Hybrid Damgård Is CCA1-Secure under the DDH Assumption

Yvo Desmedt[1], Helger Lipmaa[2], and Duong Hieu Phan[3]

[1] University College London, UK
[2] Cybernetica AS, Estonia
[3] University of Paris 8, France

Abstract. In 1991, Damgård proposed a simple public-key cryptosystem that he proved CCA1-secure under the Diffie-Hellman Knowledge assumption. Only in 2006, Gjøsteen proved its CCA1-security under a more standard but still new and strong assumption. The known CCA2-secure public-key cryptosystems are considerably more complicated. We propose a hybrid variant of Damgård's public-key cryptosystem and show that it is CCA1-secure if the used symmetric cryptosystem is CPA-secure, the used MAC is unforgeable, the used key-derivation function is secure, and the underlying group is a DDH group. The new cryptosystem is the most efficient known CCA1-secure hybrid cryptosystem based on standard assumptions.

Keywords: CCA1-security, Damgård's cryptosystem, DDH, hybrid cryptosystems.

1 Introduction

CCA2-security in the standard model is currently the strongest widely accepted security requirement for public-key cryptosystems. The first practical CCA2-secure cryptosystem was proposed by Cramer and Shoup [CS98]. In their scheme, the plaintext is a group element. However, in practice one really needs a hybrid cryptosystem where the plaintext can be an arbitrarily long bitstring. The first related hybrid cryptosystem was proposed by Shoup in [Sho00]. In [KD04], Kurosawa and Desmedt proposed another hybrid cryptosystem that, taking account the comments of Gennaro and Shoup [GS04], is up to now the most efficient published hybrid CCA2-secure cryptosystem that is based on the Decisional Diffie-Hellman (DDH) assumption.

Existing CPA-secure cryptosystems like Elgamal [Elg85] are considerably simpler. CPA-security is however a very weak security notion. In this paper we concentrate on an intermediate security notion, CCA1-security (or "non-adaptive CCA-security"). Recall that already in 1991, Damgård [Dam91] proposed a simple CCA1-secure cryptosystem, although with the security proof relying on the non-standard Diffie-Hellman Knowledge assumption [Dam91, BP04]. In 2006, Gjøsteen proved that a generalization of Damgård's cryptosystem is CCA1-secure under a strong conventional assumption [Gjø06]. Recently, Lipmaa [Lip08] gave a considerably simpler proof of Gjøsteen's result.

M.K. Franklin, L.C.K. Hui, and D.S. Wong (Eds.): CANS 2008, LNCS 5339, pp. 18–30, 2008.

Table 1. Comparison between a few discrete-logarithm based hybrid cryptosystems. Here, x is the bit length of group element representations and $|m|$ is the length of symmetrically encrypted plaintext. In encryption/decryption, e means one exponentiation, s — one symmetric-key IND-CCA secure encryption/decryption of $|m|$-bit string (this may also consist of an IND-CPA secure encryption/decryption together with a MAC on the ciphertext), t — one computation of a target collusion-resistant hash function, u — one computation of a universal one-way hash function. Non-cryptographic operations, e.g., of key-derivation functions, are not included to the computation cost. If the assumption is not well-established, a link to the paper(s) defining the assumption is given.

Name	Security	Assumption	Encrypt.	Decrypt.	|Ciphertext|	|pk|
Hybrid						
This paper	CCA1	DDH	$3e + s$	$2e + s$	$2x + \|m\| + \|\mathsf{t}\|$	x
[HK07, Sect. 4.2]	CCA2	DDH	$4e + t + s$	$2e + t + s$	$2x + \|m\| + \|\mathsf{t}\|$	$3x$+hash
[KD04, GS04]	CCA2	DDH	$4e + t + s$	$2e + t + s$	$2x + \|m\| + \|\mathsf{t}\|$	$2x$+hash
[ABR01]	CCA2	[ABR01]	$2e + s$	$1e + s$	$x + \|m\| + \|\mathsf{t}\|$	x
[Sho00]	CCA2	DDH	$5e + s$	$3e + s$	$3x + \|m\| + \|\mathsf{t}\|$	$4x$+hash
Non-hybrid						
[CS04]	CCA2	DDH	$5e + u$	$3e + u$	$4x$	$5x$+hash
Lite [CS04]	CCA1	DDH	$4e$	$3e$	$4x$	$4x$
[Dam91]	CCA1	[Gjø06, Lip08]	$3e$	$2e$	$3x$	$2x$
[Elg85]	CPA	DDH	$2e$	e	$2x$	x

We propose a Damgård-based hybrid cryptosystem that we call "Hybrid Damgård". This scheme can also be seen as a simplification of the Kurosawa-Desmedt cryptosystem [KD04]. We prove that Hybrid Damgård is CCA1-secure if the used symmetric cryptosystem is semantically secure, the used MAC is unforgeable, the used key-derivation function is secure, and the underlying group is a DDH group. Hybrid Damgård is currently the most efficient CCA1-secure hybrid cryptosystem that is based on the DDH assumption. It is essentially as efficient as Damgård's original CCA1-secure cryptosystem, requiring the encrypter and the decrypter to additionally evaluate only some secret-key or non-cryptographic operations. See Tbl. 1 for a comparison. In addition, Hybrid Damgård is a hashless cryptosystem.

In the security proof, we use a standard game hopping technique, similar to the one in [KD04, GS04]. Also our proof is only slightly more complex than that given by Gjøsteen, the additional complexity is only due to use of additional symmetric primitives.

Recent Work. Essentially the same cryptosystem was very recently discussed in [DP08] and [KPSY08]. In [DP08], the authors proved CCA2-security of the Hybrid Damgård cryptosystem under a strong knowledge assumption (corresponding to KA3 of [BP04]). One can extract a CCA1-security proof from it under a somewhat weaker knowledge assumption (corresponding to KA2 of [BP04]). In a yet unpublished eprint [KPSY08], the authors proved that the Hybrid Damgård is CCA2-secure under the DDH assumption; however, the used hash function and symmetric cryptosystem have to satisfy stronger assumptions. They also briefly mention that it is CCA1-secure under the same assumptions we use.

Notation. For a set A, let $U(A)$ denote the uniform distribution on it.

2 Preliminaries

Let $|B| < |A|$. A function $\mathsf{kdf} : A \to B$ is *key derivation function*, KDF, if the distributions $\mathsf{kdf}(U(A))$ and $U(B)$ are computationally indistinguishable. If $|A| < |B|$, then KDF is a pseudorandom generator. Otherwise, KDF may be a non-cryptographic function.

Decisional Diffie-Hellman Assumption

Definition 1. *Let $\mathbb{G}$ be a group of order q with a generator g. A DDH distinguisher $\mathcal{A}\mathsf{lice}$ has success $\mathsf{AdvDDH}_{\mathbb{G},g}(\mathcal{A}\mathsf{lice})$, defined as*

$$\left| \begin{array}{l} \Pr[x, y \leftarrow \mathbb{Z}_q : A(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1] - \\ \Pr[x, y \leftarrow \mathbb{Z}_q, z \leftarrow \mathbb{Z}_q \setminus \{xy\} : A(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] \end{array} \right|$$

in attacking DDH group $\mathbb{G}$, where the probability is taken over the choice of random variables and over the random coin tosses of $\mathcal{A}\mathsf{lice}$. We say that $\mathbb{G}$ is a (τ, ε)-DDH group if $\mathsf{AdvDDH}_{\mathbb{G},g}(\mathcal{A}\mathsf{lice}) \leq \varepsilon$ for any τ-time adversary $\mathcal{A}\mathsf{lice}$ and for any generator g.

Usually, one takes $z \leftarrow \mathbb{Z}_q$. The difference between $\mathcal{A}\mathsf{lice}$'s success in these two variants of the DDH game is clearly upper bounded by $1/q$, see e.g. [CS04, Lem. 1]. We later use a variation where also x is fixed (i.e., g^x is a subindex of AdvDDH), but this variation is equally powerful because of the random self-reducibility of DDH. Moreover, because of the random self-reducibility of DDH, the choice of g is not important.

We say that (g_1, g_2, g_3, g_4) is a *DDH tuple* if $(g_3, g_4) = (g_1, g_2)^r$ for some $r \in \mathbb{Z}_q$.

Public-Key Cryptosystems. Let $\mathsf{pub} = (\mathsf{pub.gen}, \mathsf{pub.enc}, \mathsf{pub.dec})$ be a public-key cryptosystem for a fixed security parameter λ. In particular, $\mathsf{pub.gen}(1^\lambda)$ returns a new secret/public key pair $(\mathsf{sk}, \mathsf{pk})$, $\mathsf{pub.enc}(\mathsf{pk}; m; r)$ encrypts the message m by using randomizer r, and $\mathsf{pub.dec}(\mathsf{sk}; C)$ decrypts a ciphertext C such that $\mathsf{pub.dec}(\mathsf{sk}; \mathsf{pub.enc}(\mathsf{pk}; m; \cdot)) = m$; the result of $\mathsf{pub.dec}$ may be a special symbol $\perp$.

Consider the next *CCA2 game* between the adversary $\mathcal{A}\mathsf{lice}$ and the challenger:

Setup. The challenger runs $\mathsf{pub.gen}(1^\lambda)$ to obtain a random instance of a secret and public key pair $(\mathsf{sk}, \mathsf{pk})$. It gives the public key pk to $\mathcal{A}\mathsf{lice}$.

Query phase 1. $\mathcal{A}\mathsf{lice}$ adaptively issues decryption queries C. The challenger responds with $\mathsf{pub.dec}(\mathsf{sk}; C)$.

Challenge phase. $\mathcal{A}\mathsf{lice}$ outputs two (equal length) messages $\hat{m}_0$, $\hat{m}_1$. The challenger picks a random $b_{\mathcal{A}\mathsf{lice}} \leftarrow \{0, 1\}$ and sets $\hat{C} \leftarrow \mathsf{pub.enc}(\mathsf{pk}; \hat{m}_{b_{\mathcal{A}\mathsf{lice}}}, \hat{r})$ for random $\hat{r}$. It gives $\hat{C}$ to $\mathcal{A}\mathsf{lice}$.

Query phase 2. $\mathcal{A}\mathsf{lice}$ continues to issue decryption queries C as in phase 1, but with the added constraint that $C \neq \hat{C}$. The challenger responds each time with $\mathsf{pub.dec}(\mathsf{sk}; C)$.

Guess. $\mathcal{A}\text{lice}$ outputs her guess $b'_{\mathcal{A}\text{lice}} \in \{0,1\}$ for $b_{\mathcal{A}\text{lice}}$ and wins the game if $b_{\mathcal{A}\text{lice}} = b'_{\mathcal{A}\text{lice}}$.

Definition 2 (CPA/CCA1/CCA2 Security of Public-Key Cryptosystems). *A CCA2 adversary* $\mathcal{A}\text{lice}$ *has success* $\mathsf{AdvCCA2}_{\mathsf{pub}}(\mathcal{A}\text{lice}) := |2\Pr[b_{\mathcal{A}\text{lice}} = b'_{\mathcal{A}\text{lice}}] - 1|$ *in attacking* pub*, where the probability is taken over the choice of* $b_{\mathcal{A}\text{lice}}$ *and over the random coin tosses of* $\mathcal{A}\text{lice}$*. We say that* pub *is* $(\tau, \gamma_1, \gamma_2, \mu, \varepsilon)$-CCA2-secure *if* $\mathsf{AdvCCA2}_{\mathsf{pub}}(\mathcal{A}\text{lice}) \leq \varepsilon$ *for any* τ*-time adversary* $\mathcal{A}\text{lice}$ *that makes up to* γ_i *queries in phase* $i \in \{1,2\}$*, with the total queried message length being up to* μ *bits.* pub *is* $(\tau, \gamma, \mu, \varepsilon)$-CCA1-secure *if it is* $(\tau, \gamma, 0, \mu, \varepsilon)$*-CCA2-secure.* pub *is* (τ, ε)-CPA-secure *if it is* $(\tau, 0, 0, 0, \varepsilon)$*-CCA2-secure. The values* $\mathsf{AdvCPA}^{\mathsf{pub}}$ *and* $\mathsf{AdvCCA1}^{\mathsf{pub}}$ *are defined accordingly.*

Damgård Cryptosystem [Dam91]

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and its randomly chosen generator $g \in \mathbb{G}$.
Key setup pub.gen**:** Generate $(\alpha, \beta) \leftarrow \mathbb{Z}_q^2$. Set $\mathsf{sk} \leftarrow (\alpha, \beta)$ and $\mathsf{pk} \leftarrow (c \leftarrow g^\alpha, d \leftarrow g^\beta)$.
Encryption pub.enc**:** Given a message $m \in \mathbb{G}$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u_1 \leftarrow g^r$, $u_2 \leftarrow c^r$, $e \leftarrow m \cdot d^r$. The ciphertext is (u_1, u_2, e).
Decryption pub.dec**:** Given a ciphertext (u_1, u_2, e), do the following. If $u_2 \neq u_1^\alpha$ then output $m \leftarrow \perp$. Otherwise, compute $m \leftarrow e/u_1^\beta$ and return m.

Descriptions of some other known public-key cryptosystems are given in Appendix.

Symmetric Cryptosystems. Let $\mathsf{sym} = (\mathsf{sym.gen}, \mathsf{sym.enc}, \mathsf{sym.dec})$ be a symmetric cryptosystem for a fixed security parameter λ. In particular, $\mathsf{sym.gen}(1^\lambda)$ returns a new secret key sk, $\mathsf{sym.enc}(\mathsf{sk}; m; r)$ encrypts the message m by using randomizer r, and $\mathsf{sym.dec}(\mathsf{sk}; C)$ decrypts a ciphertext C such that $\mathsf{sym.dec}(\mathsf{sk}; \mathsf{sym.enc}(\mathsf{sk}; m; r)) = m$.

CPA/CCA1/CCA2-security of symmetric cryptosystems is defined similarly as in the case of public-key cryptosystems. Consider the next *CCA2 game* between the adversary $\mathcal{A}\text{lice}$ and the challenger:

Setup. The challenger runs $\mathsf{pub.gen}(1^\lambda)$ to obtain a random instance of a secret key sk
Query phase 1. $\mathcal{A}\text{lice}$ adaptively issues encryption queries m, where the challenger responds with $\mathsf{sym.enc}(\mathsf{sk}; m, r)$ for random r, and decryption queries C, where the challenger responds with $\mathsf{sym.dec}(\mathsf{sk}; C)$.
Challenge phase. $\mathcal{A}\text{lice}$ outputs two (equal length) messages $\hat{m}_0$, $\hat{m}_1$. The challenger picks a random $b_{\mathcal{A}\text{lice}} \leftarrow \{0,1\}$ and sets $\hat{C} \leftarrow \mathsf{sym.enc}(\mathsf{sk}; \hat{m}_{b_{\mathcal{A}\text{lice}}}, \hat{r})$ for random $\hat{r}$. It gives $\hat{C}$ to $\mathcal{A}\text{lice}$.
Query phase 2. $\mathcal{A}\text{lice}$ continues to issue encryption queries m and decryption queries C as in phase 1, but with the added constraint that $C \neq \hat{C}$. The challenger as in phase 1.
Guess. $\mathcal{A}\text{lice}$ outputs her guess $b'_{\mathcal{A}\text{lice}} \in \{0,1\}$ for $b_{\mathcal{A}\text{lice}}$ and wins the game if $b_{\mathcal{A}\text{lice}} = b'_{\mathcal{A}\text{lice}}$.

Definition 3 (CPA/CCA1/CCA2 Security of Symmetric Cryptosystems). *A CCA2 adversary* $\mathcal{A}\mathsf{lice}$ *has success* $\mathsf{AdvCCA2}_{\mathsf{sym}}(\mathcal{A}\mathsf{lice}) := |2\Pr[b_{\mathcal{A}\mathsf{lice}} = b'_{\mathcal{A}\mathsf{lice}}] - 1|$ *in attacking* sym*, where the probability is taken over the choice of* $b_{\mathcal{A}\mathsf{lice}}$ *and over the random coin tosses of* $\mathcal{A}\mathsf{lice}$*. We say that* pub *is* $(\tau, \gamma_1, \gamma_2, \mu, \varepsilon)$-CCA2-secure *if* $\mathsf{AdvCCA2}_{\mathsf{pub}}(\mathcal{A}\mathsf{lice}) \le \varepsilon$ *for any* τ*-time adversary* $\mathcal{A}\mathsf{lice}$ *that makes up to* γ_i *queries in phase* $i \in \{1, 2\}$*, with the total queried message length being up to* μ *bits.* sym *is* $(\tau, \gamma, \mu, \varepsilon)$-CCA1-secure *if it is* $(\tau, \gamma, 0, \mu, \varepsilon)$*-CCA2-secure.* sym *is* (τ, ε)-CPA-secure *if it is* $(\tau, 0, 0, 0, \varepsilon)$*-CCA2-secure. The values* $\mathsf{AdvCPA}^{\mathsf{sym}}$ *and* $\mathsf{AdvCCA1}^{\mathsf{sym}}$ *are defined accordingly.*

MAC. A MAC $\mathsf{mac} = (\mathsf{mac.tag}, \mathsf{mac.ver})$, on key κ and message e produces a tag $\mathsf{t} = \mathsf{mac.tag}(\kappa; e)$. A MAC is *unforgeable* if for random κ, after obtaining $\mathsf{t}' \leftarrow \mathsf{mac.tag}(\kappa; e')$ for (at most one) adversarially chosen e', it is hard to compute a forgery, i.e., a pair (e, t) such that $e \neq e'$ but $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \top$.

A standard way of constructing a CCA2-secure symmetric cryptosystem is to encrypt a message m by using a CPA-secure cryptosystem, $e \leftarrow \mathsf{sym.enc}(K; m, r)$ and then returning e together with a tag $\mathsf{t} \leftarrow \mathsf{mac.tag}(\kappa; e)$. Here, (K, κ) is a pair of independent random keys.

3 Hybrid Damgård Cryptosystem

We now propose a new cryptosystem, *Hybrid Damgård*, an hybrid variant of the Damgård cryptosystem that uses some ideas from the Kurosawa-Desmedt cryptosystem as exposed by [GS04].

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and its two randomly chosen different generators $g_1, g_2 \in \mathbb{G}$. Choose a CPA-secure symmetric cryptosystem $\mathsf{sym} = (\mathsf{sym.gen}, \mathsf{sym.enc}, \mathsf{sym.dec})$, an unforgeable MAC $\mathsf{mac} = (\mathsf{mac.tag}, \mathsf{mac.ver})$, and a KDF kdf from $\mathbb{G}$ to the set of keys of $(\mathsf{sym}, \mathsf{mac})$.

Key setup $\mathsf{pub.gen}$**:** Generate $(\alpha_1, \alpha_2) \leftarrow \mathbb{Z}_q^2$. Set $\mathsf{sk} \leftarrow (\alpha_1, \alpha_2)$ and $\mathsf{pk} \leftarrow (c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2})$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \{0, 1\}^*$, do the following. First, generate $r \leftarrow \mathbb{Z}_q$, and randomizer ρ for sym, and then

$$u_1 \leftarrow g_1^r \ , \quad u_2 \leftarrow g_2^r \ , \quad (K, \kappa) \leftarrow \mathsf{kdf}(c^r) \ ,$$
$$e \leftarrow \mathsf{sym.enc}(K; m, \rho) \ , \quad \mathsf{t} \leftarrow \mathsf{mac.tag}(\kappa; e) \ .$$

The ciphertext is $(u_1, u_2, e, \mathsf{t})$.

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext $(u_1, u_2, e, \mathsf{t})$, do the following. Compute $(K, \kappa) \leftarrow \mathsf{kdf}(u_1^{\alpha_1} u_2^{\alpha_2})$. If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \perp$ then return $m \leftarrow \perp$ else return $m \leftarrow \mathsf{sym.dec}(K; e)$.

Theorem 1. *Fix a group* $\mathbb{G}$*, a symmetric cryptosystem* $\mathsf{sym} = (\mathsf{sym.gen}, \mathsf{sym.enc}, \mathsf{sym.dec})$*, a MAC* $\mathsf{mac} = (\mathsf{mac.tag}, \mathsf{mac.ver})$*, and a hash function* kdf *from* $\mathbb{G}$ *to the set of keys for* $(\mathsf{sym}, \mathsf{mac})$*. Then the Hybrid Damgård cryptosystem* pub *is CCA1-secure if (1) the DDH assumption holds, (2)* kdf *is a KDF, (3)* sym *is CPA-secure, and (4)* mac *is unforgeable.*

Proof. Use the next sequence of game hops. Assume that $\mathcal{A}$lice is a $(\tau, \gamma, \mu, \varepsilon)$ CCA1-adversary for pub. In every game $\underline{\textbf{Game}}_i$ we modify the CCA1 game so that there $\mathcal{A}$lice has advantage $\Pr[X_i]$, where for every i, $|\Pr[X_{i+1}] - \Pr[X_i]|$ is negligible. Moreover, $|\Pr[X_{i+1}]-\Pr[X_i]|$ is estimated by defining an event F_{i+1} such that events $X_i \wedge \neg F_{i+1}$ iff $X_{i+1} \wedge \neg F_{i+1}$. Then clearly $|\Pr[X_{i+1}] - \Pr[X_i]| \leq \Pr[F_{i+1}]$ [CS98]. The full proof is slightly more complicated since the games build up a tree instead of a chain. All games are fairly standard. Details follow.

$\underline{\textbf{Game}_0}$

This is the original CCA1 game. $\mathcal{A}$lice gets a random public key $\mathsf{pk} = (c)$, makes a number of decryption queries $(u_1, u_2, e, \mathsf{t})$, receives a challenge ciphertext $(\hat{u}_1, \hat{u}_2, \hat{e}, \widehat{\mathsf{t}})$, makes some more decryption queries $(u_1, u_2, e, \mathsf{t})$, and then makes a guess. In this game, $\mathcal{A}$lice has success $\Pr[X_0] = \varepsilon$. To simplify further analysis, we assume that the challenger has created the values $(\hat{u}_1, \hat{u}_2, \hat{K}, \hat{\kappa})$ before the phase-1 queries.

$\underline{\textbf{Game}_1}$

Here we redefine the internal way of computing the key during the decryption queries and the challenge ciphertext creation. Namely, we let $(K, \kappa) \leftarrow \mathsf{kdf}(u_1^{\alpha_1} u_2^{\alpha_2})$. This does not change the ciphertexts, and thus also in $\underline{\textbf{Game}}_1$, $\mathcal{A}$lice has success $\Pr[X_1] = \varepsilon$.

$\underline{\textbf{Game}_2}$

In this game, the challenge ciphertext is created by choosing $(\hat{u}_1, \hat{u}_2) \leftarrow (g_1^{\hat{r}_1}, g_2^{\hat{r}_2})$ for random $\hat{r}_1 \neq \hat{r}_2$. Assume that in $\underline{\textbf{Game}}_2$, $\mathcal{A}$lice has success probability $\Pr[X_2]$. We now construct a DDH adversary $\mathcal{B}$ob with advantage related to $|\Pr[X_1]-\Pr[X_2]|$. $\mathcal{B}$ob gets (g_1, q, g_2) as an input, where g_1 generates a group $\mathbb{G}$ of order q and a $g_2 \leftarrow \mathbb{G}\backslash\{g_1\}$. $\mathcal{B}$ob and $\mathcal{A}$lice choose appropriate (sym, mac, kdf). He then runs $\mathcal{A}$lice step-by-step.

- $\mathcal{B}$ob asks for his challenge $(\hat{u}_1, \hat{u}_2) \in \mathbb{G}^2$. He generates random $\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_q$, sets $\mathsf{sk} \leftarrow (\alpha_1, \alpha_2)$ and $\mathsf{pk} \leftarrow (c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2})$. He sends pk to $\mathcal{A}$lice.
- When $\mathcal{A}$lice makes a phase-1 decryption query with a purported ciphertext $(u_1, u_2, e, \mathsf{t})$, $\mathcal{B}$ob returns m according to the decryption formula: $(K, \kappa) \leftarrow \mathsf{kdf}(u_1^{\alpha_1} u_2^{\alpha_2})$. If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \perp$ then $m \leftarrow \perp$ else $m \leftarrow \mathsf{sym.dec}(K; e)$.
- When $\mathcal{A}$lice submits her message pair (m_0, m_1), $\mathcal{B}$ob sets $b_{\mathcal{A}\mathsf{lice}} \leftarrow \{0, 1\}$, and sends $(\hat{u}_1, \hat{u}_2, \hat{e}, \widehat{\mathsf{t}})$ as the challenge ciphertext to $\mathcal{A}$lice, where $\hat{e} \leftarrow \mathsf{sym.enc}(\hat{K}; m_{b_{\mathcal{A}\mathsf{lice}}}, \hat{\rho})$, for uniform randomizer $\hat{\rho}$, and $\widehat{\mathsf{t}} \leftarrow \mathsf{mac.tag}(\hat{\kappa}; \hat{e})$ for $(\hat{K}, \hat{\kappa}) \leftarrow \mathsf{kdf}(\hat{u}_1^{\alpha_1} \hat{u}_2^{\alpha_2})$.
- Finally, $\mathcal{A}$lice replies with a guess $b'_{\mathcal{A}\mathsf{lice}}$. $\mathcal{B}$ob outputs $b'_{\mathcal{B}\mathsf{ob}} \leftarrow 1$ if $b'_{\mathcal{A}\mathsf{lice}} = b_{\mathcal{A}\mathsf{lice}}$, and $b'_{\mathcal{B}\mathsf{ob}} \leftarrow 2$ otherwise.

Let $b_{\mathcal{B}\mathsf{ob}} = 1$ if $(g_1, g_2, \hat{u}_1, \hat{u}_2)$ is a random DDH tuple, and $b_{\mathcal{B}\mathsf{ob}} = 2$ if it is a random non-DDH tuple, and assume that $\Pr[b_{\mathcal{B}\mathsf{ob}} = 1] = 1/2$. In particular if $b_{\mathcal{B}\mathsf{ob}} = 2$ then $\hat{u}_1 \leftarrow g_1^{\hat{r}_1}$, $\hat{u}_2 \leftarrow g_2^{\hat{r}_2}$ for random $\hat{r}_1 \neq \hat{r}_2$.

If $b_{\mathcal{B}\mathsf{ob}} = 1$ then all steps are emulated perfectly for $\underline{\textbf{Game}}_1$. Thus, $\Pr[b'_{\mathcal{A}\mathsf{lice}} = b_{\mathcal{A}\mathsf{lice}} | b_{\mathcal{B}\mathsf{ob}} = 1] = \Pr[X_1]$. If $b_{\mathcal{B}\mathsf{ob}} = 2$ then all steps are emulated perfectly for $\underline{\textbf{Game}}_2$ and thus $\Pr[b'_{\mathcal{A}\mathsf{lice}} = b_{\mathcal{A}\mathsf{lice}} | b_{\mathcal{B}\mathsf{ob}} = 2] = \Pr[X_2]$.

Thus, $\Pr[b'_{\mathcal{B}\mathsf{ob}} = b_{\mathcal{B}\mathsf{ob}}] = \frac{1}{2}\Pr[b'_{\mathcal{B}\mathsf{ob}} = 1|b_{\mathcal{B}\mathsf{ob}} = 1] + \frac{1}{2}\Pr[b'_{\mathcal{B}\mathsf{ob}} = 2|b_{\mathcal{B}\mathsf{ob}} = 2] = \frac{1}{2}\Pr[b'_{\mathcal{A}\mathsf{lice}} = b_{\mathcal{A}\mathsf{lice}}|b_{\mathcal{B}\mathsf{ob}} = 1] + \frac{1}{2} - \frac{1}{2}\Pr[b'_{\mathcal{A}\mathsf{lice}} = b_{\mathcal{A}\mathsf{lice}}|b_{\mathcal{B}\mathsf{ob}} = 2] = \frac{1}{2} + \frac{1}{2}(\Pr[X_1] - \Pr[X_2])$, and $|\Pr[X_1] - \Pr[X_2]| = |2\Pr[b'_{\mathcal{B}\mathsf{ob}} = b_{\mathcal{B}\mathsf{ob}}] - 1|$ is the advantage of $\mathcal{B}$ob distinguishing random DDH tuples and random non-DDH tuples of form $\{(g_1, g_2, \hat{u}_1, \hat{u}_2) : (g_1, g_2, \hat{u}_1) \leftarrow \mathbb{G}^3, \hat{u}_2 \leftarrow \mathbb{G} \setminus \{\hat{u}_1\}\}$. Thus,

$$|\Pr[X_1] - \Pr[X_2]| \le \varepsilon_{\mathsf{ddh}} \ ,$$

where $\varepsilon_{\mathsf{ddh}}$ is the probability of breaking the DDH assumption, given resources comparable to the resources of the adversary.

$\underline{\mathbf{Game}_3}$

First, recall that $(\hat{u}_1, \hat{u}_2)$ is computed before the phase-1. Now, we let the decryption oracle to reject all ciphertexts (u_1, u_2) such that $(u_1, u_2) \ne (\hat{u}_1, \hat{u}_2)$ and (g_1, g_2, u_1, u_2) is not a DDH tuple. Here, F_3 is the event that such a ciphertext would have been accepted in $\underline{\mathbf{Game}}_2$. Clearly, $\Pr[F_3] \le \gamma_1 \cdot \Pr[F'_3]$, where F'_3 is the event that such a ciphertext would have been accepted in a randomly chosen phase-1 query of $\underline{\mathbf{Game}}_2$, and γ_1 is again the number of queries in phase-1. We defer the computation of $\Pr[F'_3]$ to later games where it is substantially easier to do.

Complete description of $\underline{\mathbf{Game}}_3$ is given in Fig. 1 (here we can explicitly use the value of w since we are done with a DDH reduction that had to compute w; the upcoming DDH reduction in $\underline{\mathbf{Game}}_4$ computes something different). It also points out differences between $\underline{\mathbf{Game}}_3$ and $\underline{\mathbf{Game}}_4$.

$\underline{\mathbf{Game}_4}$

In this game we change six lines as specified in Fig. 1. Let $\mathcal{A}$lice be an adversary in $\underline{\mathbf{Game}}_4$ again. Because of the change on line **D05**, other changes are only decorative and do not change $\mathcal{A}$lice's view. Thus, let F'_4 be the event that during a randomly chosen phase-1 query of $\underline{\mathbf{Game}}_3$, the line **D08** is executed.

Consider a concrete phase-1 decryption query. Then

$$\log_{g_1} c = \alpha_1 + w\alpha_2 \ , \tag{1}$$

$$\log_{g_1} v = r_1\alpha_1 + r_2 w\alpha_2 \ . \tag{2}$$

Equations (1) and (2) are linearly independent and thus v can take on any value from $\mathbb{G}$, and thus is uniformly distributed over $\mathbb{G}$. Thus,

$$\Pr[F'_4] = \Pr[F'_3] \ .$$

Now we do a fork in the hopping. Games $\underline{\mathbf{Game}}_5$ and $\underline{\mathbf{Game}}_6$ bound $\Pr[X_4]$. Game $\underline{\mathbf{Game}}'_5$ bounds $\Pr[F'_4]$.

$\underline{\mathbf{Game}_5}$

$\underline{\mathbf{Game}}_5$ is the same as $\underline{\mathbf{Game}}_4$, except that here we compute $(\hat{K}, \hat{\kappa}) \leftarrow$ "random keys". Because in $\underline{\mathbf{Game}}_4$, $\hat{v}$ is completely random, and is not used anywhere, except once as an input to kdf, then it is easy to see that

$$|\Pr[X_5] - \Pr[X_4]| \le \varepsilon_{\mathsf{kdf}} \ ,$$

Setup. Fix $\mathbb{G}$, q, two random different generators $g_1, g_2 \in \mathbb{G}$ where $g_2 = g_1^w$ for a random $w \leftarrow \mathbb{Z}_q \setminus \{1\}$, sym, mac and kdf. The challenger does the following.

S01 $\underline{\alpha_1, \alpha_2 \leftarrow \mathbb{Z}_q}$ $\quad \utilde{\alpha \leftarrow \mathbb{Z}_q}$
S02 $\underline{\mathsf{sk} \leftarrow (\alpha_1, \alpha_2)}$ $\quad \utilde{\mathsf{sk} \leftarrow \alpha}$
S03 $\underline{\mathsf{pk} \leftarrow (c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2})}$ $\quad \utilde{\mathsf{pk} \leftarrow (c \leftarrow g_1^{\alpha})}$
S04 Send the public key pk to $\mathcal{A}\mathsf{lice}$
S05 $\hat{r}_1 \leftarrow \mathbb{Z}_q, \hat{r}_2 \leftarrow \mathbb{Z}_q \setminus \{\hat{r}_1\}$
S06 $\hat{u}_1 \leftarrow g_1^{\hat{r}_1}, \hat{u}_2 \leftarrow g_2^{\hat{r}_2}$
S07 $\underline{\hat{v} \leftarrow \hat{u}_1^{\alpha_1} \hat{u}_2^{\alpha_2}}$ $\quad \utilde{\hat{v} \leftarrow \mathbb{G}}$
S08 $(\hat{K}, \hat{\kappa}) \leftarrow \mathsf{kdf}(\hat{v})$

Query phase 1. $\mathcal{A}\mathsf{lice}$ adaptively issues decryption queries $(u_1, u_2, e, \mathsf{t})$. The challenger does the following.

D01 If $(u_1, u_2) = (\hat{u}_1, \hat{u}_2)$ then
D02   If $\mathsf{mac.ver}(\hat{\kappa}; e, \mathsf{t}) = \bot$ then return $\bot$
D03   Return $\mathsf{sym.dec}(\hat{K}; e)$
D04 else if $u_1^w \neq u_2$ then
D05 $\underline{v \leftarrow u_1^{\alpha_1} u_2^{\alpha_2}}$ $\quad \utilde{v \leftarrow \mathbb{G}}$
D06 $(K, \kappa) \leftarrow \mathsf{kdf}(v)$
D07   If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \bot$ then return $\bot$
D08   Return $\bot$. `// Event` F_3`: Difference between` $\underline{\mathbf{Game}}_2$/$\underline{\mathbf{Game}}_3$
D09 else
D10 $\underline{v \leftarrow u_1^{\alpha_1} u_2^{\alpha_2}}$ $\quad \utilde{v \leftarrow u_1^{\alpha}}$
D11 $(K, \kappa) \leftarrow \mathsf{kdf}(v)$
D12   If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \bot$ then return $\bot$
D13   Return $\mathsf{sym.dec}(K; e)$

Challenge phase. $\mathcal{A}\mathsf{lice}$ outputs two (equal length) messages $\hat{m}_0$, $\hat{m}_1$. The challenger picks a random $b_{\mathcal{A}\mathsf{lice}} \leftarrow \{0, 1\}$. The challenger sets $\hat{e} \leftarrow \mathsf{sym.enc}(K; \hat{m}_{b_{\mathcal{A}\mathsf{lice}}}, \hat{\rho})$, for uniform randomizer $\hat{\rho}$, and $\widehat{\mathsf{t}} \leftarrow \mathsf{mac.tag}(\hat{\kappa}; \hat{e})$. It gives $\hat{C} \leftarrow (\hat{u}_1, \hat{u}_2, \hat{e}, \widehat{\mathsf{t}})$ to $\mathcal{A}\mathsf{lice}$.

Guess. $\mathcal{A}\mathsf{lice}$ outputs its guess $b'_{\mathcal{A}\mathsf{lice}} \in \{0, 1\}$ for $b_{\mathcal{A}\mathsf{lice}}$ and wins the game if $b_{\mathcal{A}\mathsf{lice}} = b'_{\mathcal{A}\mathsf{lice}}$.

Fig. 1. Games $\underline{\mathbf{Game}}_3$ and $\underline{\mathbf{Game}}_4$. Two games differ only in a few lines. In those lines, the part that is only executed in $\underline{\mathbf{Game}}_3$ has been underlined, while the part that is only executed in $\underline{\mathbf{Game}}_4$ has been underwaved.

where $\varepsilon_{\mathsf{kdf}}$ is the probability of distinguishing the output of kdf from completely random keys, using resources similar to the resources of the given adversary.

$\underline{\mathbf{Game}}_6$

$\underline{\mathbf{Game}}_6$ is the same as $\underline{\mathbf{Game}}_5$, except that we change the line **D03** to "return $\bot$". Let F_6 be the event that line **D03** is ever executed in $\underline{\mathbf{Game}}_6$ in any decryption request. If F_6 occurs then $\mathcal{A}\mathsf{lice}$ has broken the MAC keyed by $\hat{\kappa}$ (which in $\underline{\mathbf{Game}}_6$ is truly random). Thus, $\Pr[F_6] \leq \gamma \varepsilon_{\mathsf{mac}}$, where $\varepsilon_{\mathsf{mac}}$ is the advantage with which one can break the MAC using resources similar to those of $\mathcal{A}\mathsf{lice}$. Then, clearly,

$$|\Pr[X_6] - \Pr[X_5]| \leq \Pr[F_6] \leq \gamma \varepsilon_{\mathsf{mac}} \ .$$

Observe that $\hat{K}$ is completely random and thus used for no other purpose than to encrypt $m_{b_{\mathcal{A}\mathsf{lice}}}$. It is thus easy to see that

$$|\Pr[X_6] - 1/2| \leq \varepsilon_{\mathsf{enc}} \ ,$$

where $\varepsilon_{\mathsf{enc}}$ is the probability of breaking the semantic security of sym, using resources comparable to the resources of the adversary.

$\underline{\mathbf{Game}}_{5'}$

$\underline{\mathbf{Game}}_{5'}$ is the same as $\underline{\mathbf{Game}}_4$, except that we change the line **D06** to $(K, \kappa) \leftarrow$ "random keys". Let $F'_{5'}$ be the event that line **D08** is executed in a randomly chosen decryption query of phase-1 in $\underline{\mathbf{Game}}_{5'}$. Because in $\underline{\mathbf{Game}}_{5'}$, in line **D05**, the value of v is completely random and not used anywhere, except once as an input to kdf, then it is easy to see that

$$|\Pr[F'_{5'}] - \Pr[F'_4]| \leq \varepsilon'_{\mathsf{kdf}} \ ,$$

where $\varepsilon'_{\mathsf{kdf}}$ is the advantage with which one can distinguish the output of kdf from a random key pair, using resources similar to those of the given adversary.

Now, in $\underline{\mathbf{Game}}_{5'}$, the key κ used in line **D07** is completely random. From this, it easily follows that

$$\Pr[F'_{5'}] \leq \varepsilon'_{\mathsf{mac}} \ ,$$

where $\varepsilon'_{\mathsf{kdf}}$ is the probability of breaking mac, using resources similar to those of the given adversary.

Completing The Proof

We have

$$\Pr[F_3] \leq \gamma_1 \Pr[F'_3] = \gamma_1 \Pr[F'_4] \leq \gamma_1 (\Pr[F'_{5'}] + \varepsilon'_{\mathsf{kdf}}) \leq \gamma_1 (\varepsilon'_{\mathsf{mac}} + \varepsilon'_{\mathsf{kdf}}) \ .$$

Finally,

$$|\Pr[X_0] - 1/2| \leq \varepsilon_{\mathsf{ddh}} + \varepsilon_{\mathsf{kdf}} + \varepsilon_{\mathsf{enc}} + \gamma_1 (\varepsilon_{\mathsf{mac}} + \varepsilon'_{\mathsf{mac}} + \varepsilon'_{\mathsf{kdf}}) \ . \qquad (3)$$

□

4 Why We Cannot Prove CCA2-Security

We will now briefly show why this proof technique cannot show that Hybrid Damgård is CCA2-secure in the standard model and "standard" assumptions from KDF, MAC and secret-key cryptosystem. Consider any phase-2 decryption query in $\underline{\mathbf{Game}}_4$. Let $\hat{v} := \hat{u}_1^{\alpha_1} \hat{u}_2^{\alpha_2}$. Then from $\mathcal{A}$lice's point of view, during a query of phase-2, (α_1, α_2) is a random point satisfying two linearly independent equations, Eq. (1) and the equation

$$\log_{g_1} \hat{v} = \hat{r}_1 \alpha_1 + \hat{r}_2 w \alpha_2 \ . \qquad (4)$$

During an arbitrary query of phase-2, suppose that $\mathcal{A}$lice queries an invalid ciphertext $(u_1, u_2, e, \mathsf{t})$ to the decryption oracle where $u_1 = g_1^{r_1}$ and $u_2 = g_2^{r_2}$ with $r_1 \neq r_2$. Thus also Eq. (2) holds. Now, Eq. (1), (2) and (4) are *not* linearly independent and thus we cannot claim as in the previous papers that the value v is uniform and random.

More precisely, to distinguish v from random, $\mathcal{A}$lice participates in the next game. She first sees tuple

$$(g_1, g_2, c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2}; \hat{u}_1 \leftarrow g_1^{\hat{r}_1}, \hat{u}_2 \leftarrow g_2^{\hat{r}_2}, \hat{v} \leftarrow g_1^{\hat{r}_1 \alpha_1} g_2^{\hat{r}_2 \alpha_2})$$

for randomly chosen α_1, α_2, $\hat{r}_1 \neq \hat{r}_2$. Second, she sends to challenger a tuple

$$u_1 \leftarrow g_1^{r_1}, u_2 \leftarrow g_2^{r_2} \quad ,$$

for $r_1 \neq r_2$. Third, she gets back a value v such that either $v = u_1^{\alpha_1} u_2^{\alpha_2} = g_1^{r_1 \alpha_1} g_2^{r_2 \alpha_2}$ (if $b_{\mathcal{A}\mathsf{lice}} = 1$), or $v \leftarrow \mathbb{G}$ (if $b_{\mathcal{A}\mathsf{lice}} = 0$).

Clearly, we can assume that $\mathcal{A}$lice knows the values r_1, r_2. Note that her task is equivalent to deciding whether $v/c^{r_1} = g_2^{(r_2 - r_1)\alpha_2} = u_2^{r_2 - r_1}$ or whether $v = c^{r_1} u_2^{r_2 - r_1}$, which she can do trivially. Therefore, v is not pseudorandom.

Recently, [KPSY08] have given a CCA2-security proof of the Hybrid Damgård under a stronger assumption on the hash function.

Acknowledgments. Part of this work was done while the second and the third author were working at University College London. Yvo Desmedt is the BT Chair of Information Security and funded by EPSRC EP/C538285/1. Helger Lipmaa was supported by Estonian Science Foundation, grant #6848, European Union through the European Regional Development Fund and the 6th Framework Programme project AEOLUS (FP6-IST-15964).

References

[ABR01] Abdalla, M., Bellare, M., Rogaway, P.: The Oracle Diffie-Hellman Assumptions And An Analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)

[BP04] Bellare, M., Palacio, A.: Towards Plaintext-Aware Public-Key Encryption Without Random Oracles. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 48–62. Springer, Heidelberg (2004)

[CS98] Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)

[CS04] Cramer, R., Shoup, V.: Design And Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal of Computing 33(1), 167–226 (2004)

[Dam91] Damgård, I.: Towards Practical Public Key Systems Secure against Chosen Ciphertext Attacks. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 445–456. Springer, Heidelberg (1992)

[DP08] Desmedt, Y., Phan, D.H.: A CCA Secure Hybrid Damgård's ElGamal Encryption. In: Bao, F., Chen, K. (eds.) ProvSec 2008. LNCS, vol. 5324. Springer, Heidelberg (2008)

[Elg85] Elgamal, T.: A Public Key Cryptosystem And A Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)

[Gjø06] Gjøsteen, K.: A New Security Proof for Damgård's ElGamal. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 150–158. Springer, Heidelberg (2006)

[GS04] Gennaro, R., Shoup, V.: A Note on An Encryption Scheme of Kurosawa And Desmedt. Technical Report 2004/194, International Association for Cryptologic Research (August 10, 2004) (last revision May 18 2005), http://eprint.iacr.org/2004/194

[HK07] Hofheinz, D., Kiltz, E.: Secure Hybrid Encryption from Weakened Key Encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)

[KD04] Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)

[KPSY08] Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A New Randomness Extraction Paradigm for Hybrid Encryption. Technical Report 2008/304, International Association for Cryptologic Research (October 2008), http://eprint.iacr.org/2008/304

[Lip08] Lipmaa, H.: On CCA1-Security of Elgamal And Damgård Cryptosystems. Technical Report 2008/234, International Association for Cryptologic Research (October 2008), http://eprint.iacr.org/2008/234

[Sho00] Shoup, V.: Using Hash Functions as A Hedge against Chosen Ciphertext Attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)

A Some Known Public-Key Cryptosystems

Cramer-Shoup Cryptosystem from [CS98]

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and a universal one-way family $\mathcal{UOWHF}$ of hash functions.

Key Setup $\mathsf{pub.gen}$**:** Let $(g_1, g_2) \in \mathbb{G}^2$ be two random generators, let $(\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma) \leftarrow \mathbb{Z}_q^5$. Compute $c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2}$, $d \leftarrow g_1^{\beta_1} g_2^{\beta_2}$, $h \leftarrow g_1^{\gamma}$. Choose $\mathsf{uowhf} \leftarrow \mathcal{UOWHF}$. The public key is $\mathsf{pk} \leftarrow (g_1, g_2, c, d, h, \mathsf{uowhf})$, the private key is $\mathsf{sk} \leftarrow (\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma)$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \mathbb{G}$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u_1 \leftarrow g_1^r$, $u_2 \leftarrow g_2^r$, $e \leftarrow m \cdot h^r$, $v \leftarrow (cd^{\mathsf{uowhf}(u_1, u_2, e)})^r$. The ciphertext is (u_1, u_2, e, v).

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext (u_1, u_2, e, v), do the following. Set $k \leftarrow \mathsf{uowhf}(u_1, u_2, e)$. If $u_1^{\alpha_1 + \beta_1 k} u_2^{\alpha_1 + \beta_1 k} \neq v$ then output $m \leftarrow \perp$. Otherwise, compute $m \leftarrow e / u_1^{\gamma}$ and return m.

Cramer-Shoup Lite Cryptosystem from [CS98, Sect. 5.4]

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q.

Key Setup $\mathsf{pub.gen}$**:** Let $(g_1, g_2) \in \mathbb{G}^2$ be two random generators, let $(\alpha_1, \alpha_2, \gamma) \leftarrow \mathbb{Z}_q^3$. Compute $c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2}$, $h \leftarrow g_1^{\gamma}$. The public key is $\mathsf{pk} \leftarrow (g_1, g_2, c, h)$, the private key is $\mathsf{sk} \leftarrow (\alpha_1, \alpha_2, \gamma)$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \mathbb{G}$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u_1 \leftarrow g_1^r$, $u_2 \leftarrow g_2^r$, $e \leftarrow m \cdot h^r$, $v \leftarrow c^r$. The ciphertext is (u_1, u_2, e, v).

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext (u_1, u_2, e, v), do the following. If $u_1^{\alpha_1} u_2^{\alpha_1} \neq v$ then output $m \leftarrow \bot$. Otherwise, compute $m \leftarrow e/u_1^{\gamma}$ and return m.

Shoup Hybrid Cryptosystem from [Sho00]

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and a universal one-way family $\mathcal{UOWHF}$ of hash functions.

Key Setup $\mathsf{pub.gen}$**:** Generate a random generator $g_1 \leftarrow \mathbb{G}$, and $(w, \alpha, \beta, \gamma) \leftarrow \mathbb{Z}_q^4$. Compute $g_2 \leftarrow g_1^w$, $c \leftarrow g_1^{\alpha}$, $d \leftarrow g_1^{\beta}$, $h \leftarrow g_1^{\gamma}$. Choose $\mathsf{uowhf} \leftarrow \mathcal{UOWHF}$. The public key is $\mathsf{pk} \leftarrow (g_1, g_2, c, d, h, \mathsf{uowhf})$, the private key is $\mathsf{sk} \leftarrow (w, \alpha, \beta, \gamma)$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \{0,1\}^*$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u_1 \leftarrow g_1^r$, $u_2 \leftarrow g_2^r$, $(K, \kappa) \leftarrow \mathsf{kdf}(h^r)$, $e \leftarrow \mathsf{sym.enc}(K; m, \rho)$ for uniform randomizer ρ, $\mathsf{t} \leftarrow \mathsf{mac.tag}(\kappa; e)$, $v \leftarrow (cd^{\mathsf{uowhf}(u_1, u_2)})^r$. The ciphertext is $(u_1, u_2, v, e, \mathsf{t})$.

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext $(u_1, u_2, v, e, \mathsf{t})$, do the following. Set $k \leftarrow \mathsf{uowhf}(u_1, u_2)$, $(K, \kappa) \leftarrow \mathsf{kdf}(u_1^{\gamma})$. If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \bot$ or $u_1^{\alpha + \beta k} \neq v$ or $u_2 \neq u_1^w$ then output $m \leftarrow \bot$. Otherwise, compute $m \leftarrow \mathsf{sym.dec}(K; e)$ and return m.

DHIES Cryptosystem from [ABR01]. The DHIES cryptosystem is very simple but relies on a nonstandard assumption that was called "oracle-DDH" in [ABR01]. Briefly, it is assumed that one cannot distinguish tuples $(g^u, g^v, \mathsf{h}(g^{uv}))$ and (g^u, g^v, r) for random group elements $u, v \leftarrow \mathbb{Z}_q$ and a random string r, even if given access to an oracle that on any input $x \neq g^u$ computes $\mathsf{h}(x^v)$.

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and its randomly chosen generator $g \in \mathbb{G}$. Choose a CPA-secure symmetric cryptosystem $\mathsf{sym} = (\mathsf{sym.gen}, \mathsf{sym.enc}, \mathsf{sym.dec})$, a secure MAC $\mathsf{mac} = (\mathsf{mac.tag}, \mathsf{mac.ver})$, and a hash function family $\mathcal{H}$ from $\mathbb{G}^2$ to the set of keys of sym and mac.

Key Setup $\mathsf{pub.gen}$**:** Choose a hash function $\mathsf{h} \leftarrow \mathcal{H}$. Generate $\alpha \leftarrow \mathbb{Z}_q$. Set $\mathsf{sk} \leftarrow \alpha$ and $\mathsf{pk} \leftarrow (c \leftarrow g^{\alpha}, \mathsf{h})$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \{0,1\}^*$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u \leftarrow g^r$, $(K, \kappa) \leftarrow \mathsf{h}(c^r)$, $e \leftarrow \mathsf{sym.enc}(K; m, \rho)$ for uniform randomizer ρ, $\mathsf{t} \leftarrow \mathsf{mac.tag}(\kappa; e)$. The ciphertext is (u, e, t).

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext (u, e, t), do the following. Compute $(K, \kappa) \leftarrow \mathsf{h}(u^{\alpha})$. If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \bot$ then return $m \leftarrow \bot$ else return $m \leftarrow \mathsf{sym.dec}(K; e)$.

Kurosawa-Desmedt Hybrid Cryptosystem from [KD04]. We give a description due to [GS04] that differs from the original description from [KD04] in two aspects. It replaces the original (information-theoretically) rejection-secure CCA2-secure sym of [KD04] with a CPA-secure sym and a (computationally) secure $\mathsf{mac} = (\mathsf{mac.tag}, \mathsf{mac.ver})$. It also allows to use a computationally secure KDF.

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and its two randomly chosen different generators $g_1, g_2 \in \mathbb{G}$. Choose a CPA-secure symmetric cryptosystem $\mathsf{sym} = (\mathsf{sym.gen}, \mathsf{sym.enc}, \mathsf{sym.dec})$, a secure MAC $\mathsf{mac} = (\mathsf{mac.tag}, \mathsf{mac.ver})$, a KDF kdf from $\mathbb{G}$ to the set of keys of $(\mathsf{sym}, \mathsf{mac})$, and a target-collision-resistant function family $\mathcal{TCR} : \mathbb{G}^2 \to \mathbb{Z}_q$.

Key Setup $\mathsf{pub.gen}$**:** Choose a hash function $\mathsf{tcr} \leftarrow \mathcal{TCR}$. Generate $(\alpha_1, \alpha_2, \beta_1, \beta_2) \leftarrow \mathbb{Z}_q^4$. Set $\mathsf{sk} \leftarrow (\alpha_1, \alpha_2, \beta_1, \beta_2)$ and $\mathsf{pk} \leftarrow (c \leftarrow g_1^{\alpha_1} g_2^{\alpha_2}, d \leftarrow g_1^{\beta_1} g_2^{\beta_2}, \mathsf{tcr})$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \{0,1\}^*$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u_1 \leftarrow g_1^r$, $u_2 \leftarrow g_2^r$, $(K, \kappa) \leftarrow \mathsf{kdf}\left((cd^{\mathsf{tcr}(u_1,u_2)})^r\right)$, $e \leftarrow \mathsf{sym.enc}(K; m, \rho)$ for uniform randomizer ρ, $\mathsf{t} \leftarrow \mathsf{mac.tag}(\kappa; e)$. The ciphertext is $(u_1, u_2, e, \mathsf{t})$.

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext $(u_1, u_2, e, \mathsf{t})$, do the following. Compute $k \leftarrow \mathsf{tcr}(u_1, u_2)$, $(K, \kappa) \leftarrow \mathsf{kdf}(u_1^{\alpha_1+\beta_1 k} u_2^{\alpha_2+\beta_2 k})$. If $\mathsf{mac.ver}(\kappa; e, \mathsf{t}) = \perp$ then return $m \leftarrow \perp$ else return $m \leftarrow \mathsf{sym.dec}(K; e)$.

Hofheinz-Kiltz DDH-Based Cryptosystem. In [HK07, Sect. 4.2], the authors proposed the next DDH-based cryptosystem.

Setup: On input the security parameter λ, return a λ-bit prime q, a group $\mathbb{G}$ of order q, and its randomly chosen generator $g \in \mathbb{G}$. Choose a CCA2-secure symmetric cryptosystem $\mathsf{sym} = (\mathsf{sym.gen}, \mathsf{sym.enc}, \mathsf{sym.dec})$, a KDF kdf from $\mathbb{G}$ to the set of keys of $(\mathsf{sym}, \mathsf{mac})$, and a target-collision-resistant function family $\mathcal{TCR} : \mathbb{G} \to \mathbb{Z}_q$.

Key Setup $\mathsf{pub.gen}$**:** Choose a hash function $\mathsf{tcr} \leftarrow \mathcal{TCR}$. Generate $(\alpha_1, \alpha_2, \beta) \leftarrow \mathbb{Z}_q^3$. Set $\mathsf{sk} \leftarrow (\alpha_1, \alpha_2, \beta)$ and $\mathsf{pk} \leftarrow (c \leftarrow g^{\alpha_1}, d \leftarrow g^{\alpha_2}, h \leftarrow g^{\beta}, \mathsf{tcr})$.

Encryption $\mathsf{pub.enc}$**:** Given a message $m \in \{0,1\}^*$, do the following. First, set $r \leftarrow \mathbb{Z}_q$ and then $u_1 \leftarrow g^r$, $u_2 \leftarrow (c^{\mathsf{tcr}(u_1)} \cdot d)^r$, $K \leftarrow \mathsf{kdf}(h^r)$, $e \leftarrow \mathsf{sym.enc}(K; m, \rho)$ for uniform randomizer ρ. The ciphertext is (u_1, u_2, e).

Decryption $\mathsf{pub.dec}$**:** Given a ciphertext (u_1, u_2, e), do the following. If $u_1 \notin \mathbb{G}$ or $u_1^{\alpha_1 \cdot \mathsf{tcr}(u_1)+\alpha_2} \neq u_2$ then return $\perp$. Compute $K \leftarrow \mathsf{kdf}(u_1^{\beta})$. Return $m \leftarrow \mathsf{sym.dec}(K; e)$, possibly $m = \perp$.