On the UC-(In)Security of PAKE Protocols in the Plain Model

Naman Kumar[*]
Oregon State University

Jiayu Xu[†]
Oregon State University

November 13, 2024

Abstract

We show that the PAKE protocol of Katz, Ostrovsky and Yung (2001) is not UC-secure in the plain model. We provide the first proof of UC-security of KOY in the AGM under the hardness of the Square Diffie-Hellman (SDH) assumption.

1 Introduction

1.1 Protocol Description

Jiayu: Insert the description of KOY (revised to fit the UC formality) here. Naman: I've made edits, clarified the scheme.

We provide a high-level description of the protocol along with a sketch of the security proof below. Let $\mathbb{G}$ be an algebraic group in which DDH is believed to be hard and define $\mathsf{crs} := (g_1, g_2, h, c, d)$ to be uniformly sampled from $\mathbb{G}^5$; we set this to be the common random string. Note that crs can be interpreted to be a degenerate Cramer-Shoup public key with an unknown secret key.

- User begins by generating a pair of keys $(\mathsf{VK}, \mathsf{SK})$ for a one-time signature scheme and encrypts its password pw with label VK using the Cramer-Shoup public key embedded in the CRS. Let H be a collision-resistant hash function and r_1 be the randomness used during encryption. The resulting ciphertext consists of four group elements $\mathsf{ct}_1 := (A, B, C, D) = (g_1^{r_1}, g_2^{r_1}, h^{r_1} \cdot \mathsf{pw}, (cd^\alpha)^{r_1})$ where $\alpha = H(\mathsf{VK}, A, B, C)$. User then sends $\mathsf{msg}_1 := (\mathsf{VK}, \mathsf{ct}_1)$ to Server. Note that A, B, $C' = C/\mathsf{pw}$, D are all of the form g^{r_1} where g is some group element that Server can compute, so they also serve as a message in a Diffie–Hellman-like protocol.

- Upon receiving $(\mathsf{VK}, A, B, C, D)$, the Server samples its 'Diffie–Hellman exponents' (x_2, y_2, z_2, w_2), and computes $E = g_1^{x_2} g_2^{y_2} h^{z_2} (cd^\alpha)^{w_2}$.[1] Furthermore, Server encrypts pw with label (msg_1, E) using Cramer–Shoup encryption as done by User in msg_1. Let r_2 be the randomness used in encryption. The resulting ciphertext is $\mathsf{ct}_2 := (F, G, I, J) = (g_1^{r_2}, g_2^{r_2}, h^{r_2} \cdot \mathsf{pw}, (cd^\beta)^{r_2})$. Server sends $\mathsf{msg}_2 = (E, \mathsf{ct}_2)$ to User.

---

[*]Email: `kumarnam@oregonstate.edu`

[†]Email: `xujiay@oregonstate.edu`

[1]Note that $\alpha = H(\mathsf{VK}|A|B|C)$, so Server must wait for User's message before proceeding.

- Symmetrically, User upon receiving (E, ct_2) samples its own 'Diffie–Hellman exponents' (x_1, y_1, z_1, w_1) and computes $K = g_1^{x_1} g_2^{y_1} h^{z_1} (cd^\beta)^{w_1}$. It signs the protocol transcript $\mathsf{st} = \mathsf{Sign}_{\mathsf{SK}}(\mathsf{msg}_1 | \mathsf{msg}_2 | K)$, and sends $\mathsf{msg}_3 = (K, \mathsf{st})$ to the server.

- Server verifies the signature st and aborts if it is invalid. Otherwise the session key sk is defined as the product of

$$X_1 = E^{r_1} = A^{x_2} B^{y_2} (C/\mathsf{pw})^{z_2} D^{w_2}$$

and

$$X_2 = K^{r_2} = F^{x_1} G^{y_1} (I/\mathsf{pw})^{z_1} J^{w_1}$$

User can compute X_1 as E^{r_1} and X_2 as $F^{x_1} G^{y_1} (I/\mathsf{pw})^{z_1} J^{w_1}$, whereas Server can compute X_1 as K^{r_2} and X_2 as $F^{x_1} G^{y_1} (I/\mathsf{pw})^{z_1} J^{w_1}$.

Security. To perform authentication, User and Server need to (implicitly) prove to each other that they know pw. This is achieved as follows. Note that $X_1 = E^{r_1} = A^{x_2} B^{y_2} (C/\mathsf{pw})^{z_2} D^{w_2}$ has the following property: given E (but not x_2, y_2, z_2, w_2), if $A|B|C|D$ is a valid encryption of pw, then knowing the randomness r_1 is sufficient for computing X_1; otherwise X_1 is a uniformly random group element.[2] Therefore, an adversarial user that does not know pw, is not able to come up with a valid $A|B|C|D$, so X_1 is uniformly random in the adversary's view (and so is $\mathsf{sk} = X_1 X_2$); a symmetric argument can be made for an adversarial server. In the man-in-the-middle setting, an adversary can attempt to generate a valid ciphertext after seeing another valid ciphertext from the honest user/server, so we need the encryption scheme to be non-malleable.

The one-time signature scheme effectively forces the adversary to pass msg_2 and msg_3 without modification as long as it passes msg_1. This is to prevent a man-in-the-middle adversary from gaining information by passing all ciphertexts but modifying the rest of the messages. Specifically, (if the signature scheme is removed) consider an adversary that passes $\mathsf{msg}_1 = A|B|C|D$, changes $\mathsf{msg}_2 = E|F|G|I|J$ to $E^{\frac{1}{2}}|F|G|I|J$, and changes $\mathsf{msg}_3 = K$ to K^2; this would cause $\mathsf{sk}_S = \mathsf{sk}_U^2$. In other words, the adversary (that does not know the password) causes the two parties' session keys to be unequal but correlated, which is not allowed by the security of PAKE. Furthermore, to prevent the adversary from plugging in its own verification key (and thus knowing the corresponding secret key), VK is included in the hash that produces α. In this way if the adversary changes VK while keeping the ciphertext $A|B|C|D$, the ciphertext would become invalid.

1.2 Technical Overview

In this section, we provide a high-level explanation of why the KOY protocol is insecure in the UC framework, and how the AGM circumvents the difficulty for the UC simulator.

UC-insecurity of KOY. Our attack relies on an adversary $\mathcal{A}$ that completely disregards the presence of Server and instead interacts with User while executing Server's algorithm on its own. In particular, once the protocol is initiated by User, $\mathcal{A}$ assumes the role of the server (discarding the actual server in the process, which plays no part in the protocol) and receives $\mathsf{msg}_1 = \mathsf{VK}|A|B|C|D$. After this, $\mathcal{A}$ runs the server's algorithm on User's password pw (i.e., we

---

[2]Using the terminology of SPHF: (x_2, y_2, z_2, w_2) is the hash key and E is the corresponding projection key; the function is defined as $\mathsf{Hash}_{(x_2, y_2, z_2, w_2)}(m) = A^{x_2} B^{y_2} (C/m)^{z_2} D^{w_2}$.

assume that $\mathcal{A}$ makes a correct password guess) and computes $\mathsf{msg}_2 = E|F|G|I|J$. User then runs its session-key generating algorithm and outputs its session key $\mathsf{sk} = E^{r_1}F^{x_1}G^{y_1}(I/\mathsf{pw})^{z_1}J^{w_1}$. At this point, $\mathcal{A}$ (and $\mathcal{Z}$) have all the information they need to run the server's session-key generating algorithm locally, which computes a session key equal to sk generated by User.

To see why a simulator $\mathcal{S}$ cannot simulate this adversary, we attempt an ideal-world execution and pinpoint where it fails. Since $\mathcal{S}$ is allowed to choose $\mathsf{crs} = (g_1, g_2, h, c, d)$, it can sample g_1 at random and set h such that $h = g_1^\ell$ — in other words, $\mathcal{S}$ chooses the "CRS trapdoor" $\ell = \log_{g_1} h$. After receiving the NewSession command from $\mathcal{F}_{\mathsf{PAKE}}$, $\mathcal{S}$ must simulate User's first message msg_1. Since $\mathcal{S}$ does not know the password, at this point it must (effectively) guess some pw' at random; that is, in msg_1, $C = h^{r_1} \cdot \mathsf{pw}'$ where pw' can be no better than a random password sampled from the dictionary. (C is indistinguishable from the correct value due to the security of Cramer–Shoup encryption.) After $\mathcal{Z}$ responds with $\mathsf{msg}_2 = E|F|G|I|J$, since $F = g_1^{r_2}$ and $I = h^{r_2} \cdot \mathsf{pw}$, $\mathcal{S}$ can extract pw as I/F^ℓ. Once this has been done, $\mathcal{S}$ can send a TestPwd command to $\mathcal{F}_{\mathsf{PAKE}}$ on the correct pw; $\mathcal{F}_{\mathsf{PAKE}}$ would mark the User session compromised and thus allow $\mathcal{S}$ to choose User's session key sk (which has to be consistent with the session key User computes in the real world).[3] This is where the game-based security and UC-security of PAKE diverge: in game-based security, all security guarantees are considered lost (and the simulation of the game can stop) once the adversary guesses the correct password; whereas in UC-security the simulation has to continue. The problem here is that *even knowing the correct password* pw, $\mathcal{S}$ *still cannot determine what* sk *should be.*

Recall that sk is the product of

$$X_1 = E^{r_1} = A^{x_2}B^{y_2}(C/\mathsf{pw})^{z_2}D^{w_2}$$

and

$$X_2 = K^{r_2} = F^{x_1}G^{y_1}(I/\mathsf{pw})^{z_1}J^{w_1}$$

Computing X_2 is not a problem for $\mathcal{S}$, since $\mathsf{msg}_2 = E|F|G|I|J$ is provided to $\mathcal{S}$ directly from the environment; $\mathcal{S}$ chose x_1, y_1, z_1, w_1 on its own; and $\mathcal{S}$ has extracted pw. However, $\mathcal{S}$ is not able to compute X_1. At first glance, computing $X_1 = E^{r_1}$ might appear feasible as $\mathcal{S}$ received E as part of msg_2 and sampled r_1 before sending msg_1. However, the problem is that *the password guess* pw' *that* $\mathcal{S}$ *uses while generating* msg_1 *is likely incorrect*; as a result, E^{r_1} that $\mathcal{S}$ computes is actually equal to $A^{x_2}B^{y_2}(C/\mathsf{pw}')^{z_2}D^{w_2}$, whereas the correct value should be $A^{x_2}B^{y_2}(C/\mathsf{pw})^{z_2}D^{w_2}$. This means that $\mathcal{S}$ must know $(\mathsf{pw}/\mathsf{pw}')^{z_2}$ in order to compute the correct sk, which is infeasible unless $\mathsf{pw}' = \mathsf{pw}$ (whose probability is $1/|\mathcal{D}|$).

Simulating the session key in AGM. Our next critical observation is that the session key in the above attack can be simulated if we resort to the AGM, as the simulator $\mathcal{S}$ can extract x_2, y_2, z_2, w_2 from an algebraic environment. In more detail, suppose $\mathcal{Z}$ runs the above attack and sends E as part of msg_2. At this point all group elements that $\mathcal{Z}$ has seen are g_1, g_2, h, c, d from crs, A, B, C, D from msg_1, and pw. An algebraic $\mathcal{Z}$ must "explain" how E is computed; for now let's ignore A, B, C, D, pw and assume $\mathcal{Z}$ computes

$$E = g_1^{x_2}g_2^{y_2}h^{z_2}(cd^\alpha)^{w_2}$$

---

[3]A TestPwd *must* be run, since we require that msg_1 and msg_2 together with the randomness of the User and $\mathcal{A}$ together determine sk; allowing the simulation to proceed without a TestPwd would result in $\mathcal{F}_{\mathsf{PAKE}}$ outputting a uniformly random key.

In the real world User would compute $X_1 = E^{r_1}$, which is equal to $A^{x_2} B^{y_2} (C/\mathsf{pw})^{z_2} D^{w_2}$. In the ideal world, as explained above, $\mathcal{S}$ cannot compute X_1 as E^{r_1} since it chose the wrong pw' with high probability while generating A, B, C, D; however, after extracting the correct pw from msg_2, $\mathcal{S}$ can still compute X_1 as $A^{x_2} B^{y_2} (C/\mathsf{pw})^{z_2} D^{w_2}$, since now it sees the algebraic coefficients x_2, y_2, z_2, w_2 from $\mathcal{Z}$. In this way $\mathcal{S}$ generates X_1 that is indistinguishable from the real world.

In the general case, suppose $\mathcal{Z}$ computes

$$E = g_1^{x_2} g_2^{y_2} h^{z_2} c^{w_2} d^{v_2} A^{x_2'} B^{y_2'} C^{z_2'} D^{w_2'} \mathsf{pw}^{p_2}$$

In the real world User would compute

$$
\begin{aligned}
X_1 &= E^{r_1} \\
&= g_1^{r_1 x_2} g_2^{r_1 y_2} h^{r_1 z_2} c^{r_1 w_2} d^{r_1 v_2} A^{r_1 x_2'} B^{r_1 y_2'} C^{r_1 z_2'} D^{r_1 w_2'} \mathsf{pw}^{r_1 p_2} \\
&= g_1^{r_1 x_2} g_2^{r_1 y_2} \left(\frac{C}{\mathsf{pw}} \right)^{z_2} c^{r_1 w_2} d^{r_1 v_2} A^{r_1 x_2'} B^{r_1 y_2'} C^{r_1 z_2'} D^{r_1 w_2'} \mathsf{pw}^{r_1 p_2}
\end{aligned}
$$

Again, in the ideal world $\mathcal{S}$ can compute X_1 according to the last equation; the key point is that given $A|B|C|D$, the only place where the "wrong" password pw' is used lies in h, so we only need to "correct" h^{r_1} from C/pw' to C/pw. A difference is that now the expression of X_1 involves the Cramer–Shoup randomness r_1, which is not a problem for $\mathcal{S}$ as $\mathcal{S}$ chose r_1 itself while sending msg_1. (Of course, we have $A = g_1^{r_1}$, so the $g_1^{r_1 x_2} A^{r_1 x_2'}$ part can be rewritten as $A^{x_2 + r_1 x_2'}$, and so on. But this simplification is not necessary.)

Further subtleties in UC. As we have just seen, one critical difference between game-based security and UC-security is that in UC indistinguishability between the real view and the simulated view must remain *even when the environment sees the key of a successfully attacked session*, whereas in the game-based setting the adversary simply wins (and the simulator can "give up" on simulating the session key) once the attack on a session is successful. While the session key itself can be simulated in the AGM, this poses another potential issue that is more subtle: after seeing the session key, the environment might go back and check the validity of previous protocol messages using this information.

In more detail, assume again that the adversary interacts with User by running Server's algorithm on User's password pw. When User's session completes, the environment $\mathcal{Z}$ sees User's session key $\mathsf{sk}_U = X_1 X_2$ and the last message $\mathsf{msg}_3 = K|\sigma$. Since $\mathcal{Z}$ can compute X_2 as K^{r_2} (since r_2 is chosen by $\mathcal{Z}$ itself), it can recover $X_1 = E^{r_1}$ (where r_1 is the randomness used in msg_1). Recall that in the ideal world the Cramer–Shoup ciphertext $A|B|C|D$ generated by the simulator $\mathcal{S}$ is an encryption of some pw' that is unlikely to be the "correct" pw, so $\mathcal{Z}$ must not be able to detect this fact even after seeing E^{r_1}. In other words, (very roughly) *Cramer–Shoup must be secure even if the adversary sees E^{r_1} for some E of its choice.*

Below we analyze two simple attacks using this strategy:

1. Say $\mathcal{Z}$ chooses $E = A = g_1^{r_1}$; then $E^{r_1} = g_1^{r_1^2}$. If 2-DL is easy in the group[4], then $\mathcal{Z}$ can recover r_1 after User's session completes, and check if $C = h^{r_1} \cdot \mathsf{pw}$. In the real world this equation holds, whereas in the ideal world it does not if $\mathcal{S}$ chose the "wrong" $\mathsf{pw}' \neq \mathsf{pw}$

---

[4]The 2-DL problem is: given (g^x, g^{x^2}) for $x \leftarrow \mathbb{Z}_q$, compute x. We do not know the relative hardness between 2-DL and DDH, and it has been shown that 2-DL and CDH are separate in the AGM [**?**].

to encrypt while simulating msg_1. In fact, it seems that for the KOY protocol to be UC-AGM-secure, we need the Jiayu: FILL IN assumption. Jiayu: I think the assumption should be Decisional Square Diffie–Hellman (DSDH), which says that given g^x, it is hard to distinguish g^{x^2} from random. Note that this implies 2-DL (given (g^x, Y) where Y is either g^{x^2} or random, the DSDH solver can feed (g^x, Y) to the 2-DL solver and distinguish by observing whether the 2-DL solver wins or not)

2. Say $\mathcal{Z}$ chooses $E = c$; then $E^{r_1} = c^{r_1}$. But Cramer–Shoup is obviously *not* CCA-secure if the adversary additionally sees c^{r_1}. What saves us here is that $\mathcal{Z}$ sees E^{r_1} *only at the end of* User*'s session.* Indeed, the reduction $\mathcal{R}$ to the security of Cramer–Shoup roughly works as follows:

    (a) $\mathcal{R}$ embeds the challenge ciphertext as msg_1, User's first message intercepted by $\mathcal{A}$;

    (b) When $\mathcal{A}$ sends msg_1^* to Server and msg_2^* to User, $\mathcal{R}$ needs to query the decryption oracle to extract the password guesses contained in these two messages;

    (c) Finally, if the password guess in msg_2^* is correct, $\mathcal{R}$ needs to simulate sk_U. (This step is not needed in the game-based proof.)[5]

    c^{r_1} is needed only in step (c), which happens after all decryption oracle queries have been made. Therefore, we only need Cramer–Shoup to remain CCA-secure if the adversary *is restricted to making all decryption oracle queries before learning c^{r_1}*. We will show that Cramer–Shoup indeed satisfies this property in Jiayu: FILL IN Jiayu: I think this should work, but it needs to be double checked

By inspecting the game-based security proof of the KOY protocol, one can see that the above attacks are essentially the only scenarios where the UC-security might be broken, and the security argument for all other cases (including the adversary sending a message that contains an incorrect password guess, or modifying the signature) is essentially identical to the game-based proof. Jiayu: I hope so...

2 Preliminaries

2.1 Assumptions

Let $\mathcal{G}$ be an algorithm that on input 1^λ uniformly samples from a sequence of group distributions $(S_\lambda)_{\lambda \in \mathbb{Z}_{\geq 0}}$ of group distributions of the form $\Gamma[\hat{\mathbb{G}}, \mathbb{G}, g, q] \in [S_\lambda]$, where $\hat{\mathbb{G}}$ is a finite multiplicative abelian group, $\mathbb{G}$ is a prime-order subgroup, g is a generator of $\mathbb{G}$ and q is the order of $\mathbb{G}$. We assume that group operations in $\mathbb{G}$ can be done in (expected) polynomial time in $\mathbb{G}$, including exponentiation, selecting a random group element, and membership.

Note that a random group element can be chosen by selecting $x \xleftarrow{\$} \mathbb{Z}_q$ and computing g^x. We define $\bar{\mathbb{G}}$ to be $\mathbb{G} \setminus \{1\}$; since q is prime, $\bar{\mathbb{G}}$ is the set of generators of $\mathbb{G}$. If $x \xleftarrow{\$} \bar{\mathbb{G}}$, then we obtain a random generator.

---

[5]We omit the remaining steps which are to simulate msg_3, and on msg_3^* simulate Server's session key sk_S, as they are inconsequential to our main point.

2.1.1 The DDH Assumption

Let $\mathcal{G}$ be as above. We define for each $\lambda \in \mathbb{Z}_{\geq 0}$ and for all $\Gamma[\hat{\mathbb{G}}, \mathbb{G}, g, q] \in [S_\lambda]$ the sets

$$\mathsf{RandDH}_{\lambda,\Gamma} := \{(g, g^x, g^y, g^z) : x, y \xleftarrow{\$} \mathbb{Z}_q^*, z \xleftarrow{\$} \mathbb{Z}_q\}$$

and

$$\mathsf{DH}_{\lambda,\Gamma} := \{(g, g^x, g^y, g^{xy}) : x, y \xleftarrow{\$} \mathbb{Z}_q^*\}.$$

Where $\mathsf{DH}_{\lambda,\Gamma}$ is the set of Diffie-Hellman Triples. For any probabilistic polynomial-time distinguishing algorithm $\mathcal{A}$ and for all $\lambda \in \mathbb{Z}_{\geq 0}$, we define the DDH advantage of $\mathcal{A}$ against $\mathcal{G}$ at λ given Γ as

$$\mathsf{AdvDDH}_{\mathcal{G},\mathcal{A}}(\lambda|\Gamma) = \left| \Pr[\mathcal{A}(1^\lambda, \Gamma, \rho) = 1 : \rho \xleftarrow{\$} \mathsf{RandDH}_{\lambda,\Gamma}] - \Pr[\mathcal{A}(1^\lambda, \Gamma, \rho) = 1 : \rho \xleftarrow{\$} \mathsf{DH}_{\lambda,\Gamma}] \right|$$

The Decisional Diffie-Hellman (DDH) assumption for $\mathcal{G}$ states that for every probabilistic, polynomial-time algorithm $\mathcal{A}$, there exists some negligible function negl such that for all $\lambda \in \mathbb{Z}_{\geq 0}$,

$$\mathsf{AdvDDH}_{\mathcal{G},\mathcal{A}}(\lambda|\Gamma) \leq \mathsf{negl}(\lambda)$$

where $\Gamma \xleftarrow{\$} [S_\lambda]$.

2.1.2 The SDH Assumption

Let $\mathcal{G}$ be as above. We define for each $\lambda \in \mathbb{Z}_{\geq 0}$ and for all $\Gamma[\hat{\mathbb{G}}, \mathbb{G}, g, q] \in [S_\lambda]$ the sets

$$\mathsf{RandSDH}_{\lambda,\Gamma} := \{(g, g^x, g^y) : x, y \xleftarrow{\$} \mathbb{Z}_q^*\}$$

and

$$\mathsf{SDH}_{\lambda,\Gamma} := \{(g, g^x, g^{x^2}) : x \xleftarrow{\$} \mathbb{Z}_q^*\}.$$

Where $\mathsf{SDH}_{\lambda,\Gamma}$ is the set of Square Diffie-Hellman Triples. For any probabilistic polynomial-time distinguishing algorithm $\mathcal{A}$ and for all $\lambda \in \mathbb{Z}_{\geq 0}$, we define the SDH advantage of $\mathcal{A}$ against $\mathcal{G}$ at λ given Γ as

$$\mathsf{AdvSDH}_{\mathcal{G},\mathcal{A}}(\lambda|\Gamma) = \left| \Pr[\mathcal{A}(1^\lambda, \Gamma, \rho) = 1 : \rho \xleftarrow{\$} \mathsf{RandSDH}_{\lambda,\Gamma}] - \Pr[\mathcal{A}(1^\lambda, \Gamma, \rho) = 1 : \rho \xleftarrow{\$} \mathsf{SDH}_{\lambda,\Gamma}] \right|$$

The Square Diffie-Hellman (DDH) assumption for $\mathcal{G}$ states that for every probabilistic, polynomial-time algorithm $\mathcal{A}$, there exists some negligible function negl such that for all $\lambda \in \mathbb{Z}_{\geq 0}$,

$$\mathsf{AdvSDH}_{\mathcal{G},\mathcal{A}}(\lambda|\Gamma) \leq \mathsf{negl}(\lambda)$$

where $\Gamma \xleftarrow{\$} [S_\lambda]$.

2.2 Cramer-Shoup Encryption Scheme

2.3 UC-PAKE

We define the UC-PAKE functionality below.

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

$$\Gamma[\hat{\mathbb{G}}, \mathbb{G}, g, q] \xleftarrow{\$} \mathcal{G}(1^\lambda); \mathsf{hk} \xleftarrow{\$} \mathsf{HF.Keyspace}_{\lambda,\Gamma}$$

$$w \xleftarrow{\$} \mathbb{Z}_q^*; x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_q$$

$$g_1 \leftarrow g; g_2 \leftarrow g^w; c = g_1^{x_1} g_2^{x_2}; d \leftarrow g_1^{y_1} g_2^{y_2}; h \leftarrow g_1^{z_1} g_2^{z_2}$$

and output the public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, g_1, g_2, c, d, h)$ and the secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2)$.

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

$$\mathsf{PK} = (\Gamma, \mathsf{hk}, g_1, g_2, c, d, h) \in [\mathcal{G}_\lambda] \times [\mathsf{HF.Keyspace}_{\lambda,\Gamma}] \times \mathbb{G}^4,$$

a message $m \in G$ and a label label, compute

    E1: $r \xleftarrow{\$} \mathbb{Z}_q$;

    E2: $u_1 \xleftarrow{\$} g_1^r$;

    E3: $u_2 \xleftarrow{\$} g_2^r$;

    E4: $e' \leftarrow h^r$;

    E5: $e \leftarrow e' \cdot m$;

    E6: $\alpha \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\mathsf{label}, u_1, u_2, e)$;

    E7: $v \leftarrow (cd^\alpha)^r$;

and output the ciphertext $\psi = (u_1, u_2, e, v)$.

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

$$\mathsf{SK} = (\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2) \in [\mathcal{G}_\lambda] \times [\mathsf{HF.Keyspace}_{\lambda,\Gamma}] \times \mathbb{Z}_q^6,$$

along with a ciphertext ψ and a label label, do the following.

    D1: Parse ψ as a 4-tuple $(u_1, u_2, e, v) \in \mathbb{G}^4$, output reject and halt if ψ is not of this form.

    D2: Test if u_1, u_2 and e belong to $\mathbb{G}$; output reject and halt if this is not the case.

    D3: Compute $v' \leftarrow \mathsf{HF}_{\mathsf{hk}}^{\lambda,\Gamma}(\mathsf{label}, u_1, u_2, e)$.

    D4: Test if $v = u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha}$; output reject and halt if this is not the case.

    D5: Compute $e' = u_1^{z_1} u_2^{z_2}$.

    D6: Compute $m \leftarrow e \cdot e'^{-1}$ and output m.

Figure 1: The Cramer-Shoup Encryption Scheme with Labels.

3 Proof of UC-Insecurity

Theorem 3.1. *Assuming the hardness of fixed-CDH, the protocol of [KOY] does not UC-realize $\mathcal{F}_{\mathsf{pake}}$ in the $\mathcal{F}_{\mathsf{crs}}$-hybrid model.*

Proof. Consider the environment $\mathcal{Z}$ in Figure 2 and the dummy adversary. It follows from the correctness of the protocol that in the real-world protocol execution $\mathcal{Z}$ always outputs 1, since the algorithm of $\mathcal{Z}$ and $\mathcal{A}$ is the same as that of an honest server. At a high level, we will show that any simulator that successfully simulates the protocol against $\mathcal{Z}$ in the ideal world can be used to solve arbitrary instances of fixed-CDH.

<u>Environment $\mathcal{Z}$:</u>

0. $\mathcal{Z}$ receives the CRS (g_1, g_2, h, c, d).

1. $\mathcal{Z}$ selects $\mathsf{pw} \overset{\$}{\leftarrow} \mathcal{PW}$, where $\mathcal{PW} \subseteq \mathbb{G}$ is the password dictionary. It then sends $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{User}, \mathsf{Server}, \mathsf{pw})$ to User.

2. $\mathcal{Z}$ receives $\mathsf{msg}_1 = \mathsf{sid}|\mathsf{VK}|A|B|C|D$ from $\mathcal{A}$ and samples $x_2, y_2, z_2, w_2, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_q$. It then sets

$$
\begin{aligned}
\alpha' &:= H(A|B|C|D) \\
E &:= g_1^{x_2} g_2^{y_2} h^{z_2} (c d^{\alpha'})^{w_2} \\
F &:= g_1^{r_2} \\
G &:= g_2^{r_2} \\
I &:= h^{r_2} \cdot \mathsf{pw} \\
\beta &:= H(\mathsf{msg}_1|E|F|G|I) \\
J &:= (c d^{\beta})^{w_2}
\end{aligned}
$$

   and instructs $\mathcal{A}$ to send $\mathsf{msg}_2 := \mathsf{sid}|E|F|G|I|J$ to User.

3. $\mathcal{Z}$ receives $\mathsf{msg}_3 = \mathsf{sid}|K|\sigma$ from $\mathcal{A}$ and $(\mathsf{sid}, \mathsf{sk})$ from User.

4. $\mathcal{Z}$ sets $C' := C/\mathsf{pw}$ and then checks if $\mathsf{Vrfy}_{\mathsf{VK}}(\mathsf{msg}_1|\mathsf{msg}_2|K, \sigma) = 1$. If yes, it computes $\mathsf{sk}_S := A^{x_2} B^{y_2} (C')^{z_2} D^{w_2} K^{r_2}$ and outputs 1 if $\mathsf{sk}_S = \mathsf{sk}$. If either of the two checks fails, it outputs 0.

Figure 2: Our Setup.

Assume that there exists a negligible function $\varepsilon := \varepsilon(\lambda)$ such that there exists a "successful" simulator $\mathcal{S}$ for which $\mathcal{Z}$ outputs 1 with probability $1 - \varepsilon$ in the ideal world. First, assume that CDH is hard over $(\mathbb{G}, p, h)$. Consider reduction $\mathcal{R}$ that runs the simulator $\mathcal{S}$ as follows (note that $\mathcal{R}$ plays the role of the environment $\mathcal{Z}$, the PAKE functionality $\mathcal{F}_{\mathsf{PAKE}}$, and the dummy parties User and Server combined):

0. $\mathcal{R}$ receives $\mathsf{crs} = (g_1, g_2, h, c, d)$ from $\mathcal{S}$, outputs h to its challenger, and receives (h^a, h^b)

where $a, b \xleftarrow{\$} \mathbb{Z}_p$.

1. $\mathcal{R}$ sends $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{User}, \mathsf{Server})$ to $\mathcal{S}$.

2. $\mathcal{R}$ waits to receive $\mathsf{msg}_1 = \mathsf{sid}|\mathsf{VK}|A|B|C|D$ in response (as the first message from User to Server). It then sets $\mathsf{pw} := C/h^b$ and samples $x_2, y_2, w_2, r_2 \xleftarrow{\$} \mathbb{Z}_q$, setting

$$\alpha' := H(PIDs|A|B|C|D)$$
$$E := g_1^{x_2} g_2^{y_2} h^a (cd^{\alpha'})^{w_2}$$
$$F := g_1^{r_2}$$
$$G := g_2^{r_2}$$
$$I := h^{r_2} \cdot \mathsf{pw}$$
$$\beta := H(\mathsf{msg}_1|E|F|G|I)$$
$$J := (cd^\beta)^{w_2}$$

i.e., the same computation as that of the honest server with the special choice of $\mathsf{pw} = C/h^b$ and $z_2 = a$, and sends $\mathsf{msg}_2 := \mathsf{sid}|E|F|G|I|J$ to $\mathcal{S}$.

3. $\mathcal{R}$ receives $\mathsf{msg}_3 = \mathsf{sid}|K|\sigma$ (as the second message from User to Server) and $(\mathsf{sid}, \mathsf{sk})$ from $\mathcal{S}$ (as User's output to $\mathcal{Z}$), and checks if $\mathsf{Vrfy}_{\mathsf{VK}}(\mathsf{msg}_1|\mathsf{msg}_2|K, \sigma) = 1$. If not, $\mathcal{R}$ aborts. Otherwise it calculates
$$h' = \frac{\mathsf{sk}}{A^{x_2} B^{y_2} D^{w_2} K^{r_2}}.$$

4. $\mathcal{R}$ outputs h'.

Note that $\mathcal{S}$'s view while interacting with $\mathcal{R}$ is identical to $\mathcal{S}$'s view in the ideal world with environment $\mathcal{Z}$ in Figure 2; the difference is that $\mathcal{Z}$ samples pw and z_2 on its own, whereas $\mathcal{R}$ sets $\mathsf{pw} = C/h^b$ and $z_2 = a$ — which cannot be detected by $\mathcal{S}$. Let $C' = C/\mathsf{pw} = h^b$ and

$$\mathsf{sk}_S = A^{x_2} B^{y_2} (C')^{z_2} D^{w_2} K^{r_2} = A^{x_2} B^{y_2} (h^b)^a D^{w_2} K^{r_2}$$

as what $\mathcal{Z}$ would compute in its step 4; $\mathcal{Z}$ outputs 1 if and only if $\mathsf{sk}_S = \mathsf{sk}$, so by our assumption on $\mathcal{S}$, in $\mathcal{R}$'s interaction with $\mathcal{S}$, $\mathsf{sk}_S = \mathsf{sk}$ with probability $1 - \varepsilon$. But this gives $h' = h^{ab}$ with probability $1 - \varepsilon$, i.e., $\mathcal{R}$ wins with probability $1 - \varepsilon$, contradicting the hardness of fixed-CDH. $\qquad \square$

4 UC-Security of KOY in the Algebraic Group Model

We will show the following theorem.

Theorem 4.1. *The protocol of [KOY] UC-realizes the $\mathcal{F}_{\mathsf{PAKE}}$ functionality in the Algebraic Group Model.*

The broad strokes of our proof are as below.

1. In game $\mathbf{G}'_0$, the experiment is aborted if any of the following events happen.

   (a) The client accidentally generates the same VK twice for different sids. In this case, the challenger detects this in the message that is output, sends an abort and ends the protocol.

(b) The server generates msg_2 twice for different sids. The challenger detects this in the message that is output, sends an abort and ends the protocol.

(c) The challenger detects at some point that the adversary has been able to produce a new signed message for some honestly-generated VK without knowing the corresponding signing key (which it can't, because the VK is honestly generated – if it was the one who generated that VK, the adversary is allowed to do whatever it wants with it).

(d) At any given point, the challenger detects that a collision has occured in the hash function. In this case the challenger again sends an abort and terminates the protocol.

The difference in advantage of the adversary is negligible because the probability of any of these events happening is unlikely due to the security of the hash functions and the one-time signature scheme.

2. In game $\mathbf{G}_1$, the challenger changes the way that the crs is generated. In this case, the crs is now generated according to the techniques laid out in the Cramer-Shoup KeyGen procedure. The distribution of generated values is information-theoretically identical; the only difference is that now the challenger has planted backdoors into the description, and it knows the discrete logs of h, c and d. The simulator is going to (eventually) use this to extract a password guess.

3.

References

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

A   Security of the Cramer-Shoup Encryption Scheme with Delayed Reveal of the Secret Key

In this section we prove the delayed-reveal-CCA security of the Cramer-Shoup Cryptosystem. We first start by formally defining the delayed-reveal-CCA game.

A.1   Security Against Delayed-Reveal Chosen Ciphertext Attack

---

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

Stage 2: The adversary makes a sequence of calls to a decryption oracle. For each decryption oracle query, the adversary submits a ciphertext ψ, and the decryption oracle responds with $\mathsf{PKE.dec}(1^\lambda, \mathsf{SK}, \psi)$.

Stage 3: The adversary submits two messages $m_0, m_1 \in \mathsf{PKE.MsgSpace}_{\lambda, \mathsf{PK}}$ to an encryption oracle. We require that $|m_0| = |m_1|$.
On input (m_0, m_1), the encryption oracle computes

$$\sigma \xleftarrow{\$} \{0, 1\}; \psi^* \xleftarrow{\$} \mathsf{PKE.Enc}(1^\lambda, \mathsf{PKE}, m_\sigma)$$

and responds with the target ciphertext σ^*.

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

Stage 5: The adversary submits a halt statement to an *auxilliary oracle*, which responds with some auxiliary information aux. From this point on the adversary no longer has access to the decryption oracle, and is not able to make any further queries.

Stage 6: The adversary outputs $\hat{\sigma} \in \{0, 1\}$.

Figure 3: The Delayed-Reveal-CCA game.

We define the delayed-reveal-CCA advantage with auxiliary information aux of the adversary $\mathcal{A}$ against PKE at λ, denoted as $\mathsf{AdvDRCCA}(\lambda)$ to be $|\Pr[\sigma = \hat{\sigma}] - 1/2|$ in the above attack game in fig. 3 with corresponding value of aux. Informally, we say that a public-key cryptosystem is delayed-reveal-CCA secure with auxiliary information aux if the probability that $\sigma = \hat{\sigma}$ is negligible, ie. the adversary is able to correctly guess which message was encrypted with no more than negligible probability. This leads to the formal definition as below.

Definition A.1 (Security Against Delayed-Reveal-Chosen Ciphertext Attack (DR-CCA))**.** *Let* PKE *be any public-key encryption scheme. We say that* PKE *is delayed-reveal-CCA secure with auxiliary information* aux *if there exists some negligible function such that for all* $\lambda \in \mathbb{Z}_{\geq 0}$,

$$\mathsf{AdvDRCCA}(\lambda) \leq \mathsf{negl}(\lambda).$$

With this in hand, we can now formally state the theorem.

Theorem A.1. *If the DDH and the SDH assumptions hold for $\mathbb{G}$ and the hash function H is Target Collision-Resistant, then the Cramer-Shoup Encryption Scheme with Labels (fig. 1) is delayed-reveal-CCA secure with auxiliary information $\mathsf{aux} = (x_1, x_2, y_1, y_2, g_1^{r^2}, g_2^{r^2})$.*

A.2 Analysis of DR-CCA Security of the Cramer-Shoup Encryption Scheme

A.2.1 Overview

Our argument proceeds very closely to the proof of the CCA-security of the Cramer-Shoup encryption scheme outlined in Section 6.2 of [CS03]. The argument follows a series of games starting at game $\mathbf{G}_0$, which corresponds to the actual attack, and ending at $\mathbf{G}_8$, which is a game in which the ciphertext returned to the adversary is completely independent of the hidden bit σ. In each game, σ takes on identical values. We now define a few helpful variables to quantify the adversary's advantage in each game. Let T_i be the event in game $\mathbf{G}_i$ that $\hat{\sigma} = \sigma$. We will show for each game that $|\Pr[T_{i-1} - \Pr[T_i]]|$ is negligible. Game $\mathbf{G}_8$, which is completely independent of σ, naturally has probability $\Pr[T_8] = 1/2$, because in this game the adversary can do no better than a coin toss. We also introduce the variable $\mathsf{AdvDRCCA}_i(\lambda)$ to represent the *advantage* of the adversary in game $\mathbf{G}_i$, which is defined as $|\Pr[T_i] - 1/2|$. Ultimately our goal is to show that $\mathsf{AdvDRCCA}_0(\lambda) \leq \mathsf{negl}(\lambda)$.

The probability is taken over the following mutually independent random variables:

1. The internal coin tosses of $\mathcal{A}$.

2. The values $\mathsf{hk}, w, x_1, x_2, y_1, y_2, z_1, z_2$ generated independently by the key generation algorithm.

3. The values $\sigma \in \{0, 1\}$ and $r \in \mathbb{Z}_q$ which are generated by the encryption oracle.

Before continuing formally, we outline a brief sketch of each of the games.

- In game $\mathbf{G}_1$ we make some technical changes to the encryption algorithm which have no effect on the adversary's view. After game $\mathbf{G}_5$, the randomness r will be largely meaningless and thus cannot be used for correct encryption. This game ensures that encryption can occur without using r.

- Game $\mathbf{G}_2$ moves part of the auxiliary information, $\mathsf{aux}_1 = (g_1^{r^2}, g_2^{r^2})$ as an output of the decryption oracle. This can only increase the power of the adversary, since it is also allowed to make decryption oracle queries that involve the auxiliary information.

- Game $\mathbf{G}_3$ replaces $g_i^{r^2}$ with random values, which follows from SDH. It shows that the auxiliary information aux_1 has no effect on the adversary's advantage.

- Game $\mathbf{G}_4$ is a technical game that completes the argument of $\mathbf{G}_3$.

- Game $\mathbf{G}_5$ replaces u_2 with a random value, which follows from DDH. This breaks the correlation between u_1, u_2 and $h^r \cdot m_\sigma$, allowing m_σ to be replaced by a completely random value in future games.

- Game $\mathbf{G}_6$ is really the core of the proof. In this game the decryption oracle is modified so that it additionally checks whether u_2 is indeed g_2^r for some r. The indistinguishability of this game from the previous one follows from the fact that the adversary cannot with

any nontrivial probability submit a valid ciphertext which does not carry the correlation (g_1, g_2, h) that is given by the public key. In particular, note that $\mathbf{G}_5$ ensures that the target ciphertext ψ^* does *not* have that particular correlation. The immediate implication is that the adversary cannot with any nontrivial probability modify the target ciphertext ψ^* in such a way that obtains even a different, valid ciphertext, let alone one that provides information about m_σ. However, the argument of indistinguishability is nontrivial and spreads across more games.

- In game $\mathbf{G}_7$, m_σ is replaced by a uniformly random value. We will show that both the advantage of this game, as well as the probability that the adversary is able to come up with some 'bad' ciphertext which could have potentially provided information about m_σ, is negligible.

- Finally, game $\mathbf{G}_8$ bounds the probability of a 'bad' ciphertext by modifying the decryption oracle, which now rejects if the adversary is able to reuse the MAC part v of the ciphertext. The indistinguishability argument shows that if the adversary could make this query, then it either (a) broke the hash function, or (b) made an extremely unlikely guess, which it simply does not have enough information to with any nontrivial probability.

This completes a description of the games.

A.2.2 Notation

Before we proceed to the full proof, we describe some helpful notation. Most of this notation is borrowed from [CS03]. Let $\mathcal{A}$ be the DR-CCA adversary. We set the security parameter to be $\lambda \in \mathbb{Z}_{\geq 0}$ and the group description $\Gamma[\hat{\mathbb{G}}, \mathbb{G}, g, q] \in [S_\lambda]$.

Suppose that the public key is $(\Gamma, \mathsf{hk}, g_1, g_2, c, d, h)$ and that the secret key is $(\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2)$. Let $w := \log_g g_2$ and define x, y and z as follows:

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

In particular, $x = \log_g c$, $y = \log_g d$ and $z = \log_g h$.

When we deal with ciphertext ψ, we also define the following values:

- $u_1, u_2, e', e, v \in \mathbb{G}$, where $\psi = (u_1, u_2, e, v)$, with $h = u_1^{z_1} u_2^{z_2}$.

- $r, \hat{r} = \log_g u_2, \alpha, r_e = \log_g e, r_v = \log_g d$, and $t = x_1 r + y_1 r\alpha + x_2 \hat{r}w + y_2 \hat{r}\alpha w$.

We will also deal with the target ciphertext ψ^*. When dealing with this particular ciphertext, we denote each of the variables with a $*$, to obtain $u_1^*, u_2^*, e'^*, e^*, v^*, r^*, \hat{r}^*, \alpha^*, r_e^*, r_v^*, t^*$.

A.2.3 Proof of theorem A.1

We now proceed to describe the games in detail. Game $\mathbf{G}_0$ is the original attack game.

Game $\mathbf{G}_1$. This game is the same as $\mathbf{G}_1$ of [CS03].

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

E4′ : $e' \leftarrow u_1^{z_1} u_2^{z_1}$;

E7′ : $v \leftarrow u_1^{x_1 + y_1 \alpha} u_2^{x_2 + y_2 \alpha}$.

These changes are purely conceptual and the values of e'^* and v^* are exactly the same in both games. It follows that

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

Game $\mathbf{G}_2$. This is our first new game. This game is identical to $\mathbf{G}_1$ with the following modification.

We modify the *encryption* oracle in Stage 3 such that along with a ciphertext σ^*, it also outputs the *auxiliary information* $\mathsf{aux}_1 = (g_1^{r^2}, g_2^{r^2}) = (u_1^{*r}, u_2^{*r})$. Furthermore, we also modify the *auxiliary* oracle such that it no longer outputs the previous string, instead outputting only $\mathsf{aux}_2 = (x_1, x_2, y_1, y_2)$.

Our analysis of this game is simple. Note that by providing the adversary the values $(g_1^{r^2}, g_2^{r^2})$ *before* it can no longer make decryption oracle queries, we can only increase the power (and hence the advantage) of the adversary. It follows that $\mathsf{AdvDRCCA}_1(\lambda) \leq \mathsf{AdvDRCCA}_2(\lambda)$. Rephrasing, we can see that

$$|\Pr[T_1] - 1/2| \leq |\Pr[T_2] - 1/2|$$

and hence it is enough to show that $|\Pr[T_2] - 1/2| \leq \mathsf{negl}(\lambda)$. The assertion will follow immediately.

Game $\mathbf{G}_3$. In this game we again modify the encryption oracle. Instead of outputting the ciphertext $(\psi^*, \mathsf{aux}_1) = ((u_1^*, u_2^*, e^*, v^*), (g_1^{r^2}, g_2^{r^2}))$, the oracle samples a uniform s from $\mathbb{Z}_q$ and outputs $(\psi^*, \mathsf{aux}_1) = ((u_1^*, u_2^*, e^*, v^*), (g_1^s, g_2^{r^2}))$.

Note that the only difference between the games $\mathbf{G}_2$ and $\mathbf{G}_3$ is that the tuple $(g_1, u_1, \mathsf{aux}_1[0])$ is a uniformly distributed tuple from $\mathsf{SDH}_{\lambda, \Gamma}$ in $\mathbf{G}_2$ while it is a uniformly distributed tuple from $\mathsf{RandSDH}_{\lambda, \Gamma}$ in $\mathbf{G}_3$. It is thus immediately clear that the indistinguishability of the two games follows from the hardness of SDH. More specifically, we show the following.

Lemma A.1. *There exists some polynomial-time probabilistic algorithm $\mathcal{A}_{\mathsf{SDH}}$ such that*

$$|\Pr[T_3] - \Pr[T_2]| \leq \mathsf{AdvSDH}_{\mathcal{A}_{\mathsf{SDH}}, \mathcal{G}}(\lambda | \Gamma).$$

Proof. We will describe $\mathcal{A}_{\mathsf{SDH}}$ in detail. The algorithm takes in as input 1^λ, a group description $\Gamma[\hat{\mathbb{G}}, \mathbb{G}, g, q] \in [S_\lambda]$ and some tuple (g^a, g^b). The algorithm interacts with the DR-CCA adversary $\mathcal{A}$. First, it begins by computing

$$\mathsf{hk} \xleftarrow{\$} \mathsf{HF.Keyspace}_{\lambda, \Gamma}; w \xleftarrow{\$} \mathbb{Z}_q^*; x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_q; c \leftarrow g^{x_1 + w x_2}; d \leftarrow g^{y_1 + w y_2}; h \leftarrow g^{z_1 + w z_2}.$$

Using the above, it generates a public key $\mathsf{PK} = (\Gamma, \mathsf{hk}, g, g^w, c, d, h)$ and a secret key $\mathsf{SK} = (\Gamma, \mathsf{hk}, x_1, x_2, y_1, y_2, z_1, z_2)$, and passes PK to $\mathcal{A}$. We now show the behaviour of $\mathcal{A}_{\mathsf{SDH}}$ on encryption and decryption queries. Whenever it receives a ciphertext of the form $\psi = (u_1, u_2, e, v)$ to the decryption oracle, it uses SK to decrypt and outputs either $\perp$ or some m. Whenever $\mathcal{A}$ submits (m_0, m_1) to the encryption oracle, $\mathcal{A}_{\mathsf{SDH}}$ samples a uniform $\sigma \xleftarrow{\$} \{0, 1\}$ and computes

$$u_1^* = g^a; u_2^* = (g^a)^w; e^* = u_1^{*z_1} u_2^{*z_2} \cdot m_\sigma; v^* = \mathsf{HF}_{\mathsf{hk}}^{\lambda, \Gamma}(u_1^*, u_2^*, e^*); v^* = u_1^{*x_1 + v^* y_1} u_2^{*x_2 + v^* y_2}$$

and outputs the ciphertext–auxiliary information pair $(\psi^*, \mathsf{aux}_1) = ((u_1^*, u_2^*, e^*, v^*), (g^b, (g^b)^w))$. On input halt to the auxiliary oracle, $\mathcal{A}_{\mathsf{SDH}}$ outputs (x_1, x_2, y_1, y_2).

Once the game in $\mathbf{G}_2$ has been perfectly simulated, the adversary outputs 1 if $\hat{\sigma} = \sigma$ and 0 otherwise. It is clear from the above simulation that for any fixed λ and Γ,

$$\Pr[T_2] = \Pr[\mathcal{A}_{\mathsf{SDH}}(1^\lambda, \Gamma, \rho) = 1 : \rho \overset{\$}{\leftarrow} \mathsf{SDH}_{\lambda, \Gamma}]$$

$$\Pr[T_3] = \Pr[\mathcal{A}_{\mathsf{SDH}}(1^\lambda, \Gamma, \rho) = 1 : \rho \overset{\$}{\leftarrow} \mathsf{RandSDH}_{\lambda, \Gamma}].$$

It immediately follows that

$$|\Pr[T_3] - \Pr[T_2]| \leq \mathsf{AdvSDH}_{\mathcal{A}_{\mathsf{SDH}}, \mathcal{G}}(\lambda | \Gamma).$$

$\square$

Game $\mathbf{G}_4$. We modify the encryption algorithm again. Instead of outputting the ciphertext $(\psi^*, \mathsf{aux}_1) = ((u_1^*, u_2^*, e^*, v^*), (g_1^s, g_2^{r^2}))$ as before, the oracle samples a uniform s' from $\mathbb{Z}_q$ and outputs $(\psi^*, \mathsf{aux}_1) = ((u_1^*, u_2^*, e^*, v^*), (g_1^s, g_2^{s'}))$.

The proof essentially follows the same as that of the previous game, and can be omitted.

Game $\mathbf{G}_5$. This is game $\mathbf{G}_2$ of the proof of [CS03]. We modify the encryption oracle, replacing step **E3** of the encryption algorithm with

E3$'$: $\hat{r} \overset{\$}{\leftarrow} \mathbb{Z}_q \setminus \{r\}; u_2 \leftarrow g_2^{\hat{r}}.$

In the games up to $\mathbf{G}_4$ we had $r^* = \hat{r}^*$, however in game $\mathbf{G}_5$ r^* and $\hat{r}^*$ are completely independent except that they cannot be equal. Note, however, that like in the previous game, we have a tuple (g_1, g_2, u_1^*, u_2^*) which in game $\mathbf{G}_4$ is uniformly sampled from $\mathsf{DH}_{\lambda, \Gamma}$ while in game $\mathbf{G}_5$ is uniformly sampled from $\mathsf{RandDH}_{\lambda, \Gamma}$. This leads to the following lemma, which says that any distinguisher for the two games immediately gives a statistical distinguisher for the DDH instance.

Lemma A.2. *There exists some probabilistic-polynomial time algorithm $\mathcal{A}_{\mathsf{DDH}}$ such that*

$$|\Pr[T_5] - \Pr[T_4]| \leq \mathsf{AdvDDH}_{\mathcal{A}_{\mathsf{DDH}}, \mathcal{G}}(\lambda | \Gamma).$$

Proof. The proof of this lemma is borrowed from [CS03] and follows the same broad argument as that of lemma A.1. The adversary $\mathcal{A}_{\mathsf{DDH}}$ is initiated with 1^λ, a group description Γ, and a tuple $(g_2, u_1, u_2) \in \mathbb{G}^3$.

The adversary $\mathcal{A}$ is used in the same way, with the decryption and auxiliary oracles having the same description. The only difference is how the encryption oracle responds with the target ciphertext. In this case, when $\mathcal{A}_{\mathsf{DDH}}$ receives (m_0, m_1), it computes $\sigma \overset{\$}{\leftarrow} \{0, 1\}$ and the following values:

$$e^* = u_1^{*z_1} u_2^{*z_2} \cdot m_\sigma; v^* = \mathsf{HF}_{\mathsf{hk}}^{\lambda, \Gamma}(u_1^*, u_2^*, e^*); v^* = u_1^{*x_1 + v^* y_1} u_2^{*x_2 + v^* y_2}.$$

The auxiliary information aux_1 is sampled in the same manner, with g^s and $g^{s'}$ uniformly random elements of $\mathbb{G}$.

Once the game in $\mathbf{G}_4$ has been perfectly simulated, the adversary outputs 1 if $\hat{\sigma} = \sigma$ and 0 otherwise. It is clear from the above simulation that for any fixed λ and Γ,

$$\Pr[T_4] = \Pr[\mathcal{A}_{\mathsf{DDH}}(1^\lambda, \Gamma, \rho) = 1 : \rho \overset{\$}{\leftarrow} \mathsf{DH}_{\lambda, \Gamma}]$$

$$\Pr[T_5] = \Pr[\mathcal{A}_{\mathsf{DDH}}(1^\lambda, \Gamma, \rho) = 1 : \rho \overset{\$}{\leftarrow} \mathsf{RandDH}_{\lambda, \Gamma}].$$

It immediately follows that

$$|\Pr[T_5] - \Pr[T_4]| \leq \mathsf{AdvDDH}_{\mathcal{A}_{\mathsf{DDH}}, \mathcal{G}}(\lambda | \Gamma).$$

$\square$

Game G_6. This is game G_3 of the proof of [CS03]. In this game, we modify the decryption oracle. Instead of using the original decryption oracle, we modify the oracle, replacing the steps **D4** and **D5** with:

> **D4′**: Test if $u_2 = u_1^w$ and $v = u_1^{x+y\alpha}$. Output reject and halt if this is not the case.

> **D5′**: $h \leftarrow u_1^z$.

We first notice that the decryption oracle does not make any use of x_i, y_i, z_i except indirectly. It is easy to see that the decryption oracle of game G_5 correctly answers all queries that G_6 does. However, it is possible that G_6 may refuse to answer certain queries. Let R_6 be the event that there is some query which is asked to G_6 which would have been answered correctly by G_5, but is rejected by G_6.

We recall the following, which is Lemma 4 in [CS03].

Lemma A.3. *Let U_1, U_2 and F be events defined on some probability space. Suppose that the event $U_1 \wedge \neg F$ occurs iff $U_2 \wedge \neg F$ occurs. Then $|\Pr[U_1] - \Pr[U_2]| \le \Pr[F]$.*

We will apply this lemma in our analysis. Note that the event $T_5 \wedge \neg R_6$ and $T_6 \wedge \neg R_6$ are identical. By the lemma, we get

$$|\Pr[T_3] - \Pr[T_2]| \le \Pr[R_3]$$

and it suffices to bound $\Pr[R_3]$. This will be done in games G_7 and G_8.

Game G_7. This game is identical to G_6, however in this game the message is randomly selected instead of being m_σ. In particular, we modify the encryption oracle and replace **E5** with

> **E5′**: $r \overset{\$}{\leftarrow} \mathbb{Z}_q$; $e \leftarrow g^r$.

It follows that $\Pr[T_7] = 1/2$, because now ψ^* does not even involve σ, so the adversary's output is completely independent of it. We define the event R_7 which is the same as R_6, ie. some ciphertext ψ is submitted to the decryption oracle which is rejected by **D4′** but would have passed **D4**.

Lemma A.4. *We have*

$$\Pr[T_6] = \Pr[T_7]$$
$$\Pr[R_6] = \Pr[R_7]$$

Before we show the proof, we recall Lemma 9 from [CS03].

Lemma A.5. *Let k, n be integers with $1 \le k \le n$ and let K be a finite field. Consider a probability space with random variables $\vec{\alpha} \in K^{n \times 1}, \vec{\beta} = (\beta_1, \ldots, \beta_k)^\top \in K^{k \times 1}, \vec{\gamma} \in K^{k \times 1}$, and $M \in K^{k \times n}$, such that $\vec{\alpha}$ is uniformly distributed over $K^{n \times 1}$, $\vec{\beta} = M\vec{\alpha} + \vec{\gamma}$, and for $1 \le i \le k$, the ith rows of M and γ are determined by $\beta_1, \ldots, \beta_{i-1}$.*

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

We will require a less general version of this lemma, but it suffices for the proof below.

Proof (of lemma A.4). Our proof is essentially the same as that of [CS03], however the presence of the auxiliary information requires a more subtle analysis.

Consider the variable $X = (\mathsf{coins}, \mathsf{hk}, w, x_1, x_2, y_1, y_2, \sigma, r^*, \hat{r}^*)$, where coins is the internal randomness of the PPT adversary $\mathcal{A}$, and the quantity z. Note that X has the same values in $\mathbf{G}_6$ and $\mathbf{G}_7$.

Consider also the quantity r^*, which takes different values in $\mathbf{G}_6$ and $\mathbf{G}_7$. We can call these values r_6^* and r_7^* respectively. It is clear that both R_6 and T_6 are functions of X, z and r_6^*. Also, the events R_7 and T_7 have the same functional dependence on X, z and r_7^*. Thus, showing that the distributions

$$(X, z, r_6^*) \cong (X, z, r_7^*)$$

is enough to show the lemma. Now observe that if X and z are fixed, r_7^* is uniform over $\mathbb{Z}_q$. Furthermore, note that the uniformity of r_7^* is *completely* independent of $\mathsf{aux} = (x_1, x_2, y_1, y_2)$, since e is completely independent of aux as well. Hence, knowledge of aux has no bearing on the independence of r_7^*, and it remains so even if aux is public.

We show that the same is true for r_3^*. Consider

$$\begin{pmatrix} z \\ r_3^* \end{pmatrix} = \begin{pmatrix} 1 & w \\ r^* & w\hat{r}^* \end{pmatrix} \cdot \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} 0 \\ \log_g m_\sigma \end{pmatrix}.$$

Let the 2×2 matrix be referred to as M. Then M is fixed conditioned on X, but z_1 and z_2 are independently distributed over $\mathbb{Z}_q$ (even in the knowledge of aux). Furthermore, $\det(M) = w(\hat{r}^* - r^*) \neq 0$. The proposition then follows from lemma A.5. $\qed$

Game $\mathbf{G}_8$. This game is the same as $\mathbf{G}_7$, with a small modification to the decryption oracle. We modify it so that it applies a special rejection rule: if the adversary submits a ciphertext ψ for decryption at a point *after* the encryption oracle has been invoked, such that $(u_1, u_2, e) \neq (u_1^*, u_2^*, e^*)$, but $v = v^*$, then the decryption oracle outputs reject and halts even before executing $\mathbf{D4'}$.

We define two events:

1. C_8, which is the event that the adversary submits a ciphertext which is rejected by the special rule.

2. R_8, which is the even that some ciphertext ψ is submitted which is is passed by the special rule, however it is rejected by $\mathbf{D4'}$, and would have been accepted by $\mathbf{D4}$.

Note that the games proceed identically till C_8 occurs. Particularly, $R_7 \wedge \neg C_8$ and $R_8 \wedge \neg C_8$ are identical. We have, using lemma lemma A.3 that

$$|\Pr[R_5] - \Pr[R_4]| \leq \Pr[C_5].$$

We will show two lemmas that complete the proof. The first lemma shows that the special rejection rule can be broken if the underlying hash is broken, while the second shows that breaking the rule without breaking the hash requires an information-theoretically unlikely guess.

Lemma A.6. *There is some probabilistic polynomial-time algorithm $\mathcal{A}_{\mathsf{HASH}}$ such that*

$$\Pr[C_8] \leq \mathsf{AdvTCR}_{\mathsf{HF}, \mathcal{A}_{\mathsf{HASH}}}(\lambda | \Gamma).$$

Lemma A.7. *We have*

$$\Pr[R_8] \leq Q_{\mathcal{A}}(\lambda)/q.$$

The proof of lemma A.6 is identical to the proof of Lemma 7 presented in [CS03], hence we will not present it. The proof of lemma A.7 is also similar, however must be adapted to our setting.

Together, we show that $|\Pr[R_8] - \Pr[R_6]| \leq \mathsf{negl} \implies |\Pr[T_6] - 1/2| \leq \mathsf{negl}$, which proves the assertion that $|\Pr[T_0] - 1/2| \leq \mathsf{negl}$. We complete the proof with of lemma A.7 below.

Proof (of lemma A.7). Consider the proof of Lemma 8 presented in [CS03]. Both sub-parts of the proof rely on the fact that the vector $\mathsf{aux} = (x_1, x_2, y_1, y_2)$ is independent and uniformly distributed in the eyes of the adversary. Clearly this is not the case if aux is provided as auxiliary information. However, the proof only relies on the fact that aux be uniform and independent *at the time the decryption query is made.* Indeed, by the definition of the DR-CCA game, it is clear that once the auxiliary oracle is queried, no more decryption queries are allowed to be made. Hence, at the time any decryption query is made, aux is uniform and independent, and the proof of Lemma 8 follows without any additional concerns. $\qquad\square$

With this, we have shown that the Cramer-Shoup encryption scheme as defined in fig. 1 is DR-CCA secure with auxiliary information $\mathsf{aux} = (x_1, x_2, y_1, y_2, g_1^{r^2}, g_2^{r^2})$.