any adversary A attacking CS3b given oracle access to KDF2, there is an adversary A_1, whose running time is roughly the same as that of A, that attacks CS3b given oracle access to KDF: the adversary A_1 just does whatever A does, except that whenever A queries the oracle for KDF2, adversary A_1 queries its oracle for KDF and computes the value of KDF2 as the XOR of the value of KDF and the value of KDF1. Note, however, that the output distribution of the oracle KDF is the same as that of a random oracle, and so the security of CS3b in the random oracle model under the CDH and TCR assumptions now follows directly from Theorem 10.

We do not necessarily advocate this approach to building a KDF in practical implementations: simply assuming that a KDF implemented directly using a dedicated cryptographic hash is secure is quite reasonable, and the resulting KDF is much simpler and more efficient than any approach that makes use of a pair-wise independent hash function.

10.7 Further discussion

The scheme HEG is intended to represent a fairly traditional version of ElGamal key encapsulation. The only thing slightly non-traditional about it is the fact that the symmetric key K is derived by hashing both a (the ephemeral Diffie-Hellman public key) and b (the shared Diffie-Hellman key), rather than just b alone.

Hashing both the ephemeral and shared keys together has some quantitative security advantages. Notice that in Theorem 9, the implied CDH algorithm makes no more than $Q'_{\mathsf{A}}(\lambda)$ queries to the DDH-oracle. If we were to hash only the shared Diffie-Hellman key, we could still prove the security of HEG, but the reduction would be less efficient; in particular, the implied CDH algorithm might require up to $Q'_{\mathsf{A}}(\lambda) \cdot Q_{\mathsf{A}}(\lambda)$ queries to the DDH-oracle. A similar quantitative security advantage arises in the multi-user/multi-message model (see [BBM00]). In this model, we can exploit the well-know random self-reducibility of the CDH problem to get a more efficient reduction if we hash both keys instead of just one. Of course, these improved security reductions for HEG carry over to the security reduction for CS3b in the random oracle model.

The DHAES encryption scheme [ABR99], which is a hybrid ElGamal encryption scheme that has been proposed for standardization, also hashes both the ephemeral and shared Diffie-Hellman keys to derive a symmetric key. Indeed, the DHAES scheme can be constructed from the key encapsulation mechanism HEG using the hybrid constructions presented in §7, and it is straightforward to verify that analogues of Theorems 8 and 9 hold for the DHAES scheme as well. The DHAES scheme needs to hash both group elements because it allows the possibility of a group G whose order is a composite number. In a revised version of DHAES, called DHIES [ABR01], the group G is required to have prime order, and only the shared Diffie-Hellman key is hashed. However, as we have seen, there are still some security benefits to be gained from hashing both group elements, even if the group is of prime order, as we are assuming in this paper.

We should also point out that the because any attack on scheme CS3b can be immediately translated into an attack on HEG (see Theorem 8), it seems very unlikely that any of the side channels discussed in Remark 18 could be used to break CS3b, without breaking HEG. Thus, although the availability of such side channel information would invalidate the proof of security of CS3b under the DDH, it seems very unlikely that it could be exploited to break it.

Theorem 10 originally appeared in the paper [Sho00b]. The proof in that paper basically rolled all of the arguments used in the proofs of Theorems 8, 9, 10, along with the arguments in §10.6, into a single proof, which we have unraveled to some extent here. Our presentation here was somewhat influenced by the paper [OP01], which formally introduces the notion of the CDH assumption relative to an oracle for the DDH problem.

The security reduction in Theorem 10 is quite inefficient: we have to perform many simulations using the given adversary A just to solve one instance of the DDH problem, and then in a different simulation involving A, we have to solve many instances of the DDH problem in order to solve one instance of the CDH problem. Of course, if the DDH problem for a given group scheme turns out not to be a hard problem, then it may very well be the case that there is a much more efficient DDH algorithm than the one built using our security reduction involving A. In this case, the reduction in Theorem 10 becomes quite reasonable.

11 Summary

We conclude with a brief summary of the schemes presented in this paper:

- the public-key encryption scheme CS1, which is secure under the DDH and TCR assumptions, along with minor, slightly simpler and more efficient, variants CS1a and CS1b;
- the public-key encryption scheme CS2, which is a "hash free" variant of CS1, and which is secure under the DDH assumption;
- the key encapsulation mechanism CS3, along with variants CS3a and CS3b, which can be combined with an appropriate symmetric cipher to build a hybrid cipher; the hybrid cipher will be secure under the DDH and TCR assumptions, together with an appropriate "smoothing" assumption for the key derivation function, and an appropriate security assumption for the symmetric cipher.

Among these schemes, the one most likely to be used in practice is the hybrid construction based on CS3b. Over the other schemes, it has the following advantages:

- it is more efficient;
- it can be used to encrypt messages that are bit strings of arbitrary length;

- it is no less secure than traditional "hashed" ElGamal;
- it can be proven secure in the random oracle model under weaker CDH assumption.

A version of this scheme is currently included in a draft ISO standard for public key encryption.

Acknowledgments

Some of this work was done while the first author was at the Institute for Theoretical Computer Science, ETH, Zurich, and while the second author was at the IBM Zurich Research Lab, as well as visiting the Computer Science Department at Stanford University. Thanks to Ilia Mironov for comments on an early draft of this paper.

Thanks also to Moni Naor for his comments on an early draft of the *Crypto '98* paper on which this paper is based, and in particular, for his suggestion of using a universal one-way hash function instead of a collision resistant hash function in the design of scheme CS1, and for his suggestion of a hash-free variant, upon which scheme CS2 is loosely based.

References

[ABR99] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: an encryption scheme based on the Diffie-Hellman problem. Cryptology ePrint Archive, Report 1999/007, 1999. `http://eprint.iacr.org`.

[ABR01] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Topics in Cryptology – CT-RSA 2001*, pages 143–158, 2001. Springer LNCS 2045.

[ASW00] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000. Extended abstract in *Advances in Cryptology–Eurocrypt '98*.

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: security proofs and improvements. In *Advances in Cryptology–Eurocrypt 2000*, 2000.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology–Crypto '98*, pages 26–45, 1998.

[BGMW92] E. F. Brickell, D. M. Gordon, K. S. McCurley, and D. B. Wilson. Fast exponentiation with precomputation. In *Advances in Cryptology–Eurocrypt '92*, pages 200–207, 1992.

[BH62] P. Bateman and R. Horn. A heuristic asymptotic formula concerning the distribution of prime numbers. *Math. Comp.*, 16:363–367, 1962.

[BH65] P. Bateman and R. Horn. Primes represented by irreducible polynomials in one variable. *Proc. Sympos. Pure Math.*, 8:119–135, 1965.

[Ble98] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology–Crypto '98*, pages 1–12, 1998.

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

[Bon01] D. Boneh. Simplified OAEP for the RSA and Rabin functions. In *Advances in Cryptology–Crypto 2001*, 2001.

[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—Eurocrypt '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.

[BS96] E. Bach and J. Shallit. *Algorithmic Number Theory*, volume 1. MIT Press, 1996.

[BSS99] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.

[Can00] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000. `http://eprint.iacr.org`.

[CG99] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Advances in Cryptology–Eurocrypt '99*, pages 90–106, 1999.

[CGH98] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, 1998.

[CLRS02] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2002.

[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.

[CS00] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Transactions on Information and Systems Security*, 3(3):161–185, 2000.

[CS02] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public key encryption. In *Advances in Cryptology–Eurocrypt 2002*, pages 45–64, 2002.

[Dam91] I. Damgård. Towards practical public key cryptosystems secure against chosen ciphertext attacks. In *Advances in Cryptology–Crypto '91*, pages 445–456, 1991.

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

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

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

[DN96] C. Dwork and M. Naor. Method for message authentication from non-malleable cryptosystems, 1996. U. S. Patent No. 05539826.

[ElG85] T. ElGamal. A public key cryptosystem and signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[FOPS01] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In *Advances in Cryptology–Crypto 2001*, pages 260–274, 2001.

[FY95] Y. Frankel and M. Yung. Cryptanalysis of immunized LL public key systems. In *Advances in Cryptology–Crypto '95*, pages 287–296, 1995.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudo-random generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from any one-way function. In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, 1989.

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

[JN01] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org`.

[LL93] C. H. Lim and P. J. Lee. Another method for attaining security against adaptively chosen ciphertext attacks. In *Advances in Cryptology–Crypto '93*, pages 420–434, 1993.

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

[Man01] J. Manger. A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS # 1 v2.0. In *Advances in Cryptology–Crypto 2001*, pages 230–238, 2001.

[MvOV97] A. Menesez, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[MW98] U. Maurer and S. Wolf. Diffie-Hellman, decision Diffie-Hellman, and discrete logarithms. In *Proc. ISIT '98*, page 327. IEEE Information Theory Society, 1998.

[MW00] U. Maurer and S. Wolf. The Diffie-Hellman protocol. *Designs, Codes, and Cryptography*, 19:147–171, 2000.

[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.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[OP01] T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *Proc. 2001 International Workshop on Practice and Theory in Public Key Cryptography (PKC 2001)*, 2001.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology–Eurocrypt '99*, pages 223–238, 1999.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for digital signatures. In *21st Annual ACM Symposium on Theory of Computing*, 1990.

[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. To appear, *J. Cryptology.*

[SHA95] Secure hash standard, National Institute of Standards and Technology (NIST), FIPS Publication 180-1, April 1995.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology–Eurocrypt '97*, pages 256–266, 1997.

[Sho99] V. Shoup. On formal models for secure key exchange. Cryptology ePrint Archive, Report 1999/012, 1999. `http://eprint.iacr.org`.

[Sho00a] V. Shoup. A composition theorem for universal one-way hash functions. In *Advances in Cryptology–Eurocrypt 2000*, 2000.

[Sho00b] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in Cryptology–Eurocrypt 2000*, pages 275–288, 2000.

[Sho01] V. Shoup. OAEP reconsidered. In *Advances in Cryptology–Crypto 2001*, pages 239–259, 2001.

[Sim98] D. Simon. Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In *Advances in Cryptology–Eurocrypt '98*, pages 334–345, 1998.

[Sma99] N. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12(3):193–196, 1999.

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

[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.