A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants

Hovav Shacham*
Weizmann Institute of Science
hovav.shacham@weizmann.ac.il

Abstract

We describe a CCA-secure public-key encryption scheme, in the Cramer-Shoup paradigm, based on the Linear assumption of Boneh, Boyen, and Shacham. Through a comparison to the Kiltz tag-encryption scheme from TCC 2006, our scheme gives evidence that the Cramer-Shoup paradigm yields CCA encryption with shorter ciphertexts than the Canetti-Halevi-Katz paradigm.

We present a generalization of the Linear assumption into a family of progressively weaker assumptions and show how to instantiate our Linear Cramer-Shoup encryption using the progressively weaker members of this family.

1 Introduction

Cramer and Shoup [16, 19] presented the first practical CCA-secure public key encryption system, based on the decisional Diffie-Hellman (DDH) assumption. They later generalized their construction by considering an algebraic primitive they call universal hash proof systems [18]; they showed that this framework yields not only the original DDH-based Cramer-Shoup scheme but also encryption schemes based on quadratic residuosity and on Paillier's assumption [33].

Canetti, Halevi, and Katz (CHK) [15] proposed an alternative way to obtain CCA-secure encryption schemes, which takes a selective-ID secure identity-based encryption (IBE) scheme and a one-time strongly unforgeable signature scheme and yields a CCA-secure encryption scheme by means of a black-box transformation. Boneh and Katz [9] showed a similar transformation in which a MAC takes the place of the signature, so the resulting encryption scheme is more efficient.[1] Either of these transformations, applied to the (first) Boneh-Boyen IBE [4], yields a CCA-secure encryption scheme based on the decisional bilinear Diffie-Hellman assumption. The resulting scheme is less efficient than Cramer-Shoup, because decryption requires computing a pairing, an expensive operation [21].

At TCC 2006, Kiltz [24] observed that the full power of an IBE scheme is unnecessary for applying the CHK transform, and introduced a weaker primitive, selective-tag weakly CCA-secure tag encryption, that is sufficient. (This primitive, he shows, is implied both by selective-ID IBE and by weakly CCA-secure tag encryption, thus unifying the CHK transformation with a similar one proposed independently by MacKenzie, Reiter, and Yang [27] and based on tag encryption.)

*Supported by a Koshland Scholars Program fellowship.

[1]See also the combined journal paper [7].

In addition, he describes a concrete selective-tag weakly CCA-secure tag encryption scheme, based on the Linear assumption introduced by Boneh, Boyen, and Shacham [6]. The Kiltz tag encryption scheme makes use of a pairing in the proof of security, but not in the encryption or decryption algorithms, and the same is true also of the CCA-secure encryption scheme obtained from it by means of the CHK transformation.

Our Contribution. We develop a variant of Cramer-Shoup encryption that is secure under the Linear assumption. Like the DDH-based Cramer-Shoup scheme, our scheme is CCA secure. Our scheme makes no use of pairings either in the encryption or decryption algorithms or in the proof of security; but it remains secure even when instantiated in a group for which there exists a computable pairing. Using our scheme one can construct a variant of Boneh-Boyen-Shacham group signatures secure in the full Bellare-Micciancio-Warinschi model [3] rather than a relaxation of that model [6].

In addition to being practically useful as a replacement for Cramer-Shoup in bilinear groups, our construction fits within a line of research that seeks to weaken the assumption underlying Cramer-Shoup. For example, Shoup demonstrated a variant of Cramer-Shoup that admits a proof of security under DDH in the standard model and under CDH using random oracles [37], and Gennaro, Krawczyk, and Rabin showed how one can securely instantiate Cramer-Shoup in $\mathbb{Z}_p^*$ provided DDH holds in some subgroup [22].

More importantly, our scheme set side-by-side the Kiltz scheme allows, for the first time, an apples-to-apples comparison of the Cramer-Shoup and CHK methodologies for obtaining CCA-secure encryption. (Previous to our work, no single assumption was the basis for encryption schemes in both the Cramer-Shoup and a CHK paradigms.) The comparison shows that, at least in this case, the CHK methodology yields the scheme with shorter public keys and faster encryption and decryption algorithms but the Cramer-Shoup methodology yields the scheme with substantially shorter ciphertexts. The Cramer-Shoup–based construction also has the advantage of not requiring the presence of a computable pairing.

We generalize DDH and the Linear assumption into a family of assumptions $\{\mathrm{L}_k\}_{k\geq 1}$ that are progressively *weaker*. (Two members of this family are already familiar: L_1 is DDH, L_2 is the Linear assumption.) Specifically, we show that, in Shoup's generic group model [36], L_{k+1} holds even when L_k does not. We then describe a family of Cramer-Shoup variants secure respectively under each L_k. Other DDH-based cryptosystems can be similarly generalized. Members of the L_k family for large k can be used as a hedge against the development of algorithms for solving DDH or Linear.

2 Preliminaries

2.1 The Linear Assumption

Boneh, Boyen, and Shacham [6] introduced a decisional assumption, called Linear, intended to take the place of DDH in groups — in particular, bilinear groups [23] — where DDH is easy. For this setting, the Linear problem has desirable properties, as Boneh, Boyen and Shacham show: it is hard if DDH is hard, but, at least in generic groups [36], remains hard even if DDH is easy.

Letting G be a cyclic multiplicative group of prime order p, and letting g_1, g_2, and g_3 be arbitrary generators of G, consider the following problem:

Linear Problem in G: Given $g_1, g_2, g_3, g_1^a, g_2^b, g_3^c \in G$ as input, output `yes` if $a + b = c$ and `no` otherwise.

The advantage of an algorithm $\mathcal{A}$ in deciding the Linear problem in G is

$$\mathbf{Adv}_{\mathcal{A}}^{\text{linear}} \stackrel{\text{def}}{=} \left| \begin{array}{l} \Pr\left[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^{a+b}) = \texttt{yes} : g_1, g_2, g_3 \xleftarrow{\text{R}} G, a, b \xleftarrow{\text{R}} \mathbb{Z}_p\right] \\ \quad - \Pr\left[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, \eta) = \texttt{yes} : g_1, g_2, g_3, \eta \xleftarrow{\text{R}} G, a, b \xleftarrow{\text{R}} \mathbb{Z}_p\right] \end{array} \right| ,$$

with the probability taken over the uniform random choice of the parameters to $\mathcal{A}$ and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}$ (t, ϵ)-decides Linear in G if $\mathcal{A}$ runs in time at most t, and $\mathbf{Adv}_{\mathcal{A}}^{\text{linear}}$ is at least ϵ.

Definition 2.1. We say that the (t, ϵ)-Decision Linear Assumption holds in G if no algorithm (t, ϵ)-decides the Decision Linear problem in G.

The Linear problem is well-defined in any group where DDH is; its main use, however, is in bilinear groups, for which see, e.g., [8, 10, 34].

2.1.1 Linear Encryption

Boneh, Boyen, and Shacham describe a natural variant of ElGamal encryption that is CPA-secure under the Linear assumption:

LE.Kg. Choose a random generator $g_3 \xleftarrow{\text{R}} G$ and exponents $x_1, x_2 \xleftarrow{\text{R}} \mathbb{Z}_p$, and set $g_1 \leftarrow g_3^{1/x_1}$ and $g_2 \leftarrow g_3^{1/x_2}$. The public key is $pk = (g_1, g_2, g_3) \in G^3$; the secret key is $sk = (x_1, x_2) \in \mathbb{Z}_p^2$.

LE.Enc(pk, M)**.** To encrypt a message $M \in G$, parse pk as $(g_1, g_2, g_3) \in G^3$, choose random exponents $r_1, r_2 \xleftarrow{\text{R}} \mathbb{Z}_p$, and set

$$u_1 \leftarrow g_1^{r_1} \qquad \text{and} \qquad u_2 \leftarrow g_2^{r_2} \qquad \text{and} \qquad u_3 \leftarrow M \cdot g_3^{r_1+r_2} \ ;$$

the ciphertext is $ct = (u_1, u_2, u_3) \in G^3$.

LE.Dec(sk, ct)**.** Parse the private key sk as $(x_1, x_2) \in \mathbb{Z}_p$ and the ciphertext ct as $(u_1, u_2, u_3) \in \mathbb{Z}_p^3$, and compute and output $M \leftarrow u_3/(u_1^{x_1} u_2^{x_2})$.

Correctness is easy to verify, and CPA security follows directly from the Linear assumption.

2.2 DDH Cramer-Shoup

We recall the Cramer-Shoup encryption scheme based on DDH. We use the notation of the conference version of the Cramer-Shoup paper [16]. (We make one change in the derivation of h, following scheme CS1 of [19] rather than CS1b.) In what follows we retain this notation for our own schemes, for ease of comparison.

Let $\mathcal{HF}$ be a family of universal one-way hash functions [31] from G^3 to $\mathbb{Z}_p$. To simplify the notation, we let $\mathcal{HF}$ stand also for the family's keyspace, and write $H \xleftarrow{\text{R}} \mathcal{HF}$ for choosing a random such function.

CS.Kg. Choose random generators $g_1, g_2 \xleftarrow{R} G$ and exponents $x_1, x_2,\ y_1, y_2,\ z_1, z_2 \xleftarrow{R} \mathbb{Z}_p$ and set

$$c \leftarrow g_1^{x_1} g_2^{x_2} \qquad \text{and} \qquad d \leftarrow g_1^{y_1} g_2^{y_2} \qquad \text{and} \qquad h \leftarrow g_1^{z_1} g_2^{z_2} \ .$$

In addition, choose a UOWHF $H \xleftarrow{R} \mathcal{HF}$. The public key is $pk = (g_1, g_2, c, d, h, H) \in G^5 \times \mathcal{HF}$; the secret key is $sk = (x_1, x_2, y_1, y_2, z_1, z_2) \in \mathbb{Z}_p^6$.

CS.Enc(pk, M)**.** To encrypt a message $M \in G$, parse pk as $(g_1, g_2, c, d, h, H) \in G^5 \times \mathcal{HF}$. Choose a random exponent $r \xleftarrow{R} \mathbb{Z}_p$, and set

$$u_1 \leftarrow g_1^r \qquad \text{and} \qquad u_2 \leftarrow g_2^r \qquad \text{and} \qquad e \leftarrow M \cdot h^r;$$

now compute $\alpha \leftarrow H(u_1, u_2, e)$ and, finally, $v \leftarrow (cd^\alpha)^r$. The ciphertext is $ct = (u_1, u_2, e, v) \in G^4$.

CS.Dec(pk, sk, ct)**.** Parse the public key pk as $(g_1, g_2, c, d, h, H) \in G^5 \times \mathcal{HF}$, the private key sk as $(x_1, x_2, y_1, y_2, z_1, z_2) \in \mathbb{Z}_p^6$ and the ciphertext ct as $(u_1, u_2, e, v) \in G^3$. Compute $\alpha \leftarrow H(u_1, u_2, e)$ and test that

$$u_1^{x_1 + \alpha y_1} \cdot u_2^{x_2 + \alpha y_2} \stackrel{?}{=} v$$

holds. If it does not, output "`reject`". Otherwise, compute and output $M \leftarrow e/(u_1^{z_1} u_2^{z_2})$.

3 Linear Cramer-Shoup

We describe a variant of Cramer-Shoup encryption based on the Linear assumption. Our scheme makes no use of pairings either in the encryption or decryption algorithms or in the proof of security. It can be instantiated in groups where DDH is easy as well as in groups where DDH is hard.

For the Linear Cramer-Shoup scheme, let $\mathcal{HF}'$ be a family of universal one-way hash functions from G^4 to $\mathbb{Z}_p$.

LCS.Kg. Choose random generators $g_1, g_2, g_3 \xleftarrow{R} G$ and exponents

$$x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3 \xleftarrow{R} \mathbb{Z}_p$$

and set

$$\begin{array}{lll} c_1 \leftarrow g_1^{x_1} g_3^{x_3} & \qquad d_1 \leftarrow g_1^{y_1} g_3^{y_3} & \qquad h_1 \leftarrow g_1^{z_1} g_3^{z_3} \\ c_2 \leftarrow g_2^{x_2} g_3^{x_3} & \qquad d_2 \leftarrow g_2^{y_2} g_3^{y_3} & \qquad h_2 \leftarrow g_2^{z_2} g_3^{z_3} \ . \end{array}$$

In addition, choose a UOWHF $H \xleftarrow{R} \mathcal{HF}'$. The public key is $pk = (g_1, g_2, g_3,\ c_1, c_2, d_1, d_2, h_1, h_2, H) \in G^9 \times \mathcal{HF}'$; the secret key is $sk = (x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3) \in \mathbb{Z}_p^9$.

LCS.Enc(pk, M)**.** To encrypt a message $M \in G$, parse pk as $pk = (g_1, g_2, g_3,\ c_1, c_2, d_1, d_2, h_1, h_2, H) \in G^9 \times \mathcal{HF}'$. Choose random exponents $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$, and set

$$u_1 \leftarrow g_1^{r_1} \qquad \text{and} \qquad u_2 \leftarrow g_2^{r_2} \qquad \text{and} \qquad u_3 \leftarrow g_3^{r_1 + r_2} \qquad \text{and} \qquad e \leftarrow M \cdot h_1^{r_1} h_2^{r_2};$$

now compute $\alpha \leftarrow H(u_1, u_2, u_3, e)$ and, finally, $v \leftarrow (c_1 d_1^\alpha)^{r_1} \cdot (c_2 d_2^\alpha)^{r_2}$. The ciphertext is $ct = (u_1, u_2, u_3, e, v) \in G^5$.

$\textsf{LCS.Dec}(pk, sk, ct)$. Parse the public key pk as $(g_1, g_2, g_3,\ c_1, c_2, d_1, d_2, h_1, h_2, H) \in G^9 \times \mathcal{HF}'$, the private key sk as $(x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3) \in \mathbb{Z}_p^9$, and the ciphertext ct as $(u_1, u_2, u_3, e, v) \in G^4$. Compute $\alpha \leftarrow H(u_1, u_2, u_3, e)$ and test that

$$u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3} \stackrel{?}{=} v \tag{1}$$

holds. If it does not, output "`reject`". Otherwise, compute and output $M \leftarrow e/(u_1^{z_1} u_2^{z_2} u_3^{z_3})$.

Correctness. If the keys and encryption are generated according to the algorithms above, the test (1) in $\textsf{Dec}$ will be satisfied, since we will then have

$$\begin{aligned} u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3} &= \left(g_1^{r_1}\right)^{x_1+\alpha y_1} \cdot \left(g_2^{r_2}\right)^{x_2+\alpha y_2} \cdot \left(g_3^{r_1+r_2}\right)^{x_3+\alpha y_3} \\ &= \left(g_1^{x_1+\alpha y_1} g_3^{x_3+\alpha y_3}\right)^{r_1} \cdot \left(g_2^{x_2+\alpha y_2} g_3^{x_3+\alpha y_3}\right)^{r_2} \\ &= (c_1 d_1^{\alpha})^{r_1} \cdot (c_2 d_2^{\alpha})^{r_2} = v \ , \end{aligned}$$

as required. Next, the decryption will obtain the correct M, since

$$\begin{aligned} e/(u_1^{z_1} u_2^{z_2} u_3^{z_3}) &= e \ / \ \left(g_1^{r_1 z_1} g_2^{r_2 z_2} g_3^{(r_1+r_2)(z_3)}\right) \\ &= (e) \ / \ \left((g_1^{z_1} g_3^{z_3})^{r_1} (g_2^{z_2} g_3^{z_3})^{r_2}\right) \\ &= (M \cdot h_1^{r_1} h_2^{r_2}) \ / \ (h_1^{r_1} h_2^{r_2}) = M \ . \end{aligned}$$

Security. We now show that the the LCS scheme is CCA secure. The proof closely follows that of the Cramer-Shoup scheme.

Theorem 3.1. *The LCS scheme is secure in the CCA sense if $\mathcal{HF}'$ a secure UOWHF family and the Linear assumption holds in G.*

Proof. We show how to decide instances of the Linear problem using a CCA distinguisher $\mathcal{A}$. Consider an algorithm $\mathcal{B}$ that is given, as input, an instance $(g_1, g_2, g_3, u_1, u_2, u_3)$; its goal is to output `yes` if $\log_{g_3} u_3 = \log_{g_1} u_1 + \log_{g_2} u_2$, `no` otherwise. As in the real setup algorithm, algorithm $\mathcal{B}$ chooses

$$x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3 \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$$

and sets

$$\begin{aligned} c_1 &\leftarrow g_1^{x_1} g_3^{x_3} \qquad & d_1 &\leftarrow g_1^{y_1} g_3^{y_3} \qquad & h_1 &\leftarrow g_1^{z_1} g_3^{z_3} \\ c_2 &\leftarrow g_2^{x_2} g_3^{x_3} & d_2 &\leftarrow g_2^{y_2} g_3^{y_3} & h_2 &\leftarrow g_2^{z_2} g_3^{z_3} \ . \end{aligned}$$

It also chooses a UOWHF $H \stackrel{\mathrm{R}}{\leftarrow} \mathcal{HF}'$. It provides to $\mathcal{A}$ the public key $pk = (g_1, g_2, g_3,\ c_1, c_2, d_1, d_2, h_1, h_2, H)$, and keeps to itself the secret key $sk = (x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3)$. Algorithm $\mathcal{B}$ answers $\mathcal{A}$'s decryption queries by following $\textsf{LCS.Dec}(pk, sk, \cdot)$. (Note that this algorithm does not require knowledge of the discrete-log relationships amongst g_1, g_2, and g_3, and that $\mathcal{B}$ knows the secret key.) When $\mathcal{A}$ submits the messages M_0 and M_1 on which it wishes to be challenged, $\mathcal{B}$ chooses $b \stackrel{\mathrm{R}}{\leftarrow} \{0, 1\}$, sets $e \leftarrow M_b \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}$, $\alpha \leftarrow H(u_1, u_2, u_3, e)$, and $v \leftarrow u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3}$. It then supplies to $\mathcal{A}$ the challenge ciphertext $ct^* = (u_1, u_2, u_3, e, v)$. Algorithm $\mathcal{B}$ then responds to $\mathcal{A}$'s further decryption queries as before. Finally $\mathcal{A}$ outputs its guess b' for b. If $b = b'$, algorithm $\mathcal{B}$ outputs `yes`; otherwise it answers `no`.

Clearly, if $\mathcal{A}$ has a different advantage in guessing the bit b when $\mathcal{B}$ is run with a Linear tuple $(g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2})$ and when $\mathcal{B}$ is run with a random tuple $(g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, \eta)$, we obtain a distinguisher for the Linear problem. In the remainder of the proof, we establish that in the first case $\mathcal{A}$'s advantage is nonnegligible, as in the real distinguishing game, whereas in the second case $\mathcal{A}$'s advantage is negligible.

First, suppose that $\mathcal{B}$'s input is a Linear tuple $(g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2})$ for some (unknown) r_1 and r_2. We will show that the challenge ciphertext is properly formed and distributed. Since the public key and the decryption queries are formed exactly as in the real distinguishing game, this shows that the adversary's view is the same as in the real distinguishing game, and so is its advantage in guessing b. The first three components of the challenge ciphertext, $u_1 = g_1^{r_1}$, $u_2 = g_2^{r_2}$, and $u_3 = g_3^{r_1+r_2}$, are clearly correctly formed. Algorithm $\mathcal{B}$ computes e according to a different formula than is specified in $\mathsf{LCS.Enc}$, but it in fact computes the correct value, since

$$h_1^{r_1} h_2^{r_2} = \left(g_1^{z_1} g_3^{z_3}\right)^{r_1} \cdot \left(g_2^{z_2} g_3^{z_3}\right)^{r_2} = g_1^{r_1 z_1} g_2^{r_2 z_2} g_3^{(r_1+r_2)(z_3)} = u_1^{z_1} u_2^{z_2} u_3^{z_3} \quad .$$

Next, α is computed by $\mathcal{B}$ per $\mathsf{LCS.Enc}$. Finally, $\mathcal{B}$ computes the correct value for v since

$$\begin{aligned} (c_1 d_1^{\alpha})^{r_1} \cdot (c_2 d_2^{\alpha})^{r_2} &= \left(g_1^{x_1+\alpha y_1} g_3^{x_3+\alpha y_3}\right)^{r_1} \cdot \left(g_2^{x_2+\alpha y_2} g_3^{x_3+\alpha y_3}\right)^{r_2} \\ &= g_1^{(r_1)(x_1+\alpha y_1)} \cdot g_2^{(r_2)(x_2+\alpha y_2)} \cdot g_3^{(r_1+r_2)(x_3+\alpha y_3)} \\ &= u_1^{x_1+\alpha y_1} \cdot u_2^{x_2+\alpha y_2} \cdot u_3^{x_3+\alpha y_3} \quad . \end{aligned}$$

Observe that algorithm $\mathcal{B}$ chooses the secret key $(x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3)$ itself, and can therefore answer decryption queries by following $\mathsf{LCS.Dec}$. This means that algorithm $\mathcal{A}$'s decryption queries are answered in exactly the same way in both the simulation and in the attack game. Thus we have established that, when $\mathcal{B}$'s input is a linear tuple, it simulates $\mathcal{A}$'s environment perfectly.

To complete the proof, we argue that, when $\mathcal{B}$'s input is a random tuple, the bit b remains independent of $\mathcal{A}$'s view except with negligible probability.

Let "$\log(\cdot)$" stand for "$\log_{g_1}(\cdot)$" and define $w_2 = \log g_2$ and $w_3 = \log g_3$. Consider the three elements (z_1, z_2, z_3) of the private key. The public key values h_1 and h_2 constrain these to line on the line at the intersection of the planes defined by

$$\log h_1 = z_1 + w_0 z_0 \qquad \text{and} \qquad \log h_2 = w_2 z_2 + w_0 z_0 \quad . \tag{2}$$

Now, a decryption query for a valid ciphertext whose first three components form a valid Linear tuple $(u_1', u_2', u_3') = (g_1^{r_1'}, g_2^{r_2'}, g_3^{r_1'+r_2'})$ will allow the adversary to obtain $\left((u_1')^{z_1}(u_2')^{z_2}(u_3')^{z_3}\right)$; but in this case we have

$$\log\left((u_1')^{z_1}(u_2')^{z_2}(u_3')^{z_3}\right) = (r_1')(z_1 + w_3 z_3) + (r_2')(w_2 z_2 + w_3 z_3) \quad ,$$

which is linearly dependent on values already known to the adversary from (2). This analysis doesn't hold if the decryption oracle accepts a ciphertext whose first three elements do not form a linear tuple; below we will show that the decryption oracle accepts such invalid ciphertexts only with negligible probability.

Now consider the challenge ciphertext $ct^* = (u_1, u_2, u_3, e, v)$. Let $u_1 = g_1^{r_1}$, $u_2 = g_2^{r_2}$, and $u_3 = g_3^{r_3}$. Except with negligible probability we have $r_3 \neq r_1 + r_2$. The message M_b is blinded in e by the value $u_1^{z_1} u_2^{z_2} u_3^{z_3}$ whose discrete logarithm is

$$\log(u_1^{z_1} u_2^{z_2} u_3^{z_3}) = r_1 z_1 + r_2 w_2 z_2 + r_3 w_3 z_3 = (r_1)(z_1 + w_3 z_3) + (r_2)(w_2 z_2 + w_3 z_3) + (\Delta r)(w_3 z_3) \quad ,$$

where we set $\Delta r = r_3 - r_1 - r_2 \neq 0$. Thus to an adversary who has received decryption queries only for valid ciphertexts this value is independent of its view, namely of (2). This means that M_b is independent of the adversary's view, even given e, and thus b is, too.

What remains to show is that the decryption oracle accepts invalid ciphertexts only with negligible probability. What we will in fact show is that, given that through query i the decryption oracle has not accepted an invalid ciphertext, the probability that it accepts an invalid one at query $i+1$ is negligible. There are in fact two cases to consider: decryption queries made by algorithm $\mathcal{A}$ before it has seen the challenge ciphertext, and decryption queries made after it has seen the challenge ciphertext. In fact, the analysis in the first case follows from the analysis in the second, since the challenge ciphertext gives the adversary more information about the values $(x_1, x_2, x_3,\ y_1, y_2, y_3)$ by which Dec checks ciphertext validity.

Observe that the values $(x_1, x_2, x_3,\ y_1, y_2, y_3)$ lie along a plane in $\mathbb{Z}_p^6$ specified by

$$\begin{aligned} \log c_1 &= x_1 + w_3 x_3 & \qquad \log c_2 &= w_2 x_2 + w_3 x_3 \\ \log d_1 &= y_1 + w_3 y_3 & \qquad \log d_2 &= w_2 y_2 + w_3 y_3 \end{aligned} \tag{3}$$

In the second case, then, let $ct^* = (u_1, u_2, u_3, e, v)$ be the challenge ciphertext, and suppose that $\mathcal{A}$ submits a decryption query $(u_1', u_2', u_3', e', v')$. Here $(u_1', u_2', u_3') = (g_1^{r_1'}, g_2^{r_2'}, g_3^{r_3'})$, with $r_3' \neq r_1' + r_2'$. Let $\alpha = H(u_1, u_2, u_3, e)$ and $\alpha' = H(u_1', u_2', u_3', e')$. There are three possibilities:

Case 1. $(u_1, u_2, u_3, e) = (u_1', u_2', u_3', e')$, but $v \neq v'$. In this case, the decryption oracle will reject, since v as calculated in generating ct^* is the only correct checksum value for (u_1, u_2, u_3, e).

Case 2. $(u_1, u_2, u_3, e) \neq (u_1', u_2', u_3', e')$, yet $\alpha = \alpha'$. In this case, the adversary has generated a hash collision. We will deal with this case at the end of the proof, showing how to use an adversary that triggers it to break the UOWHF security of $\mathcal{HF}'$. There thus remains:

Case 3. $(u_1, u_2, u_3, e) \neq (u_1', u_2', u_3', e')$, and $\alpha \neq \alpha'$.

In this third case, we ask: What is the probability, given the adversary's view, that v' is correctly chosen, so that the decryption algorithm accepts it? We can write the equation expressing this, along with the equations expressing the constraints (3) and the constraint on the value v in ct^* in matrix form as

$$\begin{pmatrix} \log c_1 \\ \log c_2 \\ \log d_1 \\ \log d_2 \\ \log v \\ \log v' \end{pmatrix} = \begin{pmatrix} 1 & 0 & w_3 & 0 & 0 & 0 \\ 0 & w_2 & w_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & w_3 \\ 0 & 0 & 0 & 0 & w_2 & w_3 \\ r_1 & r_2 w_2 & r_3 w_3 & \alpha r_1 & \alpha r_2 w_2 & \alpha r_3 w_3 \\ r_1' & r_2' w_2 & r_3' w_3 & \alpha' r_1' & \alpha' r_2' w_2 & \alpha' r_3' w_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} .$$

What we wish to show is that the last line is independent of the others, so that the correct checksum v' is independent of the adversary's view. But, denoting the 6×6 matrix by M, we observe that

$$\det M = -w_2^2 w_3^2 (\alpha - \alpha')(r_3 - r_1 - r_2)(r_3' - r_1' - r_2') \neq 0 \ ,$$

so the equations are indeed independent.

What remains is only to deal with the second case above, in which $\mathcal{A}$ finds a hash collision. Against this type of adversary, we deploy a different simulation strategy. We choose generators

$g_1, g_2, g_3 \xleftarrow{R} G$ at random, along with values $u_1, u_2, u_3, e \xleftarrow{R} G$. With overwhelming probability, $(g_1, g_2, g_3, u_1, u_2, u_3)$ is not a Linear tuple. We now provide (u_1, u_2, u_3, e) to the UOWHF challenger, which responds with a hash function $H \in \mathcal{HF}'$. We now follow the simulation as specified for algorithm $\mathcal{B}$, but set the first four components of the challenge ciphertext to be (u_1, u_2, u_3, e); that is, we do not encrypt the message M_b at all. (The last component, v, is computed as it is by $\mathcal{B}$.) As argued above, the value encrypted in (u_1, u_2, u_3, e) remains independent of $\mathcal{A}$'s view unless it obtains the decryption of an invalid ciphertext; but this happens, with invalid queries of Case 1 or Case 3, with negligible probability, again as argued above. Thus the modified simulation is the same as the original simulation in the adversary's view. When, at last, the adversary makes its Case 2 query for a ciphertext $(u_1', u_2', u_3', e', v')$ such that $(u_1', u_2', u_3', e') \neq (u_1, u_2, u_3, e)$ and yet $H(u_1', u_2', u_3', e') = H(u_1, u_2, u_3, e)$, we immediately obtain the UOWHF break. $\square$

It is also possible to analyze the LCS scheme using the universal hash proof paradigm of [18]. We give the details in Appendix A.

3.1 Comparison to Kiltz Tag Encryption

With our scheme, it is possible to give an apples-to-apples comparison, for the first time, of CCA-secure encryption schemes obtained with the Cramer-Shoup (CS) and Canetti-Halevi-Katz (CHK) methodologies. Until recently, Cramer-Shoup encryption schemes were available from DDH, quadratic residuosity, and Paillier's assumption [17, 19], whereas Canetti-Halevi-Katz encryption schemes were available from decisional bilinear Diffie-Hellman [15, 4] and the Linear assumption [24]. Since there was no overlap in the underlying assumption lists, it was impossible to compare fairly any instantiations of the two methodologies. With our Linear Cramer-Shoup scheme given above, there is, for the first time, an overlap: our scheme and that given by Kiltz [24], under the Linear assumption. By comparing these two we also give a comparison of the methodologies by which they were constructed. This is what we mean by an "apples-to-apples comparison." It would, of course, also be interesting were it possible to compare the methodologies more generally, independent of any particular assumption. We view the present work as the first step towards this goal.[2]

To facilitate the comparison, we recall Kiltz's tag encryption scheme based on the Linear assumption [24]. Kiltz shows that the scheme is a selective-tag weakly-CCA secure tag encryption, and that this notion suffices for applying the CHK transform to obtain a CCA-secure encryption scheme.

Unlike the IBE schemes to which the CHK transform was first applied, Kiltz's scheme does not require pairing evaluation either to encrypt or decrypt. The pairing is used in the proof of security, however, so the scheme can nevertheless only be instantiated in bilinear groups. (Or, more generally, in a "gap" group [23] where DDH is easy and Linear is hard.)

The scheme is as follows.[3]

KT.Kg. Choose a random generator $z \xleftarrow{R} G$ and exponents $x_1, x_2 \xleftarrow{R} \mathbb{Z}_p$, and set $g_1 \leftarrow z^{1/x_1}$ and $g_2 \leftarrow z^{1/x_2}$; then choose $y_1, y_2 \xleftarrow{R} \mathbb{Z}_p$ and set $u_1 \leftarrow g_1^{y_1}$ and $u_2 \leftarrow g_2^{y_2}$. The public key is $pk = (g_1, g_2, z, u_1, u_2) \in G^5$; the secret key is $sk = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_p^4$.

[2] Or, with apologies to Kant, a prolegomena to any future metacomparison.

[3] In the description, we retain Kiltz's notation. A version closer to our own notation would require the following changes: for z, read g_3; for u_1 and u_2, c_1 and c_2; for C_1 and C_2, u_1 and u_2; for E, e; for $D_!$ and D_2, v_1 and v_2.

KT.TEnc(pk, t, M)**.** To encrypt a message $M \in G$ and tag $t \in \mathbb{Z}_p$, parse pk as $pk = (g_1, g_2, z, u_1, u_2) \in G^5$. Choose random exponents $r_1, r_2 \xleftarrow{\text{R}} \mathbb{Z}_p$, and set

$$C_1 \leftarrow g_1^{r_1} \qquad C_2 \leftarrow g_2^{r_2} \qquad E \leftarrow M \cdot z^{r_1+r_2} \qquad D_1 \leftarrow \left(u_1 z^t\right)^{r_1} \qquad D_2 \leftarrow \left(u_2 z^t\right)^{r_2} \ .$$

The ciphertext is $ct = (C_1, C_2, E, D_1, D_2) \in G^5$.

KT.TDec(sk, t, ct)**.** To decrypt ct with tag $t \in \mathbb{Z}_p$, parse the private key sk as $sk = (x_1, x_2, y_1, y_2) \in \mathbb{Z}_p^4$ and the ciphertext as $ct = (C_1, C_2, E, D_1, D_2) \in G^5$. Test that

$$C_1^{x_1 t + y_1} \stackrel{?}{=} D_1 \qquad \text{and} \qquad C_2^{x_2 t + y_2} \stackrel{?}{=} D_2$$

both hold. If they do not, output "`reject`".[4] Otherwise, compute and output $M \leftarrow e/(C_1^{x_1} C_2^{x_2})$.

Performance Comparison. We now compare Linear Cramer-Shoup to the CCA encryption scheme derived via the CHK transform from the KT scheme above.

This KT+CHK scheme has two advantages over the LCS scheme. First, its public and private keys are shorter than those in Linear Cramer-Shoup. Second, its decryption and encryption algorithms are faster: properly implemented, they require about one exponentiation less.

However, LCS has a major advantage over KT+CHK: it gives shorter ciphertexts. There are two reasons. First, it does not incur overhead from the CHK transform. Second, it does not require a pairing, so it can be instantiated in groups where element representation grows more slowly.

Ciphertexts in both the Kiltz tag encryption scheme and Linear Cramer-Shoup consist of five elements of the group G. However, Linear Cramer-Shoup is a CCA encryption scheme as is, whereas Kiltz's scheme is not; the KT+CHK encryption scheme obtained by means of CHK incurs some overhead because of the transform. Specifically, the CHK transform affixes to the underlying ciphertext a signature verification key and a signature. The Boneh-Katz variant affixes a MAC, a commitment to the MAC key, and the encryption of the corresponding decommitment. Even the second, more efficient of these adds several hundred bits to the ciphertext at the 80-bit security levels [7, Table 1] — one or two extra group elements, in effect.

It would be possible to avoid the CHK overhead by applying the Boyen-Mei-Waters (BMW) transform instead [12]. However, since the KT scheme is only secure in a "selective-tag" sense, the BMW transform would apply directly only to a tag-KEM variant (cf. [1]) of Kiltz's scheme, and it would yield a CCA-secure KEM, not a CCA encryption scheme. One could obtain a fully-secure tag encryption from Kiltz's scheme by replacing the Boneh-Boyen–like tag "hashes" [4] in D_1 and D_2 with the Waters hash [38] and apply the BMW transform to obtain a CCA-secure encryption scheme; but using the Waters hash would add several exponentiations to the encryption and decryption algorithms and greatly increase the size of the public parameters, which nullifies the advantages listed above for KT+CHK.

Since the Kiltz tag encryption scheme makes use of the pairing in its proof of security, it can only be instantiated safely in groups that feature a computable pairing, even though none of the algorithms specified in the scheme makes use of the pairing. Because of the Menezes-Okamoto-Vanstone reduction [28], there exist subexponential discrete log algorithms in such groups, so their

[4] The decryption procedure described here is a variant of that given by Kiltz. He recommends a different procedure that performs the validity test and decryption simultaneously; for invalid ciphertexts, it outputs a random element of G rather than an explicit rejection. Kiltz's procedure saves an exponentiation over the one given here.

size must be scaled more than linearly in the security parameter. By contrast, the Linear Cramer-Shoup scheme does not require a pairing (though it remains secure in the presence of one), and can therefore be instantiated in elliptic-curve groups where discrete logarithm is exponentially hard.

At the 80-bit security level, there exist groups with a computable pairing whose representation is as short — 160 bits — as that of elliptic curve groups without a pairing,[5] but for larger security levels a marked difference develops. For example, for 256-bit security, the best known curve choice with the computable pairing necessary for the Kiltz scheme yields a group G whose elements have 1024-bit representation.[6] By contrast, Linear Cramer-Shoup can be instantiated on elliptic curves without a computable pairing, so that elements of G have 512-bit representation.

Since the algorithms of the Kiltz scheme do not actually use the pairing, it would be possible to instantiate it on groups without a computable pairing, which would eliminate the scaling advantage of Linear Cramer-Shoup; but in such an instantiation the scheme could be considered secure only under a stronger "gap" variant [32] of the Linear assumption in which the reduction algorithm has access to a DDH oracle.

3.2 Application: Fully-Anonymous Group Signatures

The Linear assumption and the Linear encryption of Section 2.1.1 were introduced by Boneh, Boyen, and Shacham for use in a group signature scheme. Their scheme follows the paradigm laid out by Camenisch and Stadler [14]: a group signature consists of the encryption, under the group manager's key, of (part of) the user's membership certificate, together with a NIZK proof that the encryption was correctly performed and the certificate is valid. Boneh, Boyen, and Shacham use Strong Diffie-Hellman [5] in a bilinear group for the membership certificate. This creates a problem for the identity escrow encryption: in the bilinear group G DDH is not hard and so Cramer-Shoup is not available; but encrypting in the target group G_T (as Camenisch and Lysyanskaya do in their group signature scheme [13]) would greatly increase the length of the group signature, since elements of G_T have much longer representation than those of G. Boneh, Boyen, and Shacham propose to solve this problem by using Linear encryption in G for the identity escrow.

Boneh, Boyen, and Shacham prove their scheme secure (in the random oracle model) in a relaxation of the group signature model of Bellare, Micciancio, and Warinschi [3]. The relaxation is necessary since Linear encryption, like ElGamal encryption, is only CPA secure, so only "CPA-full-anonymity" and not the Bellare-Micciancio-Warinschi notion of "full-anonymity" is achieved. We observe that if the Linear encryption in the Boneh-Boyen-Shacham group signature is replaced with either Kiltz's tag-encryption scheme under the CHK transformation[7] or our LCS scheme (and the NIZK is modified appropriately) the result is a group signature that is provably secure in the full Bellare-Micciancio-Warinschi model, solving an open problem of Boneh, Boyen, and Shacham's. If LCS is used, the signatures remain quite short: 1765 bits at the 80-bit security level rather than 1443 bits for the CPA-fully-anonymous version.

[5] Using Barreto-Naehrig [2] or Freeman [20] curves.

[6] More precisely, the bilinear group $G < E(\mathbb{F}_q)$ should be 512 bits and the pairing target group $G_T < \mathbb{F}_{q^k}^{\times}$ should be approximately 15,000 bits [26, 25]. The best known curve choice for these parameters is that of Cocks and Pinch [35, 21], with 1024-bit finite field $\mathbb{F}_q$ and embedding degree $k = 15$. Elements of G then have 1024-bit representations.

[7] The Boneh-Katz variant cannot be used since no efficient NIZK is known for proving that the MAC was correctly computed.

4 Generalizations of the Linear Assumption

For a constant $k \geq 1$, letting G be as above, and letting letting $g_1, g_2, \ldots, g_k$ and g_0 be arbitrary generators of G, consider the following problem:

k-Linear Problem in G: Given $g_1, g_2, \ldots, g_k, g_0, g_1^{r_1}, g_2^{r_2}, \ldots, g_k^{r_k}, g_0^{r_0} \in G$ as input, output `yes` if $r_0 = \sum_{i=1}^{k} r_i$ and `no` otherwise.

For notational convenience, we refer to the k-Linear problem as L_k. Observe that L_1 is DDH and L_2 is the Linear problem of Section 2.1. For each k the L_k problem defines a language (that is a subset of G^{2k+2}) in the obvious way. Alternatively, for each k and for any fixed choice of generators $g_1, \ldots, g_k, g_0$, we can define a language that is a subset of G^{k+1}; this agrees with the languages defined in Appendix A. As we did with the Linear assumption, we define the advantage of an algorithm $\mathcal{A}$ in deciding L_k in G as

$$\mathbf{Adv}_{\mathcal{A}}^{k\text{-linear}} \stackrel{\mathrm{def}}{=} \left| \begin{array}{l} \Pr\left[\begin{array}{c} \mathcal{A}\Big(g_1, \ldots, g_k, g_0, g_1^{r_1}, \ldots, g_k^{r_k}, g_0^{\sum_{i=1}^{k} r_k}\Big) = \texttt{yes} : \\ g_1, \ldots, g_k, g_0 \xleftarrow{\mathrm{R}} G, r_1, \ldots, r_k \xleftarrow{\mathrm{R}} \mathbb{Z}_p \end{array} \right] \\ \quad - \Pr\left[\begin{array}{c} \mathcal{A}\big(g_1, \ldots, g_k, g_0, g_1^{r_1}, \ldots, g_k^{r_k}, \eta\big) = \texttt{yes} : \\ g_1, \ldots, g_k, g_0, \eta \xleftarrow{\mathrm{R}} G, r_1, \ldots, r_k \xleftarrow{\mathrm{R}} \mathbb{Z}_p \end{array} \right] \end{array} \right| ,$$

with the probability taken over the uniform random choice of the parameters to $\mathcal{A}$ and over the coin tosses of $\mathcal{A}$. We say that an algorithm $\mathcal{A}$ (t, ϵ)-decides L_k in G if $\mathcal{A}$ runs in time at most t, and $\mathbf{Adv}_{\mathcal{A}}^{k\text{-linear}}$ is at least ϵ.

Definition 4.1. We say that the (t, ϵ)-L_k assumption holds in G if no algorithm (t, ϵ) decides L_k in G.

Relationships Between L_k Problems. We give evidence that the L_k family is a family of progressively harder problems. Specifically, we prove two theorems, whose informal statements are:

Theorem 4.2. *If L_{k+1} is easy, then L_k is easy.*

Theorem 4.3. *In a generic group L_{k+1} is hard even if L_k is easy.*

These results can be viewed as generalizations of the observations about the relationship of DDH and Linear given by Boneh, Boyen, and Shacham [6]. Taken together, Theorems 4.2 and 4.3 imply the following relationship amongst the assumptions, in generic groups at least:

$$\mathrm{L}_1 \gneqq \mathrm{L}_2 \gneqq \mathrm{L}_3 \gneqq \cdots \gneqq \mathrm{L}_k \gneqq \mathrm{L}_{k+1} \gneqq \cdots$$

We stress again that as k increases the assumptions become progressively *weaker*. (By contrast, the computational problems associated with each L_k are all equivalent to each other and, in particular, to CDH.)

Formal statements and proofs for Theorems 4.2 and 4.3 are given in Appendix B.

Since the L_k assumptions become progressively weaker as k increases, one can use the following strategy as a hedge against the development of algorithms capable of solving DDH or Linear:[8]

[8] The pairing gives one such algorithm for DDH, as observed by Joux and Nguyen [23]; its first uses in cryptography were destructive, breaking schemes based on DDH in groups where it (the pairing) is efficiently computable [29].

generalize a cryptosystem so that instances of it can be proved secure based on L_k for each k, and then deploy the instantiation based on, say, L_3 or L_4. Below, we show how Cramer-Shoup and Linear Cramer-Shoup generalize to L_k. Beyond that, it is quite simple to follow this strategy also for the DDH-based PRF of Naor and Reingold [30]; and the strategy should also be useful for many other DDH-based cryptosystems.

4.1 Cramer-Shoup from Generalized Linear

We describe a family of encryption schemes based on L_k. For each k, the k-CS encryption scheme is secure in G assuming L_k is hard in G. We stress that no additional structure is required of the group G — in particular, we assume nothing about the existence or nonexistence of a bilinear or k-multilinear map.

For $k = 1$, our k-CS scheme is just the original Cramer-Shoup; for $k = 2$, it is the Linear Cramer-Shoup of Section 3.

For each k, let $\mathcal{HF}_k$ be a family of universal one-way hash functions from G^{k+2} to $\mathbb{Z}_p$.

k-**CS.Kg.** Choose random generators $g_1, \ldots, g_k,\ g_0 \xleftarrow{\mathrm{R}} G$ and exponents

$$x_1, \ldots, x_k, x_0,\ y_1, \ldots, y_k, y_0,\ z_1, \ldots, z_k, z_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_p$$

and set

$$\begin{array}{lll} c_1 \leftarrow g_1^{x_1} g_0^{x_0} & d_1 \leftarrow g_1^{y_1} g_0^{y_0} & h_1 \leftarrow g_1^{z_1} g_0^{z_0} \\ c_2 \leftarrow g_2^{x_2} g_0^{x_0} & d_2 \leftarrow g_2^{y_2} g_0^{y_0} & h_2 \leftarrow g_2^{z_2} g_0^{z_0} \\ \vdots & \vdots & \vdots \\ c_k \leftarrow g_k^{x_k} g_0^{x_0} & d_k \leftarrow g_k^{y_k} g_0^{y_0} & h_k \leftarrow g_k^{z_k} g_0^{z_0} \ . \end{array}$$

In addition, choose a UOWHF $H \xleftarrow{\mathrm{R}} \mathcal{HF}_k$. The public key is

$$pk = (g_1, \ldots, g_k, g_0,\ c_1, \ldots, c_k,\ d_1, \ldots, d_k,\ h_1, \ldots, h_k, H) \in G^{4k+1} \times \mathcal{HF}_k \ ;$$

the secret key is

$$sk = (x_1, \ldots, x_k, x_0,\ y_1, \ldots, y_k, y_0,\ z_1, \ldots, z_k, z_0) \in \mathbb{Z}_p^{3k+3} \ .$$

k-**CS.Enc**(pk, M)**.** To encrypt a message $M \in G$, parse pk as $pk = (g_1, \ldots, g_k, g_0,\ c_1, \ldots, c_k,$ $d_1, \ldots, d_k,\ h_1, \ldots, h_k, H) \in G^{4k+1} \times \mathcal{HF}_k$. Choose random exponents $r_1, \ldots, r_k \xleftarrow{\mathrm{R}} \mathbb{Z}_p$ and set

$$u_1 \leftarrow g_1^{r_1} \qquad \cdots \qquad u_k \leftarrow g_k^{r_k} \qquad \text{and} \qquad u_0 \leftarrow g_0^{\sum_{i=1}^k r_i} \qquad \text{and} \qquad e \leftarrow M \cdot \prod_{i=1}^k h_i^{r_i} \ ;$$

now compute $\alpha \leftarrow H(u_1, \ldots, u_k, u_0, e)$ and, finally, $v \leftarrow \prod_{i=1}^k (c_i d_i^\alpha)^{r_i}$. The ciphertext is $ct = (u_1, \ldots, u_k, u_0, e, v) \in G^{k+3}$.

k-**CS.Dec**(pk, sk, ct). Parse the public key pk as $pk = (g_1, \ldots, g_k, g_0,\, c_1, \ldots, c_k,\, d_1, \ldots, d_k,\, h_1, \ldots, h_k, H) \in G^{4k+1} \times \mathcal{HF}_k$, the private key sk as $sk = (x_1, \ldots, x_k, x_0,\, y_1, \ldots, y_k, y_0,\, z_1, \ldots, z_k, z_0) \in \mathbb{Z}_p^{3k+3}$, and the ciphertext ct as $ct = (u_1, \ldots, u_k, u_0, e, v) \in G^{k+3}$. Compute $\alpha \leftarrow H(u_1, \ldots, u_k, u_0, e)$ and test that

$$\Big(\prod_{i=1}^{k} u_i^{x_i+\alpha y_i}\Big) \cdot u_0^{x_0+\alpha y_0} \stackrel{?}{=} v \tag{4}$$

holds. If it does not, output "`reject`". Otherwise, compute and output $M \leftarrow e \Big/ \big((\prod_{i=1}^{k} u_i^{z_i}) \cdot u_0^{z_0}\big)$.

Correctness. We show that the k-CS scheme is correct in Appendix C.

Security. The k-CS scheme is CCA secure if L_k is hard. We note again that the proof makes no assumption about the hardness of L_{k-1} and does not require either the existence or the nonexistence of a bilinear or a k-multilinear map.

Theorem 4.4. *The k-CS scheme is secure in the CCA sense if $\mathcal{HF}_k$ a secure UOWHF family and the L_k assumption holds in G.*

The proof is quite similar to the proof of Theorem 3.1. In Appendix C we provide a proof sketch that highlights the differences between the two.

Acknowledgments

The authors thank Moni Naor, Adam Smith, and Brent Waters for helpful discussions regarding this work, and Dan Boneh for suggesting the L_k assumption family.

References

[1] M. Abe, R. Gennaro, K. Kurosawa, and V. Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 128–46. Springer-Verlag, May 2005.

[2] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. Tavares, editors, *Proceedings of SAC 2005*, volume 3897 of *LNCS*, pages 319–31. Springer-Verlag, 2006.

[3] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 614–29. Springer-Verlag, May 2003.

[4] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer-Verlag, May 2004.

[5] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, May 2004.

[6] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 41–55. Springer-Verlag, Aug. 2004.

[7] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Computing*, 36(5):1301–28, Dec. 2006.

[8] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. Extended abstract in *Proceedings of Crypto 2001*.

[9] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In A. J. Menezes, editor, *Proceedings of CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103. Springer-Verlag, Feb. 2005.

[10] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *J. Cryptology*, 17(4):297–319, Sept. 2004. Extended abstract in *Proceedings of Asiacrypt 2001*.

[11] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. In C. G. Melles, J.-P. Brasselet, G. Kennedy, K. Lauter, and L. McEwan, editors, *Topics in Algebraic and Noncommutative Geometry: Proceedings in Memory of Ruth Michler*, volume 324 of *Contemporary Mathematics*, pages 71–90. American Mathematical Society, 2003.

[12] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In V. Atluri, C. Meadows, and A. Juels, editors, *Proceedings of CCS 2005*, pages 320–329. ACM Press, Nov. 2005.

[13] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 56–72. Springer-Verlag, Aug. 2004.

[14] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). In B. Kaliski, Jr., editor, *Proceedings of Crypto 1997*, volume 1294 of *LNCS*, pages 410–24. Springer-Verlag, Aug. 1997.

[15] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–22. Springer-Verlag, May 2004.

[16] R. Cramer and V. Shoup. A practical public key encryption system provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *Proceedings of Crypto 1998*, volume 1642 of *LNCS*, pages 13–25. Springer-Verlag, Aug. 1998.

[17] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Info. & System Security*, 3(3):161–85, 2000.

[18] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, May 2002.

[19] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Computing*, 33(1):167–226, 2003.

[20] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In F. Hess, S. Pauli, and M. Pohst, editors, *Proceedings of ANTS VII*, volume 4076 of *LNCS*, pages 452–65. Springer-Verlag, July 2006.

[21] S. Galbraith. Pairings. In I. F. Blake, G. Seroussi, and N. Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pages 183–213. Cambridge University Press, 2005.

[22] R. Gennaro, H. Krawczyk, and T. Rabin. Secure hashed diffie-hellman over non-DDH groups. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 361–81. Springer-Verlag, May 2004.

[23] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from computational Diffie-Hellman in cryptographic groups. *J. Cryptology*, 16(4):239–47, Sept. 2003.

[24] E. Kiltz. Chosen-ciphertext security from tag-based encryption. In S. Halevi and T. Rabin, editors, *Proceedings of TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer-Verlag, Mar. 2006.

[25] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In N. Smart, editor, *Proceedings of Cryptography and Coding 2005*, volume 3796 of *LNCS*, pages 13–36. Springer-Verlag, Dec. 2005.

[26] A. Lenstra. Unbelievable security: Matching AES security using public key systems. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 67–86. Springer-Verlag, Dec. 2001.

[27] P. MacKenzie, M. Reiter, and K. Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In M. Naor, editor, *Proceedings of TCC 2004*, volume 2951 of *LNCS*, pages 171–90. Springer-Verlag, Feb. 2004.

[28] A. Menezes, T. Okamoto, and P. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Info. Th.*, 39(5):1639–46, 1993.

[29] V. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–61, Sept. 2004.

[30] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–62, Mar. 2004.

[31] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In D. S. Johnson, editor, *Proceedings of STOC 1989*, pages 33–43. ACM Press, May 1989.

[32] T. Okamoto and D. Pointcheval. The gap problems: A new class of problems for the security of cryptographic primitives. In K. Kim, editor, *Proceedings of PKC 2001*, volume 1992 of *LNCS*, pages 104–18. Springer-Verlag, Feb. 2001.

[33] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 223–38. Springer-Verlag, May 1999.

[34] K. Paterson. Cryptography from pairings. In I. F. Blake, G. Seroussi, and N. Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter X, pages 215–51. Cambridge University Press, 2005.

[35] M. Scott. Scaling security in pairing-based protocols. Cryptology ePrint Archive, Report 2005/139, 2005. `http://eprint.iacr.org/`.

[36] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 256–66. Springer-Verlag, May 1997.

[37] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Proceedings of Eurocrypt 2000*, volume 1807 of *LNCS*, pages 275–88. Springer Verlag, May 2000.

[38] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 114–27. Springer-Verlag, May 2005.

A Projective Hashing from the Linear Language

It is also possible to analyze the LCS scheme using the universal hash proof paradigm of [18].

Let g_1, g_2, and g_3 be randomly chosen elements of G. In Figure 1 we define the projective hash for the Linear language for (g_1, g_2, g_3). Letting $k = (z_1, z_2, z_3)$ and $(u_1, u_2) = \alpha(k) = (g_1^{z_1} g_3^{z_3}, g_2^{z_2} g_3^{z_3})$, we observe that for $x \in L$ we have $x = (u_1, u_2, u_3) = (g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2})$ and so

$$H_k(x) = g_1^{r_1 z_1} g_2^{r_2 z_2} g_3^{(r_1+r_2)(z_3)} = (g_1^{z_1} g_3^{z_3})^{r_1} (g_2^{z_2} g_3^{z_3})^{r_2} = h_1^{r_1} h_2^{r_2} \ ,$$

so $H_k(x)$ can be calculated using only $\alpha(k)$ and the witness $w = (r_1, r_2)$, as required. On the other hand, for any $x \neq L$, $s \in S$, and $\pi \in \Pi$ it is easy to see that

$$\Pr_k[H_k(x) = \pi \mid \alpha(k) = s] = 1/p \ .$$

Thus $\mathbf{H} = (H, K, L, \Pi, S, \alpha)$ is a $(1/p)$-universal projective hash family.

Now, let H_{CR} be a collision-resistant hash function. In Figure 2 we define an extended projective hash for the Linear language. Letting $k = (x_1, x_2, x_3, y_1, y_2, y_3)$ and $(c_1, c_2, d_1, d_2) = \hat{\alpha}(k)$, we observe as before that for $x \in \hat{L}$ we have $x = (u_1, u_2, u_3, e) = (g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2}, e)$ and so

$$\hat{H}_k(x) = g_1^{(r_1)(x_1+ty_1)} g_2^{(r_2)(x_2+ty_2)} g_3^{(r_1+r_2)(x_3+ty_3)} = (c_1 d_1^t)^{r_1} (c_2 d_2^t)^{r_2} \ ,$$

where $t = H_{\mathrm{CR}}(u_1, u_2, u_3, e)$, and so $\hat{H}_k(x)$ can be calculated using only $\hat{\alpha}(k)$ and the witness $w = (r_1, r_2)$, as required. However, the analysis in the Theorem above shows that for any $x, x^* \neq L$, $s \in S$, and $\pi, \pi^* \in \Pi$ such that $x \neq x^*$ and $H_{\mathrm{CR}}(x) \neq H_{\mathrm{CR}}(x^*)$ we have

$$\Pr_k[\hat{H}_k(x) = \pi \mid \hat{H}_k(x^*) = \pi^* \wedge \alpha(k) = s] = 1/p \ .$$

$$K\colon \mathbb{Z}_p^3 \qquad X\colon G^3 \qquad \Pi\colon G \qquad S\colon G^2$$
$$L\colon \{(g_1^{r_1}, g_2^{r_2}, g_3^{r_3} \mid r_1, r_2 \in \mathbb{Z}_p\}$$
$$\alpha\colon K \to S; \quad (z_1, z_2, z_3) \mapsto (g_1^{z_1} g_3^{z_3}, g_2^{z_2} g_3^{z_3})$$
$$H\colon K \times X \to \Pi; \quad \big((z_1, z_2, z_3), (u_1, u_2, u_3)\big) \mapsto u_1^{z_1} u_2^{z_2} u_3^{z_3}$$

Figure 1: Universal projective hash family for the Linear language.

$$\hat{K}\colon \mathbb{Z}_p^6 \qquad \hat{X}\colon X \times \Pi \qquad \hat{\Pi}\colon G \qquad \hat{S}\colon G^4 \qquad \hat{L}\colon L \times \Pi$$
$$\hat{\alpha}\colon \hat{K} \to \hat{S}; \quad (x_1, x_2, x_3,\ y_1, y_2, y_3) \mapsto (g_1^{x_1} g_3^{x_3}, g_2^{x_2} g_3^{x_3}, g_1^{y_1} g_3^{y_3}, g_2^{y_2} g_3^{y_3})$$
$$\hat{H}\colon \hat{K} \times \hat{X} \to \hat{\Pi}; \quad \big((x_1, x_2, x_3,\ y_1, y_2, y_3),\ (u_1, u_2, u_3, e)\big) \mapsto u_1^{x_1+ty_1} u_2^{x_2+ty_2} u_3^{x_3+ty_3}$$
$$\text{where } t = H_{\mathrm{CR}}(u_1, u_2, u_3, e)$$

Figure 2: Universal_2 projective hash family for the Linear language.

Thus $\hat{\mathbf{H}} = (\hat{H}, \hat{K}, \hat{L}, \hat{\Pi}, \hat{S}, \hat{\alpha})$ is a $(1/p)$-universal_2 projective hash family provided H_{CR} is collision resistant. The generic construction of [18], instantiated with $\mathbf{H}$ and $\hat{\mathbf{H}}$, yields the LCS scheme described above.

B Generic Group Results for Generalized Linear

Theorem 4.2 is trivial to prove. Given an L_{k+1} solver $\mathcal{A}$ and an L_k instance

$$U = (g_1, \ldots, g_k,\ g_0,\ g_1^{r_1}, \ldots, g_k^{r_k},\ \eta)\ ,$$

choose $g_{k+1} \xleftarrow{\mathrm{R}} G$ and $r_{k+1} \xleftarrow{\mathrm{R}} \mathbb{Z}_p$, and set $\eta' \leftarrow \eta \cdot g_0^{r_{k+1}}$. Then

$$U' = (g_1, \ldots, g_k, g_{k+1},\ g_0,\ g_1^{r_1}, \ldots, g_k^{r_k}, g_{k+1}^{r_{k+1}},\ \eta')$$

is a uniformly distributed L_{k+1}-language element when U is a uniformly distributed L_k-language element, and a uniformly random element of $G^{2(k+1)}$ when U is a uniformly random element of G^{2k}. Thus we can run $\mathcal{A}$ on input U' and return whatever it returns.

To prove Theorem 4.3 we in fact prove a stronger result by means of multilinear maps [11]. For our purposes, a k-multilinear map is an efficiently computable map $e_k\colon G^k \to G_T$ such that $e_k(u_1^{a_1}, \ldots, u_k^{a_k}) = e_k(u_1, \ldots, u_k)^{\prod_{i=1}^k a_i}$ for all $u_1, \ldots, u_k \in G$ and $a_1, \ldots, a_k \in \mathbb{Z}_p$; and $e_k(g, \ldots, g) \neq 1$. Observe that a 2-multilinear map is just a bilinear map.

Lemma B.1. *Given a $k+1$-multilinear map there is an efficient algorithm for deciding L_k.*

Proof. The algorithm is simply this: Given an instance

$$g_1, \ldots, g_k,\ g_0,\ g_1^{r_1}, \ldots, g_k^{r_k},\ \eta\ ,$$

output "yes" if

$$e_{k+1}(g_1,\dots,g_k,\eta) \stackrel{?}{=} \prod_{i=1}^{k} e_{k+1}\big(g_1,\dots,g_{i-1},g_i^{r_i},g_{i+1},\dots,g_k,\,g_0\big) \tag{5}$$

and "no" otherwise. This is correct because

$$\begin{aligned}\prod_{i=1}^{k} e_{k+1}\big(g_1,\dots,g_{i-1},g_i^{r_i},g_{i+1},\dots,g_k,\,g_0\big) &= \prod_{i=1}^{k} e_{k+1}(g_1,\dots,g_k,\,g_0)^{r_i}\\ &= e_{k+1}(g_1,\dots,g_k,\,g_0)^{\sum_{i=1}^{k} g_i}\quad,\end{aligned}$$

and, letting $\eta = g_0^{r_0}$ for some r_0, we see that (5) holds exactly when r_0 equals $\sum_{i=1}^{k} r_i$, as required. $\square$

Lemma B.2. *If $\mathcal{A}$ solves L_k in the generic group model while making at most q oracle queries then its success probability is at most $k(q+2k+4)^2/p$.*

Proof. Let g be a generator of G, and let $x_1,\dots,x_k,\,y \stackrel{R}{\leftarrow} \mathbb{Z}_p$; we let $g_i = g^{x_i}$ for $1 \le i \le k$, and $g_0 = g^y$. Further, let $r_1,\dots,r_k,\,s \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $d \stackrel{R}{\leftarrow} \{0,1\}$. Now set $T_d \leftarrow g^{y\sum_{i=1}^{k} r_i}$ and $T_{1-d} \leftarrow g^s$. The adversary is given as input the opaque representations for the elements

$$g,\, g^{x_1},\dots,g^{x_k},\, g^y,\, g^{x_1 r_i},\dots,g^{x_k r_k},\, T_0, T_1\ ;$$

its goal is to guess the value of d.

Internally, algorithm $\mathcal{B}$ keeps track of elements handled by $\mathcal{A}$ as polynomials in the ring $\mathbb{Z}_p[X_1,\dots,X_k,Y,\,R_1,\dots,R_k,S]$. Externally, it describes these as arbitrary opaque strings in some sufficiently large domain. To keep track of these two representations, it maintains lists $\big\{(F_i,\xi_i)\big\}$ and $\big\{(F_{T,i},\xi_{T,i})\big\}$ for elements of G and G_T, respectively. Whenever $\mathcal{A}$ makes a query for some operation, $\mathcal{B}$ looks up the operands (as represented by $\mathcal{A}$ as bit strings) in the appropriate list to recover the internal representations. Whenever $\mathcal{B}$ must provide to $\mathcal{A}$ an element for the first time it creates for it a random external representation. If the domain for the external representation is large enough the following two events happen only with negligible probability: (1) algorithm $\mathcal{A}$ makes a query for an element other than those it obtained from $\mathcal{B}$; (2) algorithm $\mathcal{B}$ chooses the same opaque representation for two different elements.

The elements which $\mathcal{B}$ provides to $\mathcal{A}$ at the beginning of the game are represented internally by polynomials as follows:

$$\begin{array}{ccccc} g\colon F=1 & g_1\colon F=X_1 & \cdots & g_k\colon F=X_k & g_0\colon F=Y\\ g_1^{r_1}\colon F=X_1R_1 & \cdots & g_k^{r_k}\colon F=X_kR_k & T_0\colon F=T_0 & T_1\colon F=T_1\end{array}$$

Observe that each polynomial above has degree at most 2. Next, algorithm $\mathcal{B}$ allows $\mathcal{A}$ to perform operations on the elements to which it is given opaque representations, according to the following rules.

Group action. Given elements in G with internal representations F_1 and F_2, set $F' \leftarrow F_1+F_2$. Add F' to the G representation list if it is not already there, and respond with its external representation. The group action for G_T is handled analogously.

Inversion. Given an element in G with internal representation F, set $F' \leftarrow -F$. Add F' to the G representation list if it is not already there, and respond with its external representation. Inversion in G_T is handled analogously.

Multilinear map. Given elements in G with internal representations $F_1, \ldots, F_k$, set $F' \leftarrow \prod_{i=1}^k F_i$. Add F' to the G_T representation list if it is not already there, and respond with its external representation.

Observe that only the multilinear map operation produces as output an internal representation F' whose degree is greater than those of the inputs. Since the map can be applied only to elements of G and produces an element in G_T, and considering the degrees of the initial elements provided to $\mathcal{A}$, we obtain the following invariants: $\deg F \leq 2$ for all F on the G representation list; and $\deg F_T \leq 2k$ for all F_T on the G_T representation list.

Finally, $\mathcal{A}$ halts and outputs its guess d' for d. Now $\mathcal{B}$ chooses random values $x_1, \ldots, x_k$, y, $r_1, \ldots, r_k$, $s \xleftarrow{\text{R}} \mathbb{Z}_p$. Suppose we now set

$$X_1 \leftarrow x_1 \qquad \cdots \qquad X_k \leftarrow x_k \qquad Y \leftarrow y$$

$$R_1 \leftarrow r_1 \qquad \cdots \qquad R_k \leftarrow r_k \qquad T_d \leftarrow y \cdot \sum_{i=1}^k r_i \qquad T_{1-d} \colon s \ ;$$

the simulation engineered by algorithm $\mathcal{B}$ is consistent with these values unless there are two distinct polynomials F_1 and F_2 on the G representation list or two distinct polynomials $F_{T,1}$ and $F_{T,2}$ on the G_T representation list that take on the same value under the assignment above. In the remainder of the proof, we will first show that the adversary is unable to cause such a collision independent of the choice of random values and then bound the probability that a choice of random values causes a collision.

The crux of the first part of the argument is this. The values of the expressions substituted for the formal variables are all independent of each other except for that of T_d, which takes on the value $y \cdot \sum_{i=1}^k r_i$. Thus the adversary, to cause an independent collision, must produce from the other terms a polynomial that is a multiple of $Y \cdot \sum_{i=1}^k R_i$, say $F = AY \sum_{i=1}^k R_i$ for some nonzero A.

By inspecting the operations we perform on the internal representations of elements on the G and G_T lists and the elements with which the G list is initially populated, one can easily ascertain that any monomial produced divisible by R_i must also be divisible by X_i. Now, for each i, consider our polynomial $F = AY \sum_{j=1}^k R_j$. Each monomial in the expansion of AYR_i must be divisible by X_i, from which it follows that $X_i \mid A$.

(In more detail, the argument goes like this. Suppose $F = AY \sum_{j=1}^k R_j = (sX_i + t)(Y \sum_{j=1}^k R_j)$, where s and t are polynomials in $\mathbb{Z}_p[X_1, \ldots, X_k, Y, R_1, \ldots, R_k, S]$ and where X_i does not divide any monomial in t. Now multiply F out, and collect terms that differ only by integer coefficient. No term that derives from $sX_iY \sum_{j=1}^k R_j$ will cancel out or otherwise collect with a term that derives from $tY \sum_{j=1}^k R_j$, since all the former will all have an X_i component and none of the latter will. Moreover, some terms deriving from the expansion of $tY \sum_{j=1}^k R_j$ will have an R_i component, since R_i, as a formal variable, is not invertible in our polynomial ring. Next, consider the operations that we can perform: addition of polynomials representing elements in G, addition of polynomials representing elements in G_T, and products of k polynomials representing elements in G. It is clear that any G_T polynomial that can be formed by these operations can also be formed without using

G addition, by distributivity. Since the terms with which we begin are all monomials representing elements in G, it follows that every G_T polynomial we can produce can be expressed in a canonical form as $\sum_l c_i \cdot p_{l1} \cdots p_{lk}$, where c_i is a coefficient in $\mathbb{Z}$, each p_{lm} is a monomial from amongst the available monomials, and no two terms differ only in coefficient; the last criterion follows since such terms could be collected. What is more, given any G_T polynomial we can produce, if we express it in canonical form, then each of its terms can be computed by as the product $\cdot p_{l1} \cdots p_{lk}$ of G monomials that is then added to itself the appropriate number of times to give the coefficient c_i.[9] It follows, then, that in any G_T polynomial we can produce, when it is expressed in canonical representation, each of its terms can be produced independently of all the rest as the product of at most k starting G monomials, together with an integer coefficient. Finally, consider again the expansion of $F = (sX_i + t)(Y \sum_{j=1}^{k} R_j)$, canonically represented. We have seen that it contains at least one term, deriving from $tY \sum_{j=1}^{k} R_j$, that is divisible by R_i but not by X_i, and it therefore follows, from the argument made above about producible G_T polynomials, that this term is itself producible on its own. But this now means that we can form a monomial product, out of the terms with which we started, that is divisible by R_i but not by X_i; and this is plainly impossible since the only term with which we started that contains R_i is $X_i R_i$, which, if used, would introduce an X_i as well. It follows that $t = 0$ and $A = sX_i$, and therefore that $X_i \mid A$. Moreover, this argument clearly applies for each i.)

Thus each term of F must be divisible by the following $k + 2$ monomials: $X_1, \ldots, X_k$; Y; and R_i for some i. But forming such a term would require taking the product of at least $k + 1$ of the polynomials available to the adversary, since Y occurs only on its own and no term $X_a X_b$ occurs for any a, b. Since the multilinear map allows the adversary to compute only the product of at most k terms, we deduce that the adversary cannot synthesize any multiple of $Y \cdot \sum_{i=1}^{k} R_i$ thereby to cause a collision.

It remains only to bound the probability that a random choice of the values $x_1, \ldots, x_k$, y, $r_1, \ldots, r_k$, s will cause some two distinct polynomials to have the same value. All polynomials on the G representation list have degree at most 2, so any two such polynomials F_i and F_j are such that $F_i(\cdots) = F_j(\cdots)$ with probability at most $2/p$ over the choice of values. Similarly, all polynomials on the G_T representation list have degree at most $2k$, so any two such polynomials $F_{T,i}$ and $F_{T,j}$ are such that $F_{T,i}(\cdots) = F_{T,j}(\cdots)$ with probability at most $2k/p$ over the choice of values. The lists are populated initially with $2k + 4$ values. If the adversary makes q queries to its oracles then the lists contain at most $q + 2k + 4$ entries, so a sum over all pairs of entries gives a bound on the success probability of the adversary:

$$\epsilon \leq \binom{q + 2k + 4}{2} \frac{2k}{p} < \frac{k(q + 2k + 4)^2}{p} \quad ,$$

□

Lemmas B.1 and B.2, taken together, show that in generic groups featuring a k-multilinear map L_k is easy but L_{k+1} is hard, which proves Theorem 4.3.

[9] In fact, for any such term the choice of P_{lm} values is unique up to permutation by our choice of initial G monomials, but this is not important.

C Security of Generalized Linear Cramer-Shoup

Correctness. If the keys and encryption are generated according to the algorithms above, the test (4) in Dec will be satisfied, since we will then have

$$\begin{aligned}\Big(\prod_{i=1}^{k} u_i^{x_i+\alpha y_i}\Big)\cdot u_0^{x_0+\alpha y_0} &= \big(g_0^{x_0+\alpha y_0}\big)^{\sum_{i=1}^{k} r_i}\cdot\prod_{i=1}^{k}\big(g_i^{x_i+\alpha y_i}\big)_i^r \\ &= \prod_{i=1}^{k}\big((g_i^{x_i}g_0^{x_0})(g_i^{y_i}g_0^{y_0})^{\alpha}\big)_i^r \\ &= \prod_{i=1}^{k}(c_i d_i^{\alpha})_i^r = v \ ,\end{aligned}$$

as required. Next, the decryption will obtain the correct M, since

$$\begin{aligned} e\ /\ \big((\prod_{i=1}^{k} u_i^{z_i})\cdot u_0^{z_0}\big) &= e\ /\ \big(g_0^{(\sum_{i=1}^{k} r_i)(z_0)}\cdot\prod_{i=1}^{k} g_i^{r_i z_i}\big) \\ &= (e)\ /\ \big(\prod_{i=1}^{k}(g_i^{z_i}g_0^{z_0})^{r_i}\big) = \big(M\cdot\prod_{i=1}^{k} h_i^{r_i}\big)\ /\ \big(\prod_{i=1}^{k}(h_i)^{r_i}\big) = M \ .\end{aligned}$$

Security We present a sketch of the proof of Theorem 4.4. The proof is quite similar to the proof of Theorem 3.1, so we focus on those portions that are different.

Proof sketch of Theorem 4.4. The proof is quite similar to the proof of Theorem 3.1. Algorithm $\mathcal{B}$ is given as input an L_k-instance $(g_1,\ldots,g_k,g_0,\ u_1,\ldots,u_k,u_0)$; its goal is to decide whether this is an L_k tuple. It follows k-$\mathsf{CS.Kg}$ in setting up the public and private keys; the public key it provides to the adversary, algorithm $\mathcal{A}$. It answers $\mathcal{A}$'s decryption queries by following k-$\mathsf{CS.Dec}(pk, sk, \cdot)$. When $\mathcal{A}$ submits the messages M_0 and M_1 on which it wishes to be challenged, $\mathcal{B}$ chooses $b \xleftarrow{\mathrm{R}} \{0,1\}$ and sets

$$e \leftarrow M_b\cdot\big(\prod_{i=1}^{k} u_i^{z_i}\big)\cdot u_0^{z_0} \qquad \alpha\leftarrow H(u_1,\ldots,u_k,u_0,e) \qquad v\leftarrow\big(\prod_{i=1}^{k} u_i^{x_i+\alpha y_i}\big)\cdot u_0^{x_0+\alpha y_0} \ .$$

It then supplies to $\mathcal{A}$ the challenge ciphertext $ct^* = (u_1,\ldots,u_k,u_0,e,v)$. Algorithm $\mathcal{B}$ then responds to $\mathcal{A}$'s further decryption queries as before. Finally $\mathcal{A}$ outputs its guess b' for b. If $b = b'$, algorithm $\mathcal{B}$ outputs `yes`; otherwise it answers `no`.

The remainder of the proof establishes that if the input to $\mathcal{B}$ is a L_k tuple then $\mathcal{A}$ guesses b with nonnegligible advantage, as in the real distinguishing challenge, whereas if the input to $\mathcal{B}$ is a random tuple $\mathcal{A}$'s advantage is negligible.

In the first case, the input is such that $u_i = g_i^{r_i}$ for $1\le i\le k$, and $u_0 = g_i^{\sum_{i=1}^{k} r_i}$, where the r_i's are random in $\mathbb{Z}_p$ and unknown to $\mathcal{B}$. It is quite easy to see that the challenge ciphertext is formed and distributed exactly as in the distinguishing challenge, and that algorithm $\mathcal{B}$ answers decryption queries exactly as in the distinguishing challenge. (The formulas it uses are different than those specified in k-$\mathsf{CS.Dec}$ above, but these different formulas yield the same values, as shown in the correctness argument above.)

In the second case, we have $u_0 = g_i^{r_0}$ for some r_0 uniformly distributed in $\mathbb{Z}_p$. let "log(·)" stand for "$\log_g(\cdot)$" for some generator g of G and define, for $0\le i\le k$, $w_i = \log g_i$. Consider the elements $(z_1,\ldots,z_k,z_0)$ of the private key. The public key values $(h_!, h_k)$ constrain these to lie on the line at the intersection of the hyperplanes defined by $\{\log h_i = w_i z_i + w_3 z_3\}_{i=1}^{k}$. Now consider a decryption query for a valid ciphertext. Its first first $k+1$ components form a valid L_k tuple $(u_1', \ldots, u_k', u_0') = (g_1^{r_1'}, \ldots, g_k^{r_k'}, g_0^{\sum_{i=1}^{k} r_i'})$ for some $r_1', \ldots, r_k'$. From algorithm $\mathcal{B}$'s response, algorithm $\mathcal{A}$ will learn $(\prod_{i=1}^{k} u_i'^{z_i}) \cdot u_0'^{z_0}$; but since $(u_1', \ldots, u_k', u_0')$ is an L_k tuple we have $\log\bigl((u_1')^{z_1}(u_2')^{z_2}(u_3')^{z_3}\bigr) = \sum_{i=1}^{k}(r_i')(w_i z_i + w_0 z_0)$, which is linearly dependent on the line equations above and therefore gives $\mathcal{A}$ no new information.

Let the challenge ciphertext be $ct^* = (u_1, \ldots, u_k, u_0, e, v)$. The message M_b is blinded in e by the value $(\prod_{i=1}^{k} u_i^{z_i}) \cdot u_0^{z_0}$ whose discrete logarithm is

$$\log\Bigl((\prod_{i=1}^{k} u_i^{z_i}) \cdot u_0^{z_0}\Bigr) = (\sum_{i=1}^{k} r_i w_i z_i) + r_0 w_0 z_0 = (\Delta r)(w_0 z_0) + \sum_{i=1}^{k}(r_i)(w_i z_i + w_0 z_0) \ ,$$

where $\Delta r = r_0 - \sum_{i=1}^{k} r_i$ is nonzero with overwhelming probability. Thus to an adversary who has received decryption queries only for valid ciphertexts this value — and therefore b — is independent of its view.

We now show that, given that through query i the decryption oracle has not accepted an invalid ciphertext, the probability that it accepts an invalid one at query $i+1$ is negligible. Suppose that $\mathcal{A}$ submits a decryption query $(u_1', \ldots, u_k', u_0', e', v')$. Here $(u_1', u_2', u_3') = (g_1^{r_1'}, \ldots, g_k^{r_k'}, g_0^{r_0'})$, with $r_0' \neq \sum_{i=1}^{k} r_i'$. Let $\alpha = H(u_1, \ldots, u_k, u_0, e)$ and $\alpha' = H(u_1', \ldots, u_k', u_0', e')$. There are three possibilities:

Case 1. $(u_1, \ldots, u_k, u_0, e) = (u_1', \ldots, u_k', u_0', e')$, but $v \neq v'$. In this case, the decryption oracle will reject, since v as calculated in generating ct^* is the only correct checksum value for $(u_1, \ldots, u_k, u_0, e)$.

Case 2. $(u_1, \ldots, u_k, u_0, e) \neq (u_1', \ldots, u_k', u_0', e')$, yet $\alpha = \alpha'$. In this case, the adversary has generated a hash collision. We can reduce this case to a break of the UOWHF security of $\mathcal{HF}_k$ just as in the proof of Theorem 3.1.

Case 3. $(u_1, \ldots, u_k, u_0, e) \neq (u_1', \ldots, u_k', u_0', e')$, and $\alpha \neq \alpha'$.

In this third case, we ask: What is the probability, given the adversary's view, that v' is correctly chosen, so that the decryption algorithm accepts it? We can write the equation expressing this, along with the equations expressing the constraints imposed by $\{c_i\}_{i=1}^{k}$ and $\{d_i\}_{i=1}^{k}$ and the constraint on the value v in ct^* in matrix form as

$$\begin{pmatrix} \log c_1 \\ \vdots \\ \log c_k \\ \log d_1 \\ \vdots \\ \log d_k \\ \log v \\ \log v' \end{pmatrix} = \begin{pmatrix} w_1 & \cdots & 0 & w_0 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & w_k & w_0 & 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 & w_1 & \cdots & 0 & w_0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & w_k & w_0 \\ r_1 w_1 & \cdots & r_k w_k & r_0 w_0 & \alpha r_1 w_1 & \cdots & \alpha r_k w_k & \alpha r_0 w_0 \\ r_1' w_1 & \cdots & r_k' w_k & r_0' w_0 & \alpha' r_1' w_1 & \cdots & \alpha' r_k' w_k & \alpha' r_0' w_0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_k \\ x_0 \\ y_1 \\ \vdots \\ y_k \\ y_0 \end{pmatrix} .$$

What we wish to show is that the last line is independent of the others, so that the correct checksum v' is independent of the adversary's view. But, denoting the $(2k+2) \times (2k+2)$ matrix by M_k, we observe that

$$\det M_k = (-1)^{k+1}\big(w_0^2 \prod\nolimits_{i=1}^{k} w_i^2\big)\big(\alpha - \alpha'\big)\big(r_0 - \sum\nolimits_{i=1}^{k} r_i\big)\big(r'_0 - \sum\nolimits_{i=1}^{k} r'_i\big) \neq 0 \ ,$$

so the equations are indeed independent. $\square$