J. Cryptol. (2010) 23: 91–120
DOI: 10.1007/s00145-009-9051-4

Journal of
CRYPTOLOGY

A New and Improved Paradigm for Hybrid Encryption Secure Against Chosen-Ciphertext Attack

Yvo Desmedt
Dept. of Computer Science, University College London, London, UK

Rosario Gennaro
IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
rosario@us.ibm.com

Kaoru Kurosawa
Dept. of Computer and Information Sciences, Ibaraki University, Ibaraki, Japan
kurosawa@mx.ibaraki.ac.jp

Victor Shoup
Computer Science Dept., NYU, New York, USA
shoup@cs.nyu.edu

Communicated by Ronald Cramer

Received 17 August 2007 and revised 26 July 2009
Online publication 11 September 2009

Abstract. We present a new encryption scheme which is secure against adaptive chosen-ciphertext attack (or CCA2-secure) in the standard model (i.e., without the use of random oracle). Our scheme is a hybrid one: it first uses a public-key step (the *Key Encapsulation Module* or *KEM*) to encrypt a random key, which is then used to encrypt the actual message using a symmetric encryption algorithm (the *Data Encapsulation Module* or *DEM*).

Our scheme is a modification of the hybrid scheme presented by Shoup in (Eurocrypt'97, Springer LNCS, vol. 1233, pp. 256–266, 1997) (based on the Cramer–Shoup scheme in CRYPTO'98, Springer LNCS, vol. 1462, pp. 13–25, 1998). Its major practical advantage is that it saves the computation of one exponentiation and produces shorter ciphertexts.

This efficiency improvement is the result of a surprising observation: previous hybrid schemes were proven secure by proving that both the KEM and the DEM were CCA2-secure. On the other hand, our KEM is not CCA2-secure, yet the whole scheme is, assuming the Decisional Diffie–Hellman (DDH) Assumption.

Finally we generalize our new scheme in two ways: (i) we show that security holds also if we use projective hash families (as the original Cramer–Shoup), and (ii) we show that in the random oracle model we can prove security under the weaker Computational Diffie–Hellman (CDH) Assumption.

Key words. Public key encryption, Chosen ciphertext security, Projective hash proofs.

1. Introduction

The notion of *chosen-ciphertext security* was introduced by Naor and Yung [13] and developed by Rackoff and Simon [15] and Dolev, Dwork, and Naor [8].

In a chosen ciphertext attack, the adversary is given access to a *decryption oracle* that allows him to obtain the decryptions of ciphertexts of his choosing. Intuitively, security in this setting means that an adversary obtains (effectively) no information about encrypted messages, provided that the corresponding ciphertexts are never submitted to the decryption oracle.

As shown in [8], security against chosen-ciphertext attack is equivalent to the notion of *nonmalleability*. An encryption scheme is said to be nonmalleable if given a ciphertext c, it is infeasible to compute a ciphertext c' whose decryption is somehow related to the decryption of c.

For these reasons, the notion of chosen-ciphertext security has emerged as the "right" notion of security for encryption schemes. Indeed it can be shown that in order to model encryption as a "secure envelope," then the encryption scheme used must be chosen-ciphertext secure.

A number of chosen ciphertext secure cryptosystems have been proposed in the literature. The first schemes were presented in [8,13,15], but they were quite impractical. The first truly practical cryptosystem that is provably secure against chosen ciphertext attack was discovered by Cramer and Shoup [4]. The security of this scheme is based on the hardness of the decisional Diffie–Hellman problem. In [5] Cramer and Shoup show that their original scheme is an instance of a more generic paradigm, which can be also instantiated with the Quadratic Residuosity and N-Residuosity assumptions.

In [18] Shoup presented an *hybrid* variant of the Cramer–Shoup cryptosystem. In a hybrid encryption scheme public-key encryption techniques are used to derive a shared key between sender and receiver: this component is called the *Key Encapsulation Module* or *KEM*. This key is then used to encrypt the actual message via symmetric-key techniques (the *Data Encapsulation Module* or *DEM*). Shoup formalized the notion of a chosen-ciphertext attack security for a KEM and showed that if both KEM and DEM are chosen-ciphertext secure, then the resulting hybrid encryption also is.

Differently than in the public-key case, symmetric encryption schemes which are secure against a chosen-ciphertext attack can be easily built out of weaker primitives. It is indeed well known that all you need is a symmetric encryption scheme E which is secure against passive adversaries, and a secure message authentication code (MAC). To encrypt a message m with keys k, K it is sufficient to encrypt m with K, i.e., compute $e = E_K(m)$, and then compute a message authentication tag for e using k, i.e., compute $t = \mathsf{MAC}_k(e)$. The final ciphertext is (e, t). The receiver, who also holds k, K, first checks that the tag t is correct and only in that case decrypts e.

1.1. *Our Contribution*

We present a new encryption scheme which is secure against adaptive chosen-ciphertext attack (or CCA2-secure) in the standard model (i.e., without the use of random oracle). Our scheme is a hybrid one: it first uses a public-key step (the *Key Encapsulation Module* or *KEM*) to encrypt a random key, which is then used to encrypt the actual message using a symmetric encryption algorithm (the *Data Encapsulation Module* or *DEM*).

Our scheme is a modification of the hybrid scheme presented by Shoup in [18] (based on the Cramer–Shoup scheme in [4]). Its major practical advantage is that it saves the computation of one exponentiation and produces shorter ciphertexts.

This efficiency improvement is the result of a surprising observation: previous hybrid schemes were proven secure by proving that both the KEM and the DEM were CCA2-secure. On the other hand, our KEM is not CCA2-secure, yet the whole scheme is, assuming the Decisional Diffie–Hellman (DDH) Assumption.

Finally we generalize our new scheme in two ways: (i) we show that security holds also if we use projective hash families (as the original Cramer-Shoup), and (ii) we show that in the random oracle model we can prove security under the weaker Computational Diffie–Hellman (CDH) Assumption.

This paper combines two separate results, presented in preliminary form in conference papers [9,12].

In the first result, Kurosawa and Desmedt [12] modified the hybrid scheme presented in [18]. In particular they modified Shoup's KEM to one which they did not know how to prove CCA2-secure. Yet, they were able to prove that the whole hybrid encryption scheme was CCA2-secure. The advantage of their modification is that, in theory, the encryption step in their scheme requires one less exponentiation and produces shorter ciphertexts (compared to Shoup's hybrid scheme).

However their proof of security relies on the use of information theoretically secure components in the symmetric step of the hybrid construction (specifically the key derivation function and the MAC). There are several reasons why this is not desirable, among them:

Efficiency The proof in [12] requires the key k to be statistically close to a random key. This means that we cannot use a pseudo-random generator to derive k from a random group element encrypted during the public-key phase. This in turn implies that the public key part of the scheme must be instantiated with larger security parameters which would result in slower execution times.[1]

Modularity we would like to have a scheme into which we can plug any secure component and it still remains secure. It would be hard to deploy a scheme if it can be used only in conjunction with certain types of MACs and Key Derivation Function (KDFs—and in particular, with KDFs and MACs that are not used at all by the designers of standard cryptographic algorithms).

The second result presented in this paper is an alternative security proof for the Kurosawa–Desmedt scheme shown by Gennaro and Shoup in [9]. This proof removes the need for information theoretically secure key derivation functions and message authentication codes, thereby effectively improving the efficiency and applicability of their scheme.

[1] For typical security parameters, this increase in computation times totally offsets the gain from performing one less exponentiation, thus making the Kurosawa–Desmedt scheme as efficient as the original Cramer–Shoup.

1.2. *Related Work*

In our preliminary versions [9,12] we were not able to prove or disprove the CCA2-security of the Kurosawa–Desmedt KEM. Recently Herranz et al. [10] proved that this KEM is actually *not* CCA2-secure.

An interesting question is, then, what property does the Kurosawa–Desmedt KEM satisfy that allows it to be "secure enough" to construct a fully CCA2-secure hybrid scheme? Abe et al. [1] give a possible answer to this question. They formalize the notion of *Tag-KEM* in which the Key Encapsulation Module is also given an external input (a tag). They define the notion of CCA2-security for Tag-KEMs and show that the Kurosawa–Desmedt scheme can be "explained" as a CCA2-secure Tag-KEM combined with a semantically-secure DEM. A different approach was taken by Hofheinz and Kiltz in [11], where they define the notion of *constrained* CCA-security for a KEM and show that, together with a (one-time) secure authenticated DEM, this yields a CCA2-secure hybrid encryption scheme, which includes Kurosawa–Desmedt as an example.

2. Preliminaries

We denote by n a security parameter. PPT denotes probabilistic polynomial time.

If S is a set, by $|S|$ we denote its cardinality. If m is a string or a number, $|m|$ denotes its bit length.

If $A(\cdot,\cdot,\ldots)$ is a probabilistic algorithm, then $x \xleftarrow{R} A(x_1, x_2, \ldots)$ denotes the experiment of running A on input $x_1, x_2, \ldots$ with x being the outcome. If S is a set, $x \xleftarrow{R} S$ denotes the experiment of choosing $x \in S$ uniformly at random. If X is a probability distribution over S, then $x \xleftarrow{R} X$ denotes the experiment of choosing $x \in S$ according to the distribution X.

We say that a real-valued function $\epsilon(\cdot)$ defined over the integers is *negligible* if for every constant $c \geq 0$, there exists an integer n_c such that for all $n > n_c$, $\epsilon(n) < n^{-c}$.

Finally we refer to probabilistic polynomial-time algorithms as *efficient algorithms*.

2.1. *Public Key Encryption*

A public key encryption scheme is a tuple of three algorithms $\mathsf{PKE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key generation algorithm $\mathcal{K}$ generates a pair $(pk, sk) \xleftarrow{R} \mathcal{K}(1^n)$, where pk is a public key, and sk is a secret key.

The encryption algorithm $\mathcal{E}$ takes a public key pk and a plaintext m, and returns a ciphertext $c \xleftarrow{R} \mathcal{E}_{pk}(m)$. The decryption algorithm $\mathcal{D}$ takes a secret key sk and a ciphertext c, and returns $\mathcal{D}_{sk}(c)$ which is either a message m or *reject*.

The chosen plaintext attack (CPA) game is defined as follows. A key pair is generated by the key generation algorithm: $(pk, sk) \xleftarrow{R} \mathcal{K}(1^n)$. Then a PPT adversary A, on input the public key pk, queries a pair of equal length messages m_0 and m_1 to an encryption oracle. The encryption oracle chooses $b \xleftarrow{R} \{0, 1\}$ and computes a challenge ciphertext $c^* \xleftarrow{R} \mathcal{E}_{pk}(m_b)$, which is given to A. The game ends with A outputting a bit $\tilde{b}$.

The adaptive chosen ciphertext attack (CCA2) game is defined similarly. The difference is that the adversary A is given access to a decryption oracle, $\mathcal{D}_{sk}(\cdot)$, which A can query on any ciphertext except the challenge ciphertext c^*.

We define the CCA2 advantage of A as a function of the security parameter as follows:

$$\mathtt{Adv}^{cca}_{A,\mathsf{PKE}}(n) \stackrel{\triangle}{=} \left|\Pr(\tilde{b} = b) - 1/2\right|. \tag{1}$$

Definition 1. We say that PKE is secure against adaptive chosen ciphertext attack (or CCA2-secure) if for all efficient adversaries A, we have that $\mathtt{Adv}^{cca}_{A,\mathsf{PKE}}(n)$ is negligible.

2.2. *Symmetric Key Encryption*

A symmetric key encryption scheme is a pair of algorithms $\mathsf{SKE} = (E, D)$.

The encryption algorithm E takes a secret key $k \in \{0,1\}^n$ and a plaintext m, and returns a ciphertext $c \stackrel{R}{\leftarrow} E_k(m)$. The decryption algorithm D takes the secret key k and a ciphertext c, and returns $D_k(c)$ which is either a message m or *reject*.

The one-time chosen plaintext attack (one-time CPA) game is defined as follows. A secret key is generated uniformly and at random $k \stackrel{R}{\leftarrow} \{0,1\}^n$. Then a PPT adversary A queries a pair of equal length messages m_0 and m_1 to an encryption oracle. The encryption oracle chooses $b \stackrel{R}{\leftarrow} \{0,1\}$ and computes a challenge ciphertext $c^* \stackrel{R}{\leftarrow} E_k(m_b)$, which is given to A. The game ends with A outputting a bit $\tilde{b}$.

We define the one-time CPA advantage of A as a function of the security parameter as follows:

$$\mathtt{Adv}^{cpa1}_{A,\mathsf{SKE}}(n) \stackrel{\triangle}{=} \left|\Pr(\tilde{b} = b) - 1/2\right|. \tag{2}$$

Definition 2. We say that SKE is one-time semantically secure (or one-time CPA-secure) if for all efficient adversaries A, we have that $\mathtt{Adv}^{cpa1}_{A,\mathsf{SKE}}(n)$ is negligible.

2.3. *Message Authentication Codes*

A message authentication code MAC is a function

$$\mathsf{MAC} : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^{\ell(n)}$$

for some polynomial $\ell(\cdot)$. The first input is the key $k \in \{0,1\}^n$, and the second input is the message $m \in \{0,1\}^*$. The output is called a "tag" $t := \mathsf{MAC}_k(e)$.

The one-time chosen message attack (CMA1) game is defined as follows. A key is selected uniformly at random $k \stackrel{R}{\leftarrow} \{0,1\}^n$. The adversary A is given $t^* := \mathsf{MAC}_k(e^*)$ for (at most one) adversarially chosen e^*, after which the adversary outputs a pair (e, t). We say that (e, t) is a *forgery* if $e \neq e^*$ and $t = \mathsf{MAC}_k(e)$. Informally we say that a MAC is secure if it is hard to compute a forgery.[2]

[2] Since we are defining MAC as a function, there is only one possible output for any input pair k, e. It is possible to define message authentication codes as two algorithms: a "tagging" and a "verifying" algorithm. The tagging algorithm could be randomized, and thus one of several tags could be computed on the same

More formally we define the success probability of A in the CMA1 game as

$$\mathtt{Succ}^{cma1}_{A,\mathsf{MAC}}(n) \stackrel{\triangle}{=} \Pr\big(e \neq e^* \wedge t = \mathsf{MAC}_k(e)\big). \tag{3}$$

Definition 3. We say that MAC is one-time unforgeable if for all efficient adversaries A, we have that $\mathtt{Succ}^{cma1}_{A,\mathsf{MAC}}(n)$ is negligible.

2.4. *Key Derivation Functions*

Note that we defined the secret keys for the message authentication code and the symmetric encryption scheme to consist of all bit strings of a given length. In the scheme we propose these keys will have to be derived from a secret value which belongs to a given domain (in our case a cyclic group of prime order). For this reason, we are going to use a *key derivation function* KDF:

$$\mathsf{KDF} : \Delta \to \{0,1\}^{2n},$$

where Δ is a given domain. If $v \in \Delta$, then $\mathsf{KDF}(v) = (k_{mac}, k_{enc})$, where k_{mac} is a message authentication key, and k_{enc} is a symmetric encryption key.

The security assumption we make on KDF is that a PPT adversary cannot distinguish between $\mathsf{KDF}(v)$ and (k_1, k_2), where v, k_1, and k_2 are randomly chosen.

More formally let A be a PPT adversary that is given as input two n-bit strings and outputs a bit. We define his advantage as

$$\mathtt{Adv}_{A,\mathsf{KDF}}(n) \stackrel{\triangle}{=} \Big|\Pr_{v \stackrel{R}{\leftarrow} \Delta}\big(A\big(\mathsf{KDF}(v)\big) = 1\big) - \Pr_{(k_1,k_2) \stackrel{R}{\leftarrow} \{0,1\}^{2n}}\big(A(k_1,k_2) = 1\big)\Big|. \tag{4}$$

Definition 4. We say that KDF is a secure key derivation function if for all efficient adversaries A, we have that $\mathtt{Adv}_{A,\mathsf{KDF}}(n)$ is negligible.

2.5. *Target Collision Resistant Hash Functions*

Let $\mathcal{H}$ be a family of hash functions where each function $H \in \mathcal{H}$ is from an arbitrary domain Δ to the set of n-bit strings.

We say that $\mathcal{H}$ is target-collision resistant (TCR)—a concept introduced by Naor and Yung [14]—if given a randomly chosen element in the domain $x \stackrel{R}{\leftarrow} \Delta$ and a randomly chosen hash function $H \stackrel{R}{\leftarrow} \mathcal{H}$, it is infeasible for an adversary A to find $y \neq x$ such that $H(x) = H(y)$.[3]

Define the success probability of the adversary A in finding a collision as

$$\mathtt{Succ}^{tcr}_{A,\mathcal{H}}(n) \stackrel{\triangle}{=} \Pr_{x \stackrel{R}{\leftarrow} \Delta, H \stackrel{R}{\leftarrow} \mathcal{H}}\big(A(x,H) = y : y \neq x \wedge H(y) = H(x)\big).$$

input pair. The security property then would be that it is hard to compute any valid message/tag pair (e,t) other than (e^*,t^*).

[3] Naor and Yung in [14] actually introduce a stronger notion—universal one-way hash function families—where the target x is chosen by the adversary before the function H is selected.

Definition 5. We say that $\mathcal{H}$ is a secure TCR hash function family if for any efficient adversary A, $\mathtt{Succ}^{tcr}_{A,\mathcal{H}}(n)$ is negligible.

It is known that TCR families can be built from arbitrary one-way functions [14,16]. In practice, one can use a dedicated cryptographic hash function, like SHA-1, appropriately keyed.

2.6. *Diffie–Hellman Assumption*

Let $\mathcal{G} = \{G_n\}$ be a family of Abelian groups where each G_n has prime order $q_n > 2^n$. We assume that multiplication and testing group membership in G_n can be efficiently performed.

The *Computational Diffie–Hellman* (*CDH*) *Assumption* [7] states that given g, h random generators of G_n, and $u = g^r$ for $r \overset{R}{\leftarrow} \mathbb{Z}_{q_n}$, it is infeasible to compute $z = h^r$. More formally, for any adversary A, define

$$\mathtt{Succ}^{cdh}_{A,G_n}(n) = \Pr_{r \overset{R}{\leftarrow} \mathbb{Z}_{q_n}}\left(A\left(g, h, u = g^r\right) = h^r\right).$$

Definition 6. We say that the CDH Assumption holds over the group family $\mathcal{G}$ if for all efficient adversaries A, we have that $\mathtt{Succ}^{cdh}_{A,G_n}(n)$ is negligible.

The *Decisional* Diffie–Hellman Assumption is stronger than the CDH in that it assumes that the value $z = h^r$ is even hard to *recognize* when given to the adversary. Given g_1, g_2 random generators of G, consider the following distributions over G_n^4:

$$DH_n = \left\{\left(g_1, g_2, g_1^r, g_2^r\right) \mid r \overset{R}{\leftarrow} \mathbb{Z}_{q_n}\right\},$$

$$NDH_n = \left\{\left(g_1, g_2, g_1^{r_1}, g_2^{r_2}\right) \mid r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_q \text{ with } r_1 \neq r_2\right\}.$$

The Decisional Diffie–Hellman (DDH) assumption claims that *DH* and *NDH* are computationally indistinguishable.

For an adversary A, we define his advantage as follows:

$$\mathtt{Adv}^{ddh}_{A,G_n}(n) \overset{\triangle}{=} \Big|\Pr_{(g_1,g_2,u_1,u_2) \overset{R}{\leftarrow} DH_n}\left(A(g_1, g_2, u_1, u_2) = 1\right) - \Pr_{(g_1,g_2,u_1,u_2) \overset{R}{\leftarrow} NDH_n}\left(A(g_1, g_2, u_1, u_2) = 1\right)\Big|.$$

Definition 7. We say that the DDH Assumption holds over the group family $\mathcal{G}$ if for all efficient adversaries A, we have that $\mathtt{Adv}^{ddh}_{A,G_n}(n)$ is negligible.

Our formulation of the DDH Assumption assumes that it is hard to distinguish between a random DH tuple (a random element of G_n^4 sampled according to DH_n) and a random non-DH tuple (a random element of G_n^4 sampled according to NDH_n). The more typical formulation considers distinguishing between DH_n and the $Rand_n$ distribution that samples u_1, u_2 uniformly at random and independently in G_n. Clearly the two

formulations are equivalent up to the negligible additive factor of $1/q_n$ which accounts for the statistical difference between the NDH_n and $Rand_n$ distributions.

In the following we may omit the security parameter n when clear from the context.

3. The Scheme

This section describes the Kurosawa–Desmedt hybrid encryption scheme, which we denote as $\mathcal{KD}$. The scheme uses a cyclic group G of prime order, a symmetric encryption scheme $\mathsf{SKE} = (E, D)$, a TCR hash function family $\mathcal{H}$ (defined over the set G^2), and a key derivation function KDF (defined over G).

Key Generation The description of a cyclic group G of prime order q is generated, along with random generators g_1 and g_2 for G. We assume that multiplication and group membership can be efficiently performed in G. We also choose $H \stackrel{R}{\leftarrow} \mathcal{H}$ where $\mathcal{H}$ is a TCR hash function family. Also a key derivation function KDF is selected (if keys are needed for it, they are appropriately generated). Then perform the following steps:

$$x_1, x_2, y_1, y_2 \stackrel{R}{\leftarrow} \mathbb{Z}_q, \qquad c \leftarrow g_1^{x_1} g_2^{x_2}, \qquad d \leftarrow g_1^{y_1} g_2^{y_2}.$$

The public key consists of the description of G, the generators g_1 and g_2, the functions KDF and H, along with the group elements c and d.

The private key consists of the public key, along with x_1, x_2, y_1, y_2.

Encryption of $m \in \{0,1\}^*$

$$\begin{array}{l} r \stackrel{R}{\leftarrow} \mathbb{Z}_q,\ u_1 \leftarrow g_1^r \in G,\ u_2 \leftarrow g_2^r \in G,\ \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q \\ v \leftarrow c^r d^{r\alpha} \in G,\ (k, K) \leftarrow \mathsf{KDF}(v),\ e \leftarrow E_K(m),\ t \leftarrow \mathsf{MAC}_k(e) \\ \text{output } C := (u_1, u_2, e, t) \end{array}$$

Decryption of $C = (u_1, u_2, e, t)$

$$\begin{array}{l} \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q,\ v \leftarrow u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \in G,\ (k, K) \leftarrow \mathsf{KDF}(v) \\ \text{if } t \neq \mathsf{MAC}_k(e) \text{ then} \\ \quad \text{output "reject"} \\ \text{else} \\ \quad m \leftarrow D_K(e) \\ \quad \text{output } m \end{array}$$

Theorem 1. *Assume that* (i) *the DDH Assumption holds over* G (*a group of prime order* q); (ii) $\mathsf{SKE} = (E, D)$ *is a one-time semantically secure symmetric encryption scheme*; (iii) MAC *is a one-time unforgeable MAC*; (iv) $\mathcal{H}$ *is a TCR hash function family*; *and* (*v*) KDF *is a secure key derivation function. Then the encryption scheme* $\mathcal{KD}$ *described above is secure against adaptive chosen ciphertext attack.*

4. Security Proof

This section presents the Gennaro–Shoup proof of Theorem 1 first presented in [9]. The main difference between this proof and the original proof by Kurosawa–Desmedt in [12] is that the latter requires the following *information theoretic* assumptions:

- *information-theoretically secure* KDF. If $v \in G$ is random, then at least the first component k of the output of $\mathsf{KDF}(v)$ should be (statistically close to) uniform.
- *information-theoretically secure MAC.* For all e and t, if k is chosen at random, then $\Pr[\mathsf{MAC}_k(e) = t]$ is negligible.

The proof of security described in the next section does *not* need these assumptions.

Kurosawa and Desmedt [12] apparently introduce these assumptions to avoid a potential circularity in their proof. Both their proof and ours use a "hybrid argument," whereby the initial attack game is transformed, using a sequence of small steps, into a game in which the adversary's advantage is clearly negligible.

At one point in their proof, in order to justify one of the steps in this sequence, they must prove that the decryption oracle will reject a certain type of "invalid" ciphertexts. We say that a ciphertext is *invalid* if $u_i = g_i^{r_i}$ with $r_1 \neq r_2 \bmod q$. In the $\mathcal{KD}$ scheme, ciphertexts are rejected only based on the MAC test. Thus the proof uses a reduction to the unforgeability of the MAC function: this requires a proof that the MAC keys are randomly distributed. But to draw this conclusion, they must first prove that invalid ciphertexts are rejected.

As it happens, one possible way out of this circularity is to introduce information-theoretic security assumptions. However, we show how to avoid this using two tools. First, we use the fact that the DDH problem has two "independent trapdoors": if one knows $w = \log_{g_1} g_2$, then it is possible to check if $\log_{g_1} u_1 = \log_{g_2} u_2 = r$ (by testing if $u_1^w = u_2$) even without knowing r. By giving this value w to the simulator in one of the games, we have a way to detect and reject invalid ciphertexts, without using the MAC test (Game 3 in the proof below). We then need to show that using w instead of the MAC test to reject invalid ciphertext does not change the adversary view of the game in a substantial way (Game $5'$ in the proof). Here we use a new technique that is perhaps not so well appreciated, which we might call "deferred analysis." We will point out below in the proof where this technique is employed. This technique has also been used before—for example, in [6], it is used to refine the proof of security of the Cramer–Shoup encryption scheme by requiring a universal one-way hash function (rather than a collision-resistant hash function) for the computation of α.

In Appendix we recall the Cramer–Shoup hybrid scheme from [18] and compare the two schemes. In particular we point out how for typical security parameters, the gains posted by the Kurosawa–Desmedt scheme may be offset by the requirement that KDF and MAC be information theoretically secure. Thus the importance of the proof presented here is not merely theoretical, but it affects the practical efficiency of the new scheme.

4.1. *Proof of Theorem 1*

Our proof, like the original one by Kurosawa and Desmedt, is also constructed as a sequence of Games. The first game, Game 0, is basically identical to the CCA2 game,

and the adversary's advantage is defined accordingly. In the last Game, the adversary will still have to guess a given bit (as in the CCA2 game), but in this last game, the bit is information-theoretically hidden to the adversary, so his advantage must be exactly $1/2$. To go from the first to the last game, we define various intermediate games. Under the assumptions of Theorem 1, each Game i must be "very similar" to Game $i-1$, meaning that the adversary's advantage in Game i is bounded away from his advantage in Game i by at most a negligible quantity. This is sufficient to complete the proof.

We start with the following simple but useful Lemma.

Lemma 1 [6, Lemma 6.2]. *Let S_1, S_2, and F be events defined on some probability space. Suppose that the event $S_1 \wedge \neg F$ occurs if and only if $S_2 \wedge \neg F$ occurs. Then*

$$\left|\Pr(S_1) - \Pr(S_2)\right| \leq \Pr(F).$$

Game 0

We now define a game, called *Game 0*, which is an interactive computation between an *adversary* A and a *simulator*. This game is simply the usual CCA2 game used to define CCA2-security, in which the simulator provides the adversary's environment.

Initially, the simulator runs the key generation algorithm, obtaining the description of G, generators g_1 and g_2, keys for KDF and H (if any), along with the values $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $c, d \in G$. The simulator gives the public key to the adversary.

During the execution of the game, the adversary makes a number of "decryption requests." Assume that these requests are $C^{(1)}, \ldots, C^{(Q)}$, where

$$C^{(i)} = \left(u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)}\right).$$

For each such request, the simulator decrypts the given ciphertext and gives the adversary the result. We denote by $\alpha^{(i)}$, $v^{(i)}$, $k^{(i)}$, and $K^{(i)}$ the corresponding intermediate quantities computed by the decryption algorithm on input $C^{(i)}$.

The adversary may also make a single "challenge request." For such a request, the adversary submits two messages m_0, m_1, which are bit strings of equal length, to the simulator; the simulator chooses $b \in \{0, 1\}$ at random and encrypts m_b, obtaining the "target ciphertext" $C^* = (u_1^*, u_2^*, e^*, t^*)$. The simulator gives C^* to the adversary. We denote by r^*, α^*, v^*, k^*, and K^* the corresponding intermediate quantities computed by the encryption algorithm.

The only restriction on the adversary's requests is that after it makes a challenge request, the subsequent decryption requests must not be the same as the target ciphertext.

At the end of the game, the adversary A outputs $\tilde{b} \in \{0, 1\}$. Let X_0 be the event that $\tilde{b} = b$. Since Game 0 is identical to the CCA2 game, we have that

$$\mathrm{Adv}^{cca}_{A,\mathcal{K}D}(n) = \left|\Pr[X_0] - 1/2\right|, \tag{5}$$

and our goal is to prove that this quantity is negligible.

We prove this by considering other games, *Game 1*, *Game 2*, etc. These games will be quite similar to Game 0 in their overall structure and will only differ from Game 0 in terms of how the simulator works. However, in each game, there will be well-defined

bits $\tilde{b}$ and b, so that in Game i, we always define X_i to the event that $\tilde{b} = b$ in that game. All of these games should be viewed as operating on the same underlying probability space.

Before moving on, we make a couple of additional assumptions about the internal structure of Game 0 that will be convenient down the road. First, we assume that g_2 is computed as

$$w \xleftarrow{R} \mathbb{Z}_q^*, \qquad g_2 \leftarrow g_1^w.$$

Note that the value of w is never explicitly used in Game 0, except to compute g_2. Second, we assume that the quantities r^*, u_1^*, u_2^*, α^*, v^*, k^*, and K^* are computed at the very start of the game (they do not depend on the values m_0, m_1 provided later by the adversary, so this can be done).

Finally we assume that the simulator computes v^* as

$$v^* \leftarrow \left(u_1^*\right)^{x_1 + y_1 \alpha^*} \left(u_2^*\right)^{x_2 + y_2 \alpha^*}.$$

This change is purely conceptual, since v^* has the same value either way.

These changes do not affect the view of the adversary, which during Game 0 remains identical to the view during the CCA2 games, thus (5) still holds.

Game 1

In Game 1 the simulator changes the way it generates (u_1^*, u_2^*), which will be computed according to the following rule:

$$r_1, r_2 \xleftarrow{R} \mathbb{Z}_q \quad \text{with } r_2 \neq r_1, \qquad u_i^* \leftarrow g_i^{r_i} \quad \text{for } i = 1, 2.$$

Lemma 2. *There exists an efficient adversary A_1 such that*

$$\left|\Pr[X_1] - \Pr[X_0]\right| \leq \mathtt{Adv}^{ddh}_{A_1, G}(n).$$

By the assumption that the DDH assumption holds over G, we have that $\mathtt{Adv}^{ddh}_{A_1, G}(n)$ is negligible.

Proof. Let $\epsilon = |\Pr[X_1] - \Pr[X_0]|$. We can easily build a distinguisher A_1 that decides the DDH problem in G with advantage ϵ.

The distinguisher A_1 runs on input $\tau := (g_1, g_2, u_1^*, u_2^*)$. It then runs a "hybrid Game 0/1" as the simulator. In this hybrid game, the simulator follows the instructions of Game 0, except that when the target ciphertext is created, it uses (u_1^*, u_2^*).

Thus if τ is a random DH tuple, Game 0/1 acts just like Game 0, and if τ is a random non-DH tuple, then Game 0/1 acts just like Game 1. The distinguishing algorithm runs Game 0/1 on input τ, and outputs 1 if $\hat{b} = b$ and outputs 0 otherwise. The distinguishing advantage of this algorithm is exactly equal to ϵ.

By the definition of $\mathtt{Adv}^{ddh}_{A_1, G}(n)$, $\epsilon \leq \mathtt{Adv}^{ddh}_{A_1, G}(n)$. □

Game 2

Game 2 is the same as Game 1, except that if the adversary ever submits $C^{(i)}$ for decryption with

$$\left(u_1^{(i)}, u_2^{(i)}\right) \neq \left(u_1^*, u_2^*\right) \quad \text{and} \quad \alpha^{(i)} = \alpha^*,$$

the simulator *rejects* the given ciphertext.

In Game 2, the simulator may reject ciphertexts that would not have been rejected in Game 1. Let us call *Rejection Rule 0* the rule by which ciphertexts are rejected as in the ordinary decryption algorithm (i.e., the message authentication tags do not match). Let us call *Rejection Rule 1* this new rejection rule, introduced in Game 2.

Let F_2 be the event that the simulator applies Rejection Rule 1 in Game 2 to a ciphertext to which Rejection Rule 0 does not apply. Game 1 and Game 2 proceed identically until this event occurs; in particular, the events $X_1 \wedge \neg F_2$ and $X_2 \wedge \neg F_2$ are the same (recall that we view all games as operating on the same underlying probability space); therefore, using Lemma 1, we get

$$\left|\Pr[X_1] - \Pr[X_2]\right| \leq \Pr[F_2]. \tag{6}$$

We now bound $\Pr[F_2]$ as follows.

Lemma 3. *There exists an efficient adversary A_2 such that* $\Pr[F_2] \leq \mathtt{Succ}^{tcr}_{A_2,\mathcal{H}}(n) + 1/q$.

By the assumption that $\mathcal{H}$ is a TCR family, then we have that $\Pr[F_2]$ is negligible.

Proof. We build a collision-finder A_2 that breaks the TCR property of $\mathcal{H}$ with probability $\Pr[F_2] - 1/q$. By the definition of $\mathtt{Succ}^{tcr}_{A_2,\mathcal{H}}(n)$, the lemma follows.

The collision finder A' is given as input a random function $H \in \mathcal{H}$ and a random point $(u_1^*, u_2^*) \in G^2$ (remember that the functions in $\mathcal{H}$ are defined over G^2).

A' runs Game 2 as the simulator and will use (u_1^*, u_2^*) to construct the target ciphertext in response to an encryption query. Notice that if the input pair (u_1^*, u_2^*) is a non-DH pair (i.e., $\log_{g_1} u_1^* \neq \log_{g_2} u_2^*$), this is exactly how Games 1 and 2 are defined (the target ciphertext is created with a random non-DH pair (u_1^*, u_2^*)).

If the pair (u_1^*, u_2^*) is a non-DH pair, then with probability $\Pr[F_2]$ the adversary outputs a pair

$$\left(u_1^{(i)}, u_2^{(i)}\right) \neq \left(u_1^*, u_2^*\right) \quad \text{and} \quad \alpha^{(i)} = \alpha^*,$$

and the collision finder outputs $(u_1^{(i)}, u_2^{(i)})$ as the desired collision.

By accounting for the $1/q$ probability that the input pair (u_1^*, u_2^*) is a DH-pair (i.e., $\log_{g_1} u_1^* = \log_{g_2} u_2^*$) we have that the collision finder works with probability $\Pr[F_2] - 1/q$, and the lemma follows. □

Game 3

In this game, the simulator makes use of the value $w \in \mathbb{Z}_q$, where $g_2 = g_1^w$. The simulator did not need to make explicit use of this value in previous games. Indeed, we could

not have used the DDH assumption if the simulator had to use w. However, we are now finished with the DDH assumption, and so the simulator is free to make use of w in this and subsequent games.

Game 3 is the same as Game 2, except that we introduce a new *Rejection Rule 2*: in responding to decryption requests, the simulator *rejects* any ciphertext $C^{(i)}$ such that

$$\left(u_1^{(i)}, u_2^{(i)}\right) \neq \left(u_1^*, u_2^*\right) \quad \text{and} \quad \left(u_1^{(i)}\right)^w \neq u_2^{(i)}.$$

Note that the condition $(u_1^{(i)})^w \neq u_2^{(i)}$ is equivalent to $\log_{g_1} u_1^{(i)} \neq \log_{g_2} u_2^{(i)}$.

Define F_3 to be the event that a ciphertext is rejected during Game 3 using Rejection Rule 2 to which Rejection Rules 0 and 1 are not applicable.

Clearly, Game 2 and Game 3 proceed identically until F_3 occurs; in particular, the events $X_2 \wedge \neg F_3$ and $X_3 \wedge \neg F_3$ are the same, and so

$$\left|\Pr[X_2] - \Pr[X_3]\right| \leq \Pr[F_3]. \tag{7}$$

We want to show that $\Pr[F_3]$ is negligible; however, we postpone this until later. This is the "deferred analysis" technique: instead of attempting to bound $\Pr[F_3]$ right now, we shall patiently wait until a later game, where it will be much easier. However, at this point we augment Game 3 just slightly, utilizing the well-known "plug and pray" technique: the simulator chooses $j \in \{1, \ldots, Q\}$ at random, and we define F_3' to be the event that in Game 3, Rejection Rules 0 and 1 do not apply to $C^{(j)}$, but Rejection Rule 2 does apply to $C^{(j)}$. Clearly,

$$\Pr[F_3] \leq Q \Pr[F_3'], \tag{8}$$

and so it suffices to show that $\Pr[F_3']$ is negligible.

It will be helpful to write down in detail how Game 3 works, starting from scratch:

- The simulator begins by generating the description of G, along with a random generator g_1, and any keys for KDF and H. It then computes:

  I1: $w \xleftarrow{R} \mathbb{Z}_q^*,\ g_2 \leftarrow g_1^w$
  I2: $x_1, x_2, y_1, y_2 \xleftarrow{R} \mathbb{Z}_q,\ c \leftarrow g_1^{x_1} g_2^{x_2},\ d \leftarrow g_1^{y_1} g_2^{y_2}$
  I3: $r_1, r_2 \xleftarrow{R} \mathbb{Z}_q$ with $r_2 \neq r_1\ u_1^* \leftarrow g_1^{r_1},\ u_2^* \leftarrow g_1^{wr_2},\ \alpha^* \leftarrow H(u_1^*, u_2^*)$
  I4: $v^* \leftarrow (u_1^*)^{x_1 + y_1\alpha^*} (u_2^*)^{x_2 + y_2\alpha^*}$
  I5: $(k^*, K^*) \leftarrow \mathsf{KDF}(v^*)$
  I6: $j \xleftarrow{R} \{1, \ldots, Q\}$

  The simulator gives the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with c and d to the adversary.
- In processing a decryption request $C^{(i)} = (u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)})$, the simulator proceeds as follows:

  D01: $\alpha^{(i)} \leftarrow H(u_1^{(i)}, u_2^{(i)})$
  D02: if $(u_1^{(i)}, u_2^{(i)}) \neq (u_1^*, u_2^*)$ and $\alpha^{(i)} = \alpha^*$ then
  D03: return "reject"
  D04: else if $(u_1^{(i)}, u_2^{(i)}) = (u_1^*, u_2^*)$ then

D05: if $t^{(i)} \neq \mathsf{MAC}_{k^*}(e^{(i)})$ then return "reject"
D06: return $D_{K^*}(e^{(i)})$
D07: else if $(u_1^{(i)})^w \neq u_2^{(i)}$ then
D08: $v^{(i)} \leftarrow (u_1^{(i)})^{x_1+y_1\alpha^{(i)}}(u_2^{(i)})^{x_2+y_2\alpha^{(i)}}$
D09: $(k^{(i)}, K^{(i)}) \leftarrow \mathsf{KDF}(v^{(i)})$
D10: if $t^{(i)} \neq \mathsf{MAC}_{k^{(i)}}(e^{(i)})$ then return "reject"
D11: return "reject"
D12: else
D13: $v^{(i)} \leftarrow (u_1^{(i)})^{x_1+y_1\alpha^{(i)}}(u_2^{(i)})^{x_2+y_2\alpha^{(i)}}$
D14: $(k^{(i)}, K^{(i)}) \leftarrow \mathsf{KDF}(v^{(i)})$
D15: if $t^{(i)} \neq \mathsf{MAC}_{k^{(i)}}(e^{(i)})$ then return "reject"
D16: return $D_{K^{(i)}}(e^{(i)})$

Note that Rejection Rule 0 is applied at lines **D05**, **D10**, and **D15**, while Rejection Rule 1 is applied at line **D03**, and Rejection Rule 2 (and no other Rejection Rule) at line **D11**.

- In processing the challenge request, the adversary gives m_0, m_1 to the simulator. The simulator computes

$$b \xleftarrow{R} \{0,1\}, \qquad e^* \leftarrow E_{K^*}(m_b), \qquad t^* \leftarrow \mathsf{MAC}_{k^*}(e^*),$$

and gives $C^* := (u_1^*, u_2^*, e^*, t^*)$ to the adversary.

We have written the logic of the decryption oracle in this particular way to facilitate further analysis. Note that the computations at **D08–D10** have no real effect, other than to determine if the event F_3' occurs; indeed, once line **D08** is reached, the ciphertext is sure to be rejected, either at line **D10** or at line **D11**. The event F_3' is simply the event that line **D11** is executed in the jth decryption request.

Game 4

Game 4 is the same as Game 3, except that we change lines **I2** and **I4** as follows:

I2: $x, y \xleftarrow{R} \mathbb{Z}_q,\ c \leftarrow g_1^x,\ d \leftarrow g_1^y$
I4: $v^* \xleftarrow{R} G$

as well as lines **D08** and **D13** as follows:

D08: $v^{(i)} \xleftarrow{R} G$
D13: $v^{(i)} \leftarrow (u_1^{(i)})^{x+y\alpha^{(i)}}$

Note that x_1, x_2, y_1, y_2 are not used at all in Game 4.

We define F_4' to be the event that line **D11** is executed in the jth decryption request in Game 4.

Lemma 4. *We claim that*

$$\Pr[X_3] = \Pr[X_4] \tag{9}$$

and

$$\Pr[F_3'] = \Pr[F_4']. \tag{10}$$

Proof. This lemma follows from a simple linear algebra argument, along the same lines as in [4]. The point is that we are simply swapping one set of 4-wise independent random variables for another; indeed, in both games, the variables c, d, v^*, and $v^{(j)}$ (as computed at line **D08**) are mutually independent and uniformly distributed over G.

More in detail: consider the variables c, d, v^*, and $v^{(j)}$ in Game 3. Consider the variables $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$: they satisfy the following equations defined by the public key:

$$x_1 + wx_2 = \log_{g_1} c \bmod q, \tag{11}$$

$$y_1 + wy_2 = \log_{g_1} d \bmod q. \tag{12}$$

We also have an equation defined by the value of v^*:

$$r_1 x_1 + wr_2 x_2 + \alpha^*(r_1 y_1 + wr_2 y_2) = \log_{g_1} v^* \bmod q. \tag{13}$$

If the adversary submits a decryption request $C^{(i)}$ such that

$$\left(u_1^{(i)}, u_2^{(i)}\right) \neq \left(u_1^*, u_2^*\right) \quad \text{and} \quad \left(u_1^{(i)}\right)^w \neq u_2^{(i)},$$

then the corresponding value of $v^{(i)}$ computed by the simulator in line **D08** is determined by the following equation:

$$r_1^{(i)} x_1 + wr_2^{(i)} x_2 + \alpha^{(i)}\left(r_1^{(i)} y_1 + wr_2^{(i)} y_2\right) = \log_{g_1} v^{(i)} \bmod q, \tag{14}$$

where $u_1^{(i)} = g_1^{r_1^{(i)}}$ and $u_2^{(i)} = g_2^{r_2^{(i)}}$.

Notice that since $r_2 \neq r_1$ (line **I3**) and $r_1^{(i)} \neq r_2^{(i)}$ (since $(u_1^{(i)})^w \neq u_2^{(i)}$) and $\alpha^* \neq \alpha^{(i)}$; otherwise line **D08** is not executed), we have that the (11), (12), (13), and (14) are linearly independent, and thus the values c, d, v^*, and $v^{(i)}$ are uniformly and independently distributed in G. Thus the change effected in Game 4 to the definitions of these values does not change their distributions.

The change in line **D13** to the definition of $v^{(i)}$ clearly does not affect its distribution either, since it is just a rewriting of the original definition of $v^{(i)}$ by setting $x_1 + wx_2 = x \bmod q$ and $y_1 + wy_2 = y \bmod q$. □

Now our sequence of games reaches a fork in the road. Games 5 and 6 below (the "left fork") are used to show that $\Pr[X_4]$ is close to $1/2$. Then we define Game $5'$ (the "right fork"), which is another modification of Game 4, to show that $\Pr[F_4']$ is small.

Game 5

Game 5 is the same as Game 4, except that we change line **I5** as follows:

I5: $(k^*, K^*) \xleftarrow{R} \{0,1\}^{2n}$

That is, we simply generate the keys k^* and K^* at random.

Lemma 5. *There exists an efficient adversary A_3 such that* $|\Pr[X_4] - \Pr[X_5]| \leq \mathtt{Adv}_{A_3,\mathsf{KDF}}(n)$.

Proof. Observe that in Game 4, v^* is completely random and is not used anywhere, except once as an input to KDF. Thus the construction of adversary A_3 is immediate, and the lemma follows from the definition of secure KDF. □

By the assumption that KDF is a secure key derivation function we have that $\mathtt{Adv}_{A_3,\mathsf{KDF}}(n)$ is negligible.

Game 6

Game 6 is the same as Game 5, except that we change line **D06** as follows:

D06: return "reject"

Let F_6 be the event that line **D06** is ever executed in Game 5, in any decryption request. Clearly, Game 5 and Game 6 proceed identically until F_6 occurs; in particular, the events $X_5 \wedge \neg F_6$ and $X_6 \wedge \neg F_6$ are the same, and so

$$\left|\Pr[X_5] - \Pr[X_6]\right| \leq \Pr[F_6]. \tag{15}$$

Moreover, as we show in the following lemma, if F_6 occurs, the adversary has effectively broken the message authentication code keyed by k^* (which in Game 6 is truly random).

Lemma 6. *There exists an efficient adversary A_4 such that*

$$\Pr[F_6] \leq Q \cdot \mathtt{Succ}^{cma1}_{A_4,\mathsf{MAC}}(n). \tag{16}$$

Proof. We construct a forger A_4 that wins in the CMA1 game defined for message authentication codes. Remember that A' queries an oracle on a message e^*. In response the oracle selects a random key k^* and returns $t^* = \mathsf{MAC}_{k^*}(e^*)$.

The forger A_4 will run Game 6 as the simulator, except that it will not select a random key k^*. Instead, when it needs to process the encryption request, it computes e^* and queries the MAC oracle on it, getting back t^*. It will then include t^* in the target ciphertext.

Now with probability $\Pr[F_6]$ one of the queries $C^{(i)}$ processed in lines **D04–D06** will contain a valid tag under k^*, which constitutes a forgery. There are at most such Q queries, and the forger A_4 will select one pair $(e^{(i)}, t^{(i)})$ at random, hoping it is a forgery. Thus, its probability of success is $\frac{\Pr[F_6]}{Q}$.

By the definition of $\mathtt{Succ}^{A_4,cma1}_{\mathsf{MAC}}(n)$ the lemma follows. □

Lemma 7. *There exists an efficient adversary A_5 such that*

$$\left|\Pr[X_6] - 1/2\right| \leq \mathtt{Adv}^{cpa1}_{A_5,\mathsf{SKE}}(n). \tag{17}$$

Proof. Observe that the key K^* in Game 6 is truly random and is used for no other purpose than to encrypt m_b. Remember that $\Pr[X_6]$ is the probability of guessing the bit b determining which message between m_0 and m_1 is encrypted in e^* under K^*. Thus by the definition of $\mathtt{Adv}^{cpa}_{A_5,\mathsf{SKE}}(n)$ the lemma follows. □

By assumption, $\mathtt{Succ}^{cma1}_{A_4,\mathsf{MAC}}(n)$ and $\mathtt{Adv}^{cpa1}_{A_5,\mathsf{SKE}}(n)$ are negligible quantities.

Game 5′

We now backtrack to the fork in the road. Game 5′ is the same as Game 5, except that we change line **D09** as follows:

D09: $(k^{(i)}, K^{(i)}) \xleftarrow{R} \{0,1\}^{2n}$

Define $F'_{5'}$ to be the event that line **D11** is executed in the jth decryption request in Game 5′.

Lemma 8. *There exists efficient adversaries A_6 and A_7 such that*

$$\left|\Pr[F'_4] - \Pr[F'_{5'}]\right| \leq \mathtt{Adv}_{A_6,\mathsf{KDF}}(n) \tag{18}$$

and

$$\Pr[F'_{5'}] \leq \mathtt{Succ}^{cma1}_{A_7,\mathsf{MAC}}(n). \tag{19}$$

Proof. Recall that F'_4 is defined as the event that line **D11** is executed in Game 4.

Observe also that in Game 4, at line **D08**, the value $v^{(j)}$ is completely random and is not used anywhere, except once as an input to KDF. In Game 5 we are replacing the output of this invocation of KDF with random keys.

Thus any substantial difference between the probabilities $\Pr[F'_4]$ and $\Pr[F'_{5'}]$ can be used to construct adversary A_6 which breaks the security of the KDF function, and (18) follows.

Observe that in Game 5′, the key $k^{(j)}$ used in the message authentication code at line **D10** is completely random. Thus if line **D11** is executed, then we immediately have the construction of the adversary A_7 which outputs a forgery for the MAC over a random key (indeed we have this forgery without even obtaining a valid tag first). From this (19) follows. □

By assumption $\mathtt{Adv}_{A_6,\mathsf{KDF}}(n)$ and $\mathtt{Succ}^{A_7,cma1}_{\mathsf{MAC}}(n)$ are negligible.

Completing the Proof

We have

$$\begin{aligned}
\Pr[F_3] &\leq Q \Pr[F'_3] \quad \big[\text{by (8)}\big] \\
&= Q \Pr[F'_4] \quad \big[\text{by (10)}\big] \\
&\leq Q\big(\Pr[F'_{5'}] + \mathtt{Adv}_{A_6,\mathsf{KDF}}(n)\big) \quad \big[\text{by (18)}\big] \\
&\leq Q\big(\mathtt{Succ}^{cma1}_{A_7,\mathsf{MAC}}(n) + \mathtt{Adv}_{A_6,\mathsf{KDF}}(n)\big) \quad \big[\text{by (19)}\big].
\end{aligned}$$

Thus, we have

$$\Pr[F_3] \leq Q\big(\mathtt{Succ}^{cma1}_{A_7,\mathsf{MAC}}(n) + \mathtt{Adv}_{A_6,\mathsf{KDF}}(n)\big). \tag{20}$$

Finally, by combining Lemma 2, (6) with Lemma 3, (7) and (20), and Lemmas 4, 5, 6, and 7 we have

$$\begin{aligned}\big|\Pr[X_0] - 1/2\big| \leq\ & \mathtt{Adv}^{ddh}_{A_1,G}(n) + \mathtt{Succ}^{tcr}_{A_2,\mathcal{H}}(n) \\ & + \mathtt{Adv}^{cpa1}_{A_5,\mathsf{SKE}}(n) \\ & + \mathtt{Adv}_{A_3,\mathsf{KDF}}(n) + Q\,\mathtt{Succ}^{cma1}_{A_4,\mathsf{MAC}}(n) \\ & + Q\,\mathtt{Adv}_{A_6,\mathsf{KDF}}(n) + Q\,\mathtt{Succ}^{cma1}_{A_7,\mathsf{MAC}}(n) + 1/q. \end{aligned} \tag{21}$$

By assumption, the right-hand side of the above equation is negligible, which finishes the proof.

4.2. *Concrete Security*

Theorem 1 and its proof are stated in asymptotic terms, but the proof easily yields concrete bounds on the complexity of the reduction.

Assume that we have an adversary A that runs in time t, makes Q decryption queries, and breaks the chosen-ciphertext security of $\mathcal{KD}$ with advantage ϵ. Let t' be an upper bound on the time needed by the simulator to run any of the Games in the proof above (i.e., the time needed to set up the emulation in which the adversary A is run).

Then we can construct the following adversaries: A_{ddh}, A_{tcr}, A_{kdf}, A_{mac}, A_{ske}, which respectively break the DDH problem, the TCR function, the KDF function, the MAC, and the secret key encryption scheme with probabilities $\epsilon_{ddh}, \epsilon_{tcr}, \epsilon_{kdf}, \epsilon_{mac}, \epsilon_{ske}$.

These adversaries run the adversary A and the appropriate simulation (i.e., one of the adversaries $A_1, \ldots, A_7$ in the appropriate Game); thus their running time is at most $t + t'$. Notice that t' is very small (just generating the public keys and answering the decryption queries), so the running time of these adversaries is very close to the running time of A.

By (21) we have that

$$\epsilon < \epsilon_{ddh} + \epsilon_{tcr} + \epsilon_{ske} + (Q+1)\epsilon_{kdf} + 2Q\epsilon_{mac}.$$

Thus it must be that at least one of the following equations must be true:

$$\epsilon_{ddh}, \epsilon_{tcr}, \epsilon_{ske} > \frac{\epsilon}{5}, \qquad \epsilon_{kdf} > \frac{\epsilon}{5(Q+1)}, \qquad \epsilon_{mac} > \frac{\epsilon}{10Q}.$$

5. Modeling the KDF as a Random Oracle

In this section we present a slight modification of the previous scheme that can be proven secure assuming only the *Computational Diffie–Hellman Assumption*, instead of the stronger Decisional version. However the price to pay is to model the KDF as a Random Oracle.

In the Random Oracle model [2] we assume that an oracle implementing a random function is available to all parties. We use this oracle to implement a "complicated" hash function. We know that this is an unrealistic model of computations and that a proof of security in the random oracle model may not necessarily translate into a security proof in the real model (see [3]). However proving security in the random oracle model is a well-established methodology since it intuitively assures us of the security of the scheme, unless the hash function unexpectedly interferes with the other components.

Our encryption scheme is already proven secure if the DDH Assumption holds. Thus this additional proof will guarantee that even if the DDH is shown not to hold, we still have a proof of security under the weaker CDH. Our technique is based on the similar result by Shoup for the Cramer–Shoup encryption scheme [18].

The Modified Scheme The only modification in the scheme is the computation of the symmetric keys. We apply the function KDF not only to v but also to (u_1, u_2). So the description of the encryption and decryption procedures is the same except for the line computing k, K, which in this scheme is as follows:

$$(k, K) \leftarrow \mathsf{KDF}(u_1, u_2, v).$$

Theorem 2. *Assume that* (i) *the CDH Assumption holds over* G; (ii) $\mathsf{SKE} = (E, D)$ *is a one-time semantically secure symmetric encryption scheme*; (iii) MAC *is a one-time unforgeable MAC function*; (iv) $\mathcal{H}$ *is a TCR hash function family*; *and* (v) KDF *is modeled as a Random Oracle. Then the encryption scheme* $\mathcal{KD}$ *described above is secure against adaptive chosen ciphertext attack.*

Proof. The first thing to notice is that the proof of Theorem 1 applies to this scheme as well.[4]

Therefore a successful adversary for this scheme (i.e., one that wins the CCA2 game with nonnegligible probability) can be used to build a DDH distinguisher $\mathcal{D}'$ which decides the DH problem with nonnegligible probability over its input and internal coin tosses, i.e., $\mathtt{Adv}^{ddh}_{\mathcal{D}',G}(n)$ is nonnegligible.

We can then use the random self-reducibility of the DDH problem (see [17,18]—recall that we are using a prime order group G) to build a distinguisher $\mathcal{D}$ which (with overwhelming probability over its internal coin tosses) answers correctly for every input, i.e., $\mathcal{D}(g_1, g_2, u_1, u_2)$ outputs 1 if and only if $\log_{g_1} u_1 = \log_{g_2} u_2$.

Therefore, for the rest of the proof, we are going to assume that we have such a distinguisher $\mathcal{D}$. As before, we structure this proof as a sequence of games between a simulator (playing the role of the decryptor) and the adversary. In these games the function KDF is implemented as a random oracle, and the simulator provides simulated answers for queries to KDF. Game 0 as before is a regular CCA2 game.

[4] We note that for this modified scheme to remain secure under the assumptions of Theorem 1, the definition of security for the KDF function must be modified to assume that for any fixed (u_1, u_2), if v is random, then $\mathsf{KDF}(u_1, u_2, v)$ is pseudorandom.

Game 1

We now describe Game 1. In this Game the simulator is given as input a CDH target problem defined by a group G of prime order q and the values $g, h, u = g^r$ for $r \in_R \mathbb{Z}_q$. We assume that no polynomial-time algorithm can compute the value $z = h^r$ with nonnegligible probability.

First, the simulator sets up the public key as follows. The underlying group is set to G. Then the simulator chooses $w, w_1, w_2 \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and sets $g_1 = g$, $g_2 = g^w$, $c = g^{w_1}$, and $d = h^{w_2}$.

We first show how the simulator answers queries to the random oracle KDF. It maintains a list S which is initially empty and is used to store query/answer pairs for the random oracle. When the adversary queries KDF on (u_1, u_2, v), the simulator first checks if a pair of the form $[(u_1, u_2, v), (k, K)] \in S$ and if so, it returns (k, K). Otherwise it chooses $(k, K) \stackrel{R}{\leftarrow} \{0,1\}^{2n}$, adds $[(u_1, u_2, v), (k, K)]$ to S, and returns (k, K).

We now describe how the simulator answers the encryption query. When presented with messages m_0, m_1, the simulator selects $b \stackrel{R}{\leftarrow} \{0,1\}$ and "encrypts" m_b as follows. It chooses $(k^*, K^*) \stackrel{R}{\leftarrow} \{0,1\}^{2n}$ and computes $e^* \leftarrow E_{K^*}(m_b)$ and $t^* \leftarrow \mathsf{MAC}_{k^*}(e^*)$. The simulator also sets $u_1^* = u$, $u_2^* = u^w$, and $\alpha^* = H(u_1^*, u_2^*)$. The target ciphertext is (u_1^*, u_2^*, e^*, t^*). Note that the value v^* associated with this ciphertext should be

$$v^* = \left(cd^{\alpha^*}\right)^r = u^{w_1} h^{\alpha^* r \cdot w_2^{-1}},$$

which means that if the adversary ever queries the correct v^*, we can compute h^r and solve the CDH problem as follows:

$$h^r = \left(v^* u^{-w_1}\right)^{\frac{w_2}{\alpha^*}}.$$

Finally we show how the simulator handles decryption queries. When the adversary submits (u_1, u_2, e, t), the simulator performs the following:

D01: Set $\alpha = H(u_1, u_2)$
D02: If $u_2 \neq u_1^w$ return "reject"
D03: else if for all $[(u_1', u_2', v'), (k, K)] \in S$, we have $(u_1, u_2) \neq (u_1', u_2')$, return "reject"
D04: else if for all $[(u_1, u_2, v'), (k, K)] \in S$, we have $\mathcal{D}(g_1, cd^\alpha, u_1, v') = 0$, return "reject"
D05: else if there exists $[(u_1, u_2, v), (k, K)] \in S$, s.t. $\mathcal{D}(g_1, cd^\alpha, u_1, v) = 1$, then
D06: if $t \neq \mathsf{MAC}_k(e)$ return "reject"
D07: else return $D_K(e)$.

We are left to argue that this simulation is indistinguishable from the real CCA2 game, unless the adversary queries (u_1^*, u_2^*, v^*) to the random oracle, but, as we showed above, if the adversary does that, we can solve the CDH problem.

To argue that the simulation is indistinguishable from the real CCA2 game, we point out that:

- Line **D02** rejects "invalid" ciphertexts where $\log_{g_1} u_1 \neq \log_{g_2} u_2$. By an argument similar to the one in Lemma 4 we know that the value v associated with such

a ciphertext is uniformly distributed in G, and thus the adversary has negligible probability of finding it and thus querying it to the random oracle.

- Lines **D03** and **DO4** reject valid ciphertexts for which the adversary has not queried the correct v yet. Since KDF is modeled as a random oracle, the keys (k, K) are uniformly distributed, and thus the probability that the ciphertext is correct is negligible as well.

Thus the only possible difference between the real CCA2 game and Game 1 is detectable when the adversary queries v^* to the random oracle, but at that point we can stop the simulation and output h^r, contradicting the security of the CDH Assumption. □

6. Hash Proof Systems

In [5] Cramer and Shoup showed that their original scheme in [4] was a special instance of a generic paradigm based on *hash proof systems*. An advantage of this abstraction is that new CCA-secure schemes can be built based on different computational assumptions, such as Quadratic Residuosity and N-Residuosity mod N^2. In this section we show that the Kurosawa–Desmedt scheme can also be generalized to a scheme that uses hash proof systems.

Smooth Projective Hashing [5] Let X be a set, and $L \subset X$ a language. Let $\mathcal{H} = \{H_a\}$ a family of hash functions from X onto some set Y. Loosely speaking, the family $\mathcal{H}$ is projective if there exists a projection key that defines the action of H_a over the subset L of the domain X. That is, there exists a projection function $\theta(\cdot)$ that maps keys a into their projections $s = \theta(a)$. The projection key s is such that for every $x \in L$, it holds that the value of $H_a(x)$ is uniquely determined by s and x. In contrast, nothing is guaranteed for $x \notin L$, and it may not be possible to compute $H_a(x)$ from s and x. A smooth projective hash function has the additional property that for $x \notin L$, the projection key s actually says *nothing* about the value of $H_a(x)$. More specifically, given $x \notin L$ and $s = \theta(a)$, the value $H_a(x)$ follows a distribution which is statistically close to the uniform one over Y. Our scheme requires an additional property which we call *strong 2-universality*. Basically it is required that for $x \notin L$, even given $s = \theta(a)$ *and the value* $H_a(x')$ *for* $x' \notin L$ *and* $x' \neq x$, the distribution of the value $H_a(x)$ is statistically close to the uniform distribution over Y.

A smooth projective hash family is called a hash proof system if L is an NP-language and there are two efficient algorithms to compute H. The first computes $H_a(x)$ given $x \in X$ and the key a. The other computes $H_a(x)$ given the projection $s = \theta(a)$ and the values $x \in L$ together with a witness of the fact that $x \in L$. For our application to encryption, we are going to require that L is a hard on the average NP-language, in other words, that it is computationally infeasible to distinguish $x \in_R L$ from $x \in_R X \setminus L$.

We now formally define the properties of hash proof systems that we are going to require.

Subset Decision Problem this is defined by an *instance generator* algorithm $\mathcal{I}G$ that on input 1^n (where n is the security parameter) outputs a set X_n and a subset $L_n \subset X_n$, with the following properties:

- *Efficient Samplability:* There exist efficient algorithms that sample the uniform distribution in L_n and $X_n \setminus L_n$.
- *Computational Hardness:* For any PPT adversary A, define

$$\mathtt{Adv}_{A,\mathcal{IG}}(n) \stackrel{\triangle}{=} \left| \Pr_{x \leftarrow X_n \setminus L_n}\big(A(x)=1\big) - \Pr_{x \leftarrow L_n}\big(A(x)=1\big) \right|.$$

  We assume that for all efficient adversaries A, $\mathtt{Adv}_{A,\mathcal{IG}}(n)$ is negligible.
- *Existence of witnesses:* There exist a polynomial-time (in n) computable relation $\mathcal{R}$ such that

$$x \in L_n \quad \text{if and only if} \quad \exists w : \mathcal{R}(x,w)=1.$$

  The string w is called a witness for x. It is possible to efficiently sample a pair (x,w) according to the uniform distribution.

Subset Decision Problems with Trapdoor Our proof requires the subset decision problem to have a "master" trapdoor that allows us to decide it efficiently. While this property was not required by the proof of the original Cramer–Shoup scheme, all their instances of subset decision problems have such a trapdoor (see paragraph after the proof of Theorem 3).

We say that a subset decision problem has a trapdoor if there exists an alternative instance generator algorithm $\mathcal{IG}'$ that on input 1^n outputs X_n, L_n, T_n. The distribution of X_n, L_n output by $\mathcal{IG}'$ is indistinguishable from X_n, L_n output by $\mathcal{IG}$, and thus in particular the three properties stated above are satisfied.

Moreover we assume that there exists a PPT machine D that on input X_n, L_n, T_n, and $x \in X_n$, outputs $D(X_n, L_n, T_n, x) = 1$ if $x \in L_n$ and $D(X_n, L_n, T_n, x) = 0$ if $x \in X_n \setminus L_n$.

Hash Proof Systems For every set X_n, there exists a family of hash functions $\mathcal{H} = \{H_a\}_{a \in \Gamma}$, where Γ is a set of keys, and $H_a : X_n \to Y_n$. There exists also a projection function $\theta : \Gamma \to \Delta$. The functions must satisfy the following properties:

- *Efficient Samplability:* There exists an efficient algorithm that samples the uniform distribution over Γ.
- *Efficient Computation via Key:* There exists an efficient algorithm that on input $x \in X_n$ and a, computes $H_a(x)$.
- *Efficient Computation via Projection and Witness:* There exists an efficient algorithm that on input $x \in L_n$, a witness w for x, and the projection $s = \theta(a)$, computes $H_a(x)$.
- *Strong 2-universality:* for every $x, x' \in X_n \setminus L_n$, consider the following experiment. Choose $a \leftarrow \Gamma$ uniformly at random and compute $s = \theta(a)$ and $H_a(x') = y'$. We require that the distribution of $H_a(x)$, given the events $s = \theta(a)$ and $H_a(x') = y'$, is statistically indistinguishable from the uniform distribution over Y.

The Modified Scheme Using the techniques from [5], the Kurosawa and Desmedt scheme described in the previous section can be generalized using hash proof systems over subset decision problems as follows (we drop the security parameter index n for

clarity). The new scheme $\mathcal{K}D'$ is defined as follows. The sets X, L are generated using $\mathcal{I}G$ and can be shared by all parties. Each receiver chooses its secret key as $a \leftarrow \Gamma$, and the projection $s = \theta(a)$ will be its public key. The key derivation function is defined over Y.

To encrypt m, the sender chooses an element $x \leftarrow L$ together with its witness w. He then computes $v = H_a(x)$ using the projection s and the witness. Then the keys $(k, K) = \mathsf{KDF}(v)$ are derived as before. The rest of the encryption procedure remains the same, i.e., $e = E_K(m)$ and $t = MAC_k(e)$. The ciphertext is x, e, t.

The receiver on input (x, e, t) computes $v' = H_a(x)$ using the secret key a. It then computes $(k, K) = \mathsf{KDF}(v')$. If $t = MAC_k(e)$, then it decrypts $m = D_K(e)$.

Theorem 3. *Assume that* (i) X, L *is a subset decision problem with trapdoor and that* $\mathcal{H}$ *is a strongly* 2*-universal hash proof system over it*; (ii) $\mathsf{SKE} = (E, D)$ *is a one-time semantically secure symmetric encryption scheme*; (iii) MAC *is a one-time unforgeable MAC function*; *and* (iv) KDF *is a secure key derivation function. Then the encryption scheme* $\mathcal{K}D'$ *described above is secure against adaptive chosen ciphertext attack.*

Proof. The proof follows the spirit of the proof of Theorem 1 but with some important differences. This proof is also structured as a sequence of games.

Game 0

As in the proof of Theorem 1, *Game 0* is basically the CCA2 game where the simulator provides the adversary's environment. The simulator runs $\mathcal{I}G$ for the subset decision problem to obtain X, L (we omit the security parameter n). Then the simulator chooses the secret key $a \in \Gamma$ and publishes $s = \theta(a)$ as the public key.

During the execution of the game, the adversary A makes a number of decryption requests: $C^{(1)}, \ldots, C^{(Q)}$, where $C^{(i)} = [x^{(i)}, e^{(i)}, t^{(i)}]$ which the simulator decrypts using the secret key a and gives the result to the adversary. We denote by $v^{(i)}$, $k^{(i)}$, and $K^{(i)}$ the corresponding intermediate quantities computed by the decryption algorithm on input $C^{(i)}$.

The adversary A may also make a single "challenge request." For such a request, the adversary submits two messages m_0, m_1, which are bit strings of equal length, to the simulator; the simulator chooses $b \in \{0, 1\}$ at random and encrypts m_b, obtaining the "target ciphertext" $C^* = (x^*, e^*, t^*)$. The simulator gives C^* to the adversary. We denote by v^*, k^*, and K^* the corresponding intermediate quantities computed by the encryption algorithm.

The only restriction on the adversary's requests is that after it makes a challenge request, subsequent decryption requests must not be the same as the target ciphertext.

At the end of the game, the adversary A outputs $\tilde{b} \in \{0, 1\}$. Let X_0 be the event that $\tilde{b} = b$. Since Game 0 is identical to the CCA2 game, we have that

$$\mathtt{Adv}^{cca}_{A,\mathcal{K}D'}(n) = \left|\Pr[X_0] - 1/2\right|, \tag{22}$$

and our goal is to prove that this quantity is negligible.

As in the proof of Theorem 1, we consider a sequence of games. In each game, there will be well-defined bits $\tilde{b}$ and b, so that in Game i, we always define X_i to the event

that $\tilde{b} = b$ in that game. All of these games should be viewed as operating on the same underlying probability space.

Game 1

In this proof, the simulator in Game 1 chooses x^* at random in $X \setminus L$. Then v^* is computed by the simulator as $H_a(x^*)$ (recall that the simulator knows a).

As in the proof of Theorem 1, the adversary's probability of success in Game 1 can increase with respect to Game 0, by at most a negligible quantity. Indeed we can construct an adversary A_1 such that the increase in the probability of success is bounded by $\mathtt{Adv}_{A_1,\mathcal{IG}}(n)$, which by assumption is negligible.

This is proven by an argument analogous to Lemma 2 translated to the setting of subset decision problems.

Game 2

Game 2 in the proof of Theorem 1 is not necessary in this proof, since the (target) collision-resistant hash function is "incorporated" in the strong 2-universality of $H_a(\cdot)$. To maintain the correspondence with the proof of Theorem 1, here we assume that Game 2 is identical to Game 1.

Notice that at this point we have Rejection Rule 0 defined as the rejection rule of the ordinary CCA2 game (i.e., the MAC tag does not match). We do not have a Rejection Rule 1. Again to maintain the correspondence with the proof of Theorem 1, we assume that Rejection Rule 1 is empty.

Game 3

In Game 3 the simulator runs $\mathcal{IG}'$ to generate X, L together with the trapdoor T (the value w is the master trapdoor for the DDH decision problem used in Theorem 1). The simulator did not need to make explicit use of the trapdoor in previous games. Indeed, we could not have used the hardness of the subset decision problem if the simulator knew the trapdoor. However, we are now finished with that hardness assumption, so the simulator is free to make use of T in this and subsequent games.

The simulator uses the trapdoor T to detect adversary-generated ciphertexts where $x^{(i)} \in X \setminus L$ and reject them. We call this *Rejection Rule 2* as in the proof of Theorem 1. Similarly we define event F_3: a ciphertext is rejected during Game 3 using Rejection Rule 2 to which Rejection Rules 0 and 1 are not applicable. We also choose $j \in_R [1..Q]$ and define F_3' to be the event that in Game 3, Rejection Rules 0 and 1 do not apply to $C^{(j)}$, but Rejection Rule 2 does apply to $C^{(j)}$.

As in the previous proof, Game 2 and Game 3 proceed identically until F_3 occurs and

$$\left|\Pr[X_2] - \Pr[X_3]\right| \leq \Pr[F_3] \leq Q \Pr[F_3']. \tag{23}$$

As in the previous proof, we postpone showing that $\Pr[F_3']$ is negligible until later.

Let us write down in detail how Game 3 works, starting from scratch:

- The simulator begins by generating the environment for the simulation:

  I1: $X, L, T \leftarrow \mathcal{I}G'$
  I2: $a \xleftarrow{R} \Gamma \quad s \leftarrow \theta(a)$
  I3: $x^* \xleftarrow{R} X \setminus L$
  I4: $v^* \leftarrow H_a(x^*)$
  I5: $(k^*, K^*) \leftarrow \mathsf{KDF}(v^*)$
  I6: $j \xleftarrow{R} \{1, \ldots, Q\}$

  The values X, L, and s are given to the adversary.
- In processing a decryption request $C^{(i)} = [x^{(i)}, e^{(i)}, t^{(i)}]$, the simulator proceeds as follows. We start labeling decryption instructions from line D04 in order to maintain the correspondence between the instructions of the simulator in this proof, with the ones of the simulator in the proof of Theorem 1 (lines **D01–03** there dealt with the (target) collision-resistant function, which is not an issue in this proof).

  D04: if $x^{(i)} = x^*$ then
  D05: if $t^{(i)} \neq \mathsf{MAC}_{k^*}(e^{(i)})$ then return "reject"
  D06: return $D_{K^*}(e^{(i)})$
  D07: else if $x^{(i)} \in X \setminus L$ (decided using T) then
  D08: $v^{(i)} \leftarrow H_a(x^{(i)})$
  D09: $(k^{(i)}, K^{(i)}) \leftarrow \mathsf{KDF}(v^{(i)})$
  D10: if $t^{(i)} \neq \mathsf{MAC}_{k^{(i)}}(e^{(i)})$ then return "reject"
  D11: return "reject"
  D12: else
  D13: $v^{(i)} \leftarrow H_a(x^{(i)})$
  D14: $(k^{(i)}, K^{(i)}) \leftarrow \mathsf{KDF}(v^{(i)})$
  D15: if $t^{(i)} \neq \mathsf{MAC}_{k^{(i)}}(e^{(i)})$ then return "reject"
  D16: return $D_{K^{(i)}}(e^{(i)})$

  Note that Rejection Rule 0 is applied at lines **D05**, **D10**, and **D15**, and Rejection Rule 2 (and no other Rejection Rule) at line **D11**.
- In processing the challenge request, the adversary gives m_0, m_1 to the simulator. The simulator computes

$$b \xleftarrow{R} \{0, 1\}, \qquad e^* \leftarrow E_{K^*}(m_b), \qquad t^* \leftarrow \mathsf{MAC}_{k^*}\left(e^*\right),$$

  and gives $C^* := (x^*, e^*, t^*)$ to the adversary.

We have written the logic of the decryption oracle in this particular way to facilitate further analysis. Note that the computations at **D08–D10** have no real effect, other than to determine if the event F_3' occurs; indeed, once line **D08** is reached, the ciphertext is sure to be rejected, either at line **D10** or at line **D11**. The event F_3' is simply the event that line **D11** executes in the jth decryption request.

Game 4

Game 4 is the same as Game 3, except that we change lines **I4** and **D08** as follows:

I4: $v^* \stackrel{R}{\leftarrow} Y$

D08: $v^{(i)} \stackrel{R}{\leftarrow} Y$

We define F_4' to be the event that line **D11** is executed in the jth decryption request in Game 4.

Lemma 9. *Assuming that $H_a(\cdot)$ is a strongly 2-universal projective hash function, we have that*

$$\Pr[X_3] = \Pr[X_4] \tag{24}$$

and

$$\Pr[F_3'] = \Pr[F_4']. \tag{25}$$

Proof. This lemma is analogous to Lemma 4 in the proof of Theorem 1. The difference is that here we prove the claim using the generic property of strong 2-universality of the projective hash function, while in the proof of Lemma 4 we had a linear algebra argument to prove the statement from scratch (and implicitly proving that the projective hash function used there to compute v was strongly 2-universal).

First we prove that changing line **D08** makes no difference. Notice that line **D08** is invoked when $x^{(i)} \in X \setminus L$. By the property of strong 2-universality the value $v^{(i)}$ computed as $H_a(x^{(i)})$ in Game 3 is uniformly distributed in Y *even when given* $s = \theta(a)$ *and the value* $v^* = H_a(x^*)$ *for another* $x^* \in X \setminus L$. Thus to compute $v^{(i)} \stackrel{R}{\leftarrow} Y$ in Game 4 creates the same view for the adversary.

By the same argument changing line **I4** in Game 4 does not affect the adversary's view (since v^* is also computed in Game 3 as $H_a(x^*)$ for $x^* \in X \setminus L$). □

Now the proof continues exactly as the proof of Theorem 1, using Games 5, 6, and 5′ to work on the "symmetric" components of the scheme (the KDF, MAC, and encryption functions). The final bound on the adversary's advantage is

$$\begin{aligned} \left|\Pr[X_0] - 1/2\right| \leq\ & \mathtt{Adv}_{A_1,\mathcal{IG}}(n) + \mathtt{Adv}^{cpa1}_{A_5,\mathsf{SKE}}(n) \\ & + \mathtt{Adv}_{A_3,\mathsf{KDF}}(n) + Q\mathtt{Succ}^{cma1}_{A_4,\mathsf{MAC}}(n) \\ & + Q\mathtt{Adv}_{A_6,\mathsf{KDF}}(n) + Q\mathtt{Succ}^{cma1}_{A_7,\mathsf{MAC}}(n), \end{aligned} \tag{26}$$

where the adversaries $A_1, \ldots, A_7$ are analogous to the same-numbered adversaries in the proof of Theorem 1 (again notice that there is no adversary A_2 breaking the target collision-resistant hash in this scheme). □

Examples of Subset Decision Problems with Master Trapdoor Cramer and Shoup [5] proposed three Subset Decision Problems that could be used to build projective hash functions.

- *DDH-Based.* Here the instance generator on input n sets X_n to be a group G_n of prime order $q_n > 2^n$. The subset L_n is defined by two random generators g_1, g_2 of

G_n as

$$L_n = \left\{ \left(g_1, g_2, g_1^r, g_2^r \right) \mid r \in \mathbb{Z}_{q_n} \right\}.$$

Clearly under the DDH Assumption this is a Subset Decision Problem as defined above. The master trapdoor here is the discrete log of g_2 w.r.t. g_1.

- *QR-Based.* Here the instance generator on input n sets X_n to be J_N the subgroup of elements of Jacobi symbol 1 in Z_N^*, where N is the product of two safe prime numbers $N = p_n q_n$ with $p_n, q_n > 2^n$. We recall that since p_n, q_n are safe primes, we have that $p_n = 2p'_n + 1$ and $q_n = 2q'_n + 1$ with p'_n, q'_n primes. The subset L_n is defined as the subgroup of Quadratic Residues in Z_N^*. Clearly if we assume that deciding Quadratic Residuosity is hard, then this is a Subset Decision Problem as defined above. The master trapdoor here is the factorization of N.
- *N-Residuosity-Based.* In this case the instance generator on input n sets $X_n = Z_{N^2}^*$ as above. The subset L_n is defined as the subgroup of N-Residues in $Z_{N^2}^*$. Clearly if we assume that deciding N-Residuosity is hard in $Z_{N^2}^*$, then this is a Subset Decision Problem as defined above. The master trapdoor here is the factorization of N as well.

Removing the Master Trapdoor We note that the original proof by Kurosawa and Desmedt [12] (the one that uses information-theoretic KDF and MAC functions) can be used to remove the requirement that the subset decision problem has a master trapdoor.

Theorem 4. *Assume that* (i) X, L *is a subset decision problem and that* $\mathcal{H}$ *is a strongly* 2*-universal hash proof system over it*; (ii) $\mathsf{SKE} = (E, D)$ *is a one-time semantically secure symmetric encryption scheme*; (iii) MAC *is an information-theoretically secure MAC function*; *and* (iv) KDF *is an information-theoretically secure key derivation function. Then the encryption scheme* $\mathcal{K}D'$ *described above is secure against adaptive chosen ciphertext attack.*

Proof. Consider the proof of Theorem 3. Game 0 and Game 1 are identical to those in that proof, and Game 2 is also set to be identical to Game 1.

In this proof there is no Game 3, since the subset decision problem does not admit a master trapdoor. So we assume Game 3 to be identical to Game 1. The instructions for the simulator are as follows (again lines are labeled in order to maintain a correspondence between the two proofs).

- The simulator begins by generating the environment for the simulation:

  I1: $X, L \leftarrow \mathcal{I}G$
  I2: $a \xleftarrow{R} \Gamma \quad s \leftarrow \theta(a)$
  I3: $x^* \xleftarrow{R} X \setminus L$
  I4: $v^* \leftarrow H_a(x^*)$
  I5: $(k^*, K^*) \leftarrow \mathsf{KDF}(v^*)$
  I6: $j \xleftarrow{R} \{1, \ldots, Q\}$

  The values X, L, and s are given to the adversary.

- In processing a decryption request $C^{(i)} = [x^{(i)}, e^{(i)}, t^{(i)}]$, the simulator proceeds as follows.

D04: if $x^{(i)} = x^*$ then
D05: if $t^{(i)} \neq \mathsf{MAC}_{k^*}(e^{(i)})$ then return "reject"
D06: return $D_{K^*}(e^{(i)})$
D12: else
D13: $v^{(i)} \leftarrow H_a(x^{(i)})$
D14: $(k^{(i)}, K^{(i)}) \leftarrow \mathsf{KDF}(v^{(i)})$
D15: if $t^{(i)} \neq \mathsf{MAC}_{k^{(i)}}(e^{(i)})$ then return "reject"
D16: return $D_{K^{(i)}}(e^{(i)})$

  Note that Rejection Rule 0 is applied at lines **D05** and **D15**. There are no other Rejection Rules.
- In processing the challenge request, the adversary gives m_0, m_1 to the simulator. The simulator computes

$$b \xleftarrow{R} \{0,1\}, \qquad e^* \leftarrow E_{K^*}(m_b), \qquad t^* \leftarrow \mathsf{MAC}_{k^*}\big(e^*\big),$$

  and gives $C^* := (x^*, e^*, t^*)$ to the adversary.

We now define Game 4 by modifying the way v^* is computed. We change line **I4** to set v^* as a random element in Y. We need to prove that the adversary's view does not substantially change between Games 3 and 4. In the previous proof we argued this point by rejecting invalid ciphertexts (i.e., ciphertexts where $x_i \in X \setminus L$) in Game 3. In turn, this allowed us to claim that the adversary never gained any information about the value $v^{(i)}$ associated with an invalid ciphertext. Since $H_a(\cdot)$ is strongly 2-universal, the value of $H_a(x^*)$ is then uniformly distributed.

However, in this proof the simulator cannot detect invalid ciphertexts, so we need to argue that the adversary does not obtain any information about a $v^{(i)}$ associated to an invalid ciphertext, in a different way. We make the following claim

Lemma 10. *In both Games* 3 *and* 4, *if the adversary submits* $C^{(i)} = [x^{(i)}, e^{(i)}, t^{(i)}]$ *with* $x^{(i)} \in X \setminus L$, *then line* **D16** *is executed with negligible probability.*

Proof. Because $H_a(\cdot)$ is strongly 2-universal, the value $v^{(i)} = H_a(x^{(i)})$ for $x^{(i)} \in X \setminus L$ is uniformly distributed even given $s = \theta(a)$ and $v^* = H_a(x^*)$ (we point out that the adversary does not see v^* but sees a MAC and an ecryption computed with keys derived by v^*—in any case, given the adversary view, $v^{(i)}$ is uniformly distributed).

Since the KDF and MAC functions are information-theoretically secure, we have that the value $\mathsf{MAC}_{k^{(i)}}(e^{(i)})$ is also uniformly distributed, given the view of the adversary. Recall that line **D16** is executed if the adversary submits $t^{(i)} = \mathsf{MAC}_{k^{(i)}}(e^{(i)})$, but the probability that the adversary finds the correct $t^{(i)}$ is thus negligible. □

Now we can argue that the change made in Game 4 makes at most a negligible difference in the adversary's success probability. From now on the proof continues unchanged with Games 5 and 6. Game 5′ is not needed in this proof since the above lemma bounds

immediately the increase in the adversary's advantage (while in the previous proof we had postponed computing this bound to Game 5′). □

Appendix: The Original Scheme

The Cramer–Shoup hybrid encryption scheme proposed in [4], and refined in [18], uses the same tools as the one described above. However key generation, encryption, and decryption algorithms are different.

Key Generation The description of the group G is generated, along with a random generator g_1 for G. Any keys for KDF and H are also generated. Then:

$$w, x, y, z \xleftarrow{R} \mathbb{Z}_q, \qquad g_2 \leftarrow g_1^w, \qquad c \leftarrow g_1^x, \qquad d \leftarrow g_1^y, \qquad h \leftarrow g_1^z.$$

The public key consists of the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with the group elements c, d, h. The private key consists of the public key, along with w, x, y, z.

Encryption of $m \in \{0,1\}^*$

$$\begin{array}{l} r \xleftarrow{R} \mathbb{Z}_q,\ \kappa \leftarrow h^r,\ u_1 \leftarrow g_1^r \in G,\ u_2 \leftarrow g_2^r \in G,\ \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q \\ v \leftarrow c^r d^{r\alpha} \in G,\ (k, K) \leftarrow \mathsf{KDF}(\kappa),\ e \leftarrow E_K(m),\ t \leftarrow \mathsf{MAC}_k(e) \\ \text{output } C := (u_1, u_2, v, e, t) \end{array}$$

Decryption of $C = (u_1, u_2, v, e, t)$

$$\begin{array}{l} \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q,\ v' \leftarrow u_1^{x+y\alpha} \in G,\ \kappa' \leftarrow u_1^z,\ (k, K) \leftarrow \mathsf{KDF}(\kappa') \\ \text{if } t \neq \mathsf{MAC}_k(e) \text{ or } v' \neq v \text{ or } u_2 \neq u_1^w \text{ then} \\ \quad \textit{reject} \\ \text{else} \\ \quad m \leftarrow D_K(e) \\ \quad \text{output } m \end{array}$$

Notice that compared to the Kurosawa–Desmedt scheme, the encryption algorithm in this scheme computes an extra exponentiation (the computation of κ) and a longer ciphertext (it includes the group element v). However, that does not translate into a direct gain in efficiency.

In the Cramer–Shoup scheme we can choose the prime q to be 160-bit long. This results in a random value κ which is computationally indistinguishable from a random group element. Then, under a suitable computational assumption on KDF, we can derive keys k, K of any required length using a pseudo-random number generator.

On the other hand, the key k in the Kurosawa–Desmedt scheme must be derived from v in an information-theoretic way. We cannot apply a pseudo-random number generator, otherwise we lose the information-theoretic security. For common security, parameters k is required to be at least 160-bit long. The only way we know how to do this is to map v into a 160-bit string using universal hashing and the Entropy Smoothing Theorem.

But this requires v to come from a distribution with min-entropy at least, say, 320. Considering that from κ we also need to derive the key K (say, other 128 bits), then it seems that the group G must have order q of at least about 450 bits. This increase in the security parameter clearly offsets the gain obtained by dropping one exponentiation.

Using our proof, however, we can claim that the Kurosawa–Desmedt scheme can be used with a group G of order q, where q is a 160-bit prime.

We also note that the scheme in [18] is optimized so that all exponentiations in the decryption algorithm are with respect to the same base—this allows for speedups using techniques for exponentiation with preprocessing. A similar optimization can be applied to the Kurosawa–Desmedt scheme.

References

[1] M. Abe, R. Gennaro, K. Kurosawa, Tag-KEM/DEM: a new framework for hybrid encryption. *J. Cryptol.* **21**(1), 97–130 (2008)

[2] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in *1993 ACM Conference on Computer and Communications Security* (ACM, New York, 1993), pp. 62–73

[3] R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited. *J. ACM* **51**(4), 557–594 (2004)

[4] R. Cramer, V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, in *CRYPTO'98*. Springer LNCS, vol. 1462 (Springer, Berlin, 1998), pp. 13–25

[5] R. Cramer, V. Shoup, Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption, in *EuroCrypt'02*. Springer LNCS, vol. 2332 (Springer, Berlin, 2002), pp. 45–64

[6] R. Cramer, V. Shoup, Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**, 167–226 (2003)

[7] W. Diffie, M. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **IT-22**(6), 644–654 (1976)

[8] D. Dolev, C. Dwork, M. Naor, Non-malleable cryptography, in *STOC'91* (1991), pp. 542–552

[9] R. Gennaro, V. Shoup, A Note on An Encryption Scheme of Kurosawa and Desmedt. IACR Eprint Archive http://eprint.iacr.org/2004/194

[10] J. Herranz, D. Hofheinz, E. Kiltz, The Kurosawa–Desmedt Key Encapsulation is not Chosen-Ciphertext Secure. IACR Eprint Archive http://eprint.iacr.org/2006/207

[11] D. Hofheinz, E. Kiltz, Secure hybrid encryption from weakened key encapsulation, in *CRYPTO 2007*. Springer LNCS, vol. 4622 (Springer, Berlin, 2007), pp. 553–571

[12] K. Kurosawa, Y. Desmedt, A New Paradigm of Hybrid Encryption Scheme, in *CRYPTO'04*. Springer LNCS, vol. 3152 (Springer, Berlin, 2004), pp. 426–442

[13] M. Naor, M. Yung, Public-key cryptosystems provably secure against chosen ciphertext attacks, in *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, STOC'90* (ACM, New York, 1990), pp. 427–437

[14] M. Naor, M. Yung, Universal one-way hash functions and their cryptographic applications, in *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, STOC'89* (ACM, New York, 1989), pp. 33–43

[15] C. Rackoff, D. Simon, Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack, in *CRYPTO'91*. Springer LNCS, vol. 576 (Springer, Berlin, 1991), pp. 433–444

[16] J. Rompel, One-way functions are necessary and sufficient for secure signatures, in *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, STOC'90* (ACM, New York, 1990), pp. 387–394

[17] V. Shoup, Lower bounds for discrete logarithms and related problems, in *EuroCrypt'97*. Springer LNCS, vol. 1233 (Springer, Berlin, 1997), pp. 256–266

[18] V. Shoup, Using hash functions as a hedge against chosen ciphertext attack, in *EuroCrypt'00*. Springer LNCS, vol. 1807 (Springer, Berlin, 2000), pp. 275–288