An extended abstract of this paper appears in Advances in Cryptology — EUROCRYPT '08, Lecture Notes in Computer Science Vol. 4965, N. Smart ed., Springer-Verlag, 2008. This is the full version.

The Twin Diffie-Hellman Problem and Applications

David Cash[1] Eike Kiltz[2] Victor Shoup[3]

June 11, 2008

Abstract

We propose a new computational problem called the *twin Diffie-Hellman problem*. This problem is closely related to the usual (computational) Diffie-Hellman problem and can be used in many of the same cryptographic constructions that are based on the Diffie-Hellman problem. Moreover, the twin Diffie-Hellman problem is at least as hard as the ordinary Diffie-Hellman problem. However, we are able to show that the twin Diffie-Hellman problem remains hard, even in the presence of a decision oracle that recognizes solutions to the problem — this is a feature not enjoyed by the ordinary Diffie-Hellman problem. In particular, we show how to build a certain "trapdoor test" that allows us to effectively answer such decision oracle queries without knowing any of the corresponding discrete logarithms. Our new techniques have many applications. As one such application, we present a new variant of ElGamal encryption with very short ciphertexts, and with a very simple and tight security proof, in the random oracle model, under the assumption that the ordinary Diffie-Hellman problem is hard. We present several other applications as well, including: a new variant of Diffie and Hellman's non-interactive key exchange protocol; a new variant of Cramer-Shoup encryption, with a very simple proof in the standard model; a new variant of Boneh-Franklin identity-based encryption, with very short ciphertexts; a more robust version of a password-authenticated key exchange protocol of Abdalla and Pointcheval.

1 Introduction

In some situations, basing security proofs on the hardness of the Diffie-Hellman problem is hindered by the fact that recognizing correct solutions is also apparently hard (indeed, the hardness of the latter problem is the decisional Diffie-Hellman assumption). There are a number of ways for circumventing these technical difficulties. One way is to simply make a stronger assumption, namely, that the Diffie-Hellman problem remains hard, even given access to a corresponding decision oracle. Another way is to work with groups that are equipped with efficient pairings, so that such a decision

[1] College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA. cdc@gatech.edu. Part of this work completed while at CWI.

[2] Cryptology & Information Security Group, CWI, Amsterdam, The Netherlands. kiltz@cwi.nl. Supported by the research program Sentinels.

[3] Dept. of Computer Science, New York University, Courant Institute, 251 Mercer Street, New York, NY 10012, USA. shoup@cs.nyu.edu. Supported by NSF award number CNS-0716690.

oracle is immediately available. However, we would like to avoid making stronger assumptions, or working with specialized groups, if at all possible.

In this paper, we introduce a new problem, the *twin Diffie Hellman problem*, which has the following interesting properties:

- the twin Diffie-Hellman problem can easily be employed in many cryptographic constructions where one would usually use the ordinary Diffie-Hellman problem, without imposing a terrible efficiency penalty;
- the twin Diffie-Hellman problem is hard, even given access to a corresponding decision oracle, assuming the ordinary Diffie-Hellman problem (without access to any oracles) is hard.

Using the twin Diffie-Hellman problem, we construct a new variant of ElGamal encryption that is secure against chosen ciphertext attack, in the random oracle model, under the assumption that the ordinary Diffie-Hellman problem is hard. Compared to other ElGamal variants with similar security properties, our scheme is attractive in that it has very short ciphertexts and a very simple and tight security proof.

At the heart of our method is a "trapdoor test" which allows us to implement an effective decision oracle for the twin Diffie-Hellman problem without knowing any of the corresponding discrete logarithms. This trapdoor test has many applications, including: a new variant of Diffie and Hellman's non-interactive key exchange protocol [12], which is secure in the random oracle model assuming the Diffie-Hellman problem is hard; a new variant of Cramer-Shoup encryption [10] with a very simple security proof, in the standard model, under the *hashed* decisional Diffie-Hellman assumption; a new variant of Boneh-Franklin identity-based encryption [6], with very short ciphertexts, and a simple and tighter security proof in the random oracle model, assuming the bilinear Diffie-Hellman problem is hard; a very simple and efficient method of securing a password-authenticated key exchange protocol of Abdalla and Pointcheval [2] against server compromise, which can be proved secure, using our trapdoor test, in the random oracle model, under the Diffie-Hellman assumption.

1.1 Hashed ElGamal Encryption and its relation to the Diffie-Hellman problem

To motivate the discussion, consider the "hashed" ElGamal encryption scheme [1]. This public-key encryption scheme makes use of a group $\mathbb{G}$ of prime order q with generator $g \in \mathbb{G}$, a hash function H, and a symmetric cipher (E, D). A public key for this scheme is a random group element X, with corresponding secret key x, where $X = g^x$. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y, \quad Z := X^y, \quad k := \mathsf{H}(Y, Z), \quad c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, c). Decryption works in the obvious way: given the ciphertext (Y, c), and secret key x, one computes

$$Z := Y^x, \quad k := \mathsf{H}(Y, Z), \quad m := \mathsf{D}_k(c).$$

The Diffie-Hellman Assumption. Clearly, the hashed ElGamal encryption scheme is secure *only if* it is hard to compute Z, given the values X and Y. Define

$$\mathrm{dh}(X, Y) := Z, \quad \text{where } X = g^x,\ Y = g^y, \text{ and } Z = g^{xy}. \tag{1}$$

The problem of computing $\mathrm{dh}(X, Y)$ given random $X, Y \in \mathbb{G}$ is the *DH problem*. The *DH assumption* asserts that this problem is hard. However, this assumption is *not* sufficient to establish the security of hashed ElGamal against a *chosen ciphertext attack*, regardless of what security properties the hash function H may enjoy.

To illustrate the problem, suppose that an adversary selects group elements $\hat{Y}$ and $\hat{Z}$ in some arbitrary way, and computes $\hat{k} := \mathsf{H}(\hat{Y}, \hat{Z})$ and $\hat{c} := \mathsf{E}_{\hat{k}}(\hat{m})$ for some arbitrary message $\hat{m}$. Further, suppose the adversary gives the ciphertext $(\hat{Y}, \hat{c})$ to a "decryption oracle," obtaining the decryption m. Now, it is very likely that $\hat{m} = m$ if and only if $\hat{Z} = \mathrm{dh}(X, \hat{Y})$. Thus, the decryption oracle can be used by the adversary as an oracle to answer questions of the form "is $\mathrm{dh}(X, \hat{Y}) = \hat{Z}$?" for group elements $\hat{Y}$ and $\hat{Z}$ of the adversary's choosing. In general, the adversary would not be able to efficiently answer such questions on his own, and so the decryption oracle is leaking some information about that secret key x which could conceivably be used to break the encryption scheme.

The Strong DH Assumption. Therefore, to establish the security of hashed ElGamal against chosen ciphertext attack, we need a stronger assumption. For $X, \hat{Y}, \hat{Z} \in \mathbb{G}$, define the predicate

$$\mathrm{dhp}(X, \hat{Y}, \hat{Z}) := \quad \mathrm{dh}(X, \hat{Y}) \stackrel{?}{=} \hat{Z}.$$

At a bare minimum, we need to assume that it is hard to compute $\mathrm{dh}(X, Y)$, given random $X, Y \in \mathbb{G}$, along with access to a *decision oracle* for the predicate $\mathrm{dhp}(X, \cdot, \cdot)$, which on input $(\hat{Y}, \hat{Z})$, returns $\mathrm{dhp}(X, \hat{Y}, \hat{Z})$. This assumption is called the *strong DH assumption* [1].[1] Moreover, it is not hard to prove, if H is modeled as a random oracle, that hashed ElGamal is secure against chosen ciphertext attack under the strong DH assumption, and under the assumption that the underlying symmetric cipher is itself secure against chosen ciphertext attack. This was proved in [1, 24], for a variant scheme in which Y is not included in the hash; including Y in the hash gives a more efficient security reduction (see [11]). Note that the strong DH assumption is different (and weaker) than the so called *gap DH assumption* [28] where an adversary gets access to a *full* decision oracle for the predicate $\mathrm{dhp}(\cdot, \cdot, \cdot)$, which on input $(\hat{X}, \hat{Y}, \hat{Z})$, returns $\mathrm{dhp}(\hat{X}, \hat{Y}, \hat{Z})$.

1.2 The Twin Diffie-Hellman Assumptions

For general groups, the strong DH assumption may be strictly stronger than the DH assumption. One of the main results of this paper is to present a slightly modified version of the DH problem that is just as useful as the (ordinary) DH problem, and which is just as hard as the (ordinary) DH problem, *even given access to a corresponding decision oracle*. Using this, we get a modified version of hashed ElGamal encryption which can be proved secure under the (ordinary) DH assumption, in the random oracle model. This modified system is just a bit less efficient than the original system.

Again, let $\mathbb{G}$ be a cyclic group with generator g, and of prime order q. Let dh be defined as in (1). Define the function

$$\begin{aligned} 2\mathrm{dh}: \qquad & \mathbb{G}^3 \to \mathbb{G}^2 \\ & (X_1, X_2, Y) \mapsto (\mathrm{dh}(X_1, Y), \mathrm{dh}(X_2, Y)). \end{aligned}$$

[1] We remark that in more recent papers the name strong DH assumption also sometimes refers to a different assumption defined over bilinear maps [4]. We follow the original terminology from [1].

We call this the *twin DH function*. One can also define a corresponding *twin DH predicate*:

$$2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) := \quad 2\mathrm{dh}(X_1, X_2, \hat{Y}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2).$$

The *twin DH assumption* states it is hard to compute $2\mathrm{dh}(X_1, X_2, Y)$, given random $X_1, X_2, Y \in \mathbb{G}$. It is clear that the DH assumption implies the twin DH assumption. The *strong twin DH assumption* states that it is hard to compute $2\mathrm{dh}(X_1, X_2, Y)$, given random $X_1, X_2, Y \in \mathbb{G}$, along with access to a *decision oracle* for the predicate $2\mathrm{dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, which on input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$, returns $2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

One of our main results is the following:

Theorem 1. *The (ordinary) DH assumption holds if and only if the strong twin DH assumption holds.*

The non-trivial direction to prove is that the DH assumption implies the strong twin DH assumption.

A Trapdoor Test. While Theorem 1 has direct applications, the basic tool that is used to prove the theorem, which is a kind of "trapdoor test," has even wider applications. Roughly stated, the trapdoor test works as follows: given a random group element X_1, we can efficiently construct a random group element X_2, together with a secret "trapdoor" τ, such that

- X_1 and X_2 are independent (as random variables), and
- if we are given group elements $\hat{Y}, \hat{Z}_1, \hat{Z}_2$, computed as functions of X_1 and X_2 (but not τ), then using τ, we can efficiently evaluate the predicate $2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$, making a mistake with only negligible probability.

We note that our trapdoor test actually appears implicitly in Shoup's DH self-corrector [32]; apparently, its implications were not understood at the time, although the techniques of Cramer and Shoup [10] are in some sense an extension of the idea. We discuss the connection between our trapdoor test and Shoup's DH self-corrector in Section 8.

1.3 Applications and Results

1.3.1 The twin ElGamal encryption scheme

Theorem 1 suggests the following *twin ElGamal encryption scheme*. This scheme makes use of a hash function H and a symmetric cipher (E, D). A public key for this scheme is a pair of random group elements (X_1, X_2), with corresponding secret key is (x_1, x_2), where $X_i = g^{x_i}$ for $i = 1, 2$. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y, \quad Z_1 := X_1^y, \quad Z_2 := X_2^y, \quad k := \mathsf{H}(Y, Z_1, Z_2), \quad c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, c). Decryption works in the obvious way: given the ciphertext (Y, c), and secret key (x_1, x_2), one computes

$$Z_1 := Y^{x_1}, \quad Z_2 := Y^{x_2}, \quad k := \mathsf{H}(Y, Z_1, Z_2), \quad m := \mathsf{D}_k(c).$$

The arguments in [1] and [11] trivially carry over, so that one can easily show that the twin ElGamal encryption scheme is secure against chosen ciphertext attack, under the strong twin DH assumption,

and under the assumption that (E, D) is secure against chosen ciphertext attack, if H is modeled as a random oracle. Again, by Theorem 1, the same holds under the (ordinary) DH assumption.

Note that the ciphertexts for this scheme are extremely compact — no redundancy is added, as in the Fujisaki-Okamoto transformation [13]. Moreover, the security reduction for our scheme is very tight. We remark that this seems to be the first DH-based encryption scheme with short ciphertexts. All other known constructions either add redundancy to the ciphertext [13, 29, 33, 9, 3] or resort to assumptions stronger than DH [1, 11, 24].

1.3.2 The twin DH key-exchange protocol

In their paper [12], Diffie and Hellman presented the following simple, *non-interactive* key exchange protocol. Alice chooses a random $x \in \mathbb{Z}_q$, computes $X := g^x \in \mathbb{G}$, and publishes the pair $(\texttt{Alice}, X)$ in a public directory. Similarly, Bob chooses a random $y \in \mathbb{Z}_q$, computes $Y := g^y \in \mathbb{G}$, and publishes the pair $(\texttt{Bob}, Y)$ in a public directory. Alice and Bob may compute the shared value $Z := g^{xy} \in \mathbb{G}$ as follows: Alice retrieves Bob's entry from the directory and computes Z as Y^x, while Bob retrieves Alice's key X, and computes Z as X^y. Before using the value Z, it is generally a good idea to hash it, together with Alice's and Bob's identities, using a cryptographic hash function H. Thus, the key that Alice and Bob actually use to encrypt data using a symmetric cipher is $k := \mathsf{H}(\texttt{Alice}, \texttt{Bob}, Z)$.

Unfortunately, the status of the security of this scheme is essentially the same as that of the security of hashed ElGamal against chosen ciphertext attack, if we allow an adversary to place arbitrary public keys in the public directory (without requiring some sort of "proof of possession" of a secret key).

To avoid this problem, we define the *twin DH protocol*, as follows: Alice's public key is (X_1, X_2), and her secret key is (x_1, x_2), where $X_i = g^{x_i}$ for $i = 1, 2$; similarly, Bob's public key is (Y_1, Y_2), and his secret key is (y_1, y_2), where $Y_i = g^{y_i}$ for $i = 1, 2$; their shared key is

$$k := \mathsf{H}(\texttt{Alice}, \texttt{Bob}, \mathrm{dh}(X_1, Y_1), \mathrm{dh}(X_1, Y_2), \mathrm{dh}(X_2, Y_1), \mathrm{dh}(X_2, Y_2)),$$

where H is a hash function. Of course, Alice computes the 4-tuple of group elements in the hash as

$$(Y_1^{x_1}, Y_2^{x_1}, Y_1^{x_2}, Y_2^{x_2}),$$

and Bob computes them as

$$(X_1^{y_1}, X_1^{y_2}, X_2^{y_1}, X_2^{y_2}).$$

Using the "trapdoor test," it is a simple matter to show that the twin DH protocol satisfies a natural and strong definition of security, under the (ordinary) DH assumption, if H is modeled as a random oracle.

1.3.3 A variant of Cramer-Shoup encryption

We present a variant of the public-key encryption scheme by Cramer and Shoup [10]. Using our trapdoor test, along with techniques originally developed for identity-based encryption [4], we give an extremely simple proof of its security against chosen-ciphertext attack, in the standard model, under the decisional DH assumption [14]: given X and Y, it is hard to distinguish $\mathrm{dh}(X, Y)$ from Z, for random $X, Y, Z \in \mathbb{G}$. In fact, our proof works under the weaker *hashed* decisional DH assumption: given X and Y, it is hard to distinguish $\mathsf{H}(\mathrm{dh}(X, Y))$ from k, for random $X, Y \in \mathbb{G}$, and random k in the range of H. As a simple extension we show that a recent variant of the

Kurosawa-Desmedt scheme from [22] can be obtained using our framework. This variant has shorter ciphertexts, and its security relies on the hashed DH assumption.

Obviously, our variants are secure under the DH assumption if H is modeled as a random oracle. We also note that by using the Goldreich-Levin theorem, a simple extension of our scheme, which is still fairly practical, can be proved secure against chosen ciphertext attack under the DH assumption.

The observation that a variant of the Cramer-Shoup encryption scheme can be proved secure under the hashed decisional DH assumption was also made by Brent Waters, in unpublished work (personal communication, 2006) and independently by Goichiro Hanaoka and Kaoru Kurosawa, in recent work [21]. In the same paper, Hanaoka and Kurosawa give two schemes based on techniques different from ours, where the first achieves CCA security in the standard model based on the DH assumption and the second achieves security based on the hashed DH assumption and has ciphertext lengths equal to those of the Kurosawa-Desmedt scheme.

1.3.4 Identity-based encryption

Strong versions of assumptions also seem necessary to analyze some identity-based encryption (IBE) schemes that use bilinear pairings. As a further contribution, we give a twin version of the *bilinear* DH (BDH) assumption and prove that its (interactive) strong twin BDH variant is implied by the standard bilinear DH assumption.

The well-known IBE scheme of Boneh and Franklin [6] achieves security against chosen ciphertext attacks, in the random oracle model, by applying the Fujisaki-Okamoto transformation. Our techniques give a different scheme with shorter ciphertexts and a tighter security reduction.

The same technique can also be applied to the scheme by Kasahara and Sakai [31] which is based on a stronger bilinear assumption but has improved efficiency.

1.3.5 Other applications

Our twinning technique and in particular the trapdoor test can be viewed as a general framework that gives a method for "updating" a protocol Π whose security relies on the strong DH assumption to a protocol Π' that has roughly the same complexity as Π, but whose security is solely based on the DH assumption. Apart from the applications mentioned above, we remark that this technique can also be applied to the undeniable signatures and designated confirmer signatures from [28], the key-exchange protocols from [23], and the public-key encryption scheme from [7].

As another application of our trapdoor test, in Section 9 we show how one can easily convert the very elegant and efficient protocol of Abdalla and Pointcheval [2] for password-authenticated key exchange, into a protocol that provides security against server compromise, without adding any messages to the protocol, and still basing the security proof, in the random oracle model, on the DH assumption.

2 A trapdoor test and a proof of Theorem 1

It is not hard to see that the strong twin DH assumption implies the DH assumption. To prove that the DH implies the strong twin DH assumption, we first need our basic tool, a "trapdoor test." Its purpose will be intuitively clear in the proof of Theorem 1: in order to reduce the strong twin DH assumption to the DH assumption, the DH adversary will have to answer decision oracle queries

without knowing the discrete logarithms of the elements of the strong twin DH problem instance. This tool gives us a method for doing so.

Theorem 2 (Trapdoor Test). *Let $\mathbb{G}$ be a cyclic group of prime order q, generated by $g \in \mathbb{G}$. Suppose X_1, r, s are mutually independent random variables, where X_1 takes values in $\mathbb{G}$, and each of r, s is uniformly distributed over $\mathbb{Z}_q$, and define the random variable $X_2 := g^s/X_1^r$. Further, suppose that $\hat{Y}, \hat{Z}_1, \hat{Z}_2$ are random variables taking values in $\mathbb{G}$, each of which is defined as some function of X_1 and X_2. Then we have:*

(i) X_2 is uniformly distributed over $\mathbb{G}$;

(ii) X_1 and X_2 are independent;

(iii) if $X_1 = g^{x_1}$ and $X_2 = g^{x_2}$, then the probability that the truth value of

$$\hat{Z}_1^r \hat{Z}_2 = \hat{Y}^s \tag{2}$$

does not agree with the truth value of

$$\hat{Z}_1 = \hat{Y}^{x_1} \wedge \ \hat{Z}_2 = \hat{Y}^{x_2} \tag{3}$$

is at most $1/q$; moreover, if (3) holds, then (2) certainly holds.

Proof. Observe that $s = rx_1 + x_2$. It is easy to verify that X_2 is uniformly distributed over $\mathbb{G}$, and that X_1, X_2, r are mutually independent, from which (i) and (ii) follow. To prove (iii), condition on fixed values of X_1 and X_2. In the resulting conditional probability space, r is uniformly distributed over $\mathbb{Z}_q$, while x_1, x_2, $\hat{Y}$, $\hat{Z}_1$, and $\hat{Z}_2$ are fixed. If (3) holds, then by multiplying together the two equations in (3), we see that (2) certainly holds. Conversely, if (3) does not hold, we show that (2) holds with probability at most $1/q$. Observe that (2) is equivalent to

$$(\hat{Z}_1/\hat{Y}^{x_1})^r = \hat{Y}^{x_2}/\hat{Z}_2. \tag{4}$$

It is not hard to see that if $\hat{Z}_1 = \hat{Y}^{x_1}$ and $\hat{Z}_2 \neq \hat{Y}^{x_2}$, then (4) certainly does not hold. This leaves us with the case $\hat{Z}_1 \neq \hat{Y}^{x_1}$. But in this case, the left hand side of (4) is a random element of $\mathbb{G}$ (since r is uniformly distributed over $\mathbb{Z}_q$), but the right hand side is a fixed element of $\mathbb{G}$. Thus, (4) holds with probability $1/q$ in this case. □

Using this tool, we can easily prove Theorem 1. So that we can give a concrete security result, let us define some terms. For an adversary $\mathcal{B}$, let us define his *DH advantage*, denoted $\mathsf{AdvDH}_{\mathcal{B},\mathbb{G}}$, to be the probability that $\mathcal{B}$ computes $\mathrm{dh}(X, Y)$, given random $X, Y \in \mathbb{G}$. For an adversary $\mathcal{A}$, let us define his *strong twin DH advantage*, denoted $\mathsf{Adv2DH}_{\mathcal{A},\mathbb{G}}$, to be the probability that $\mathcal{A}$ computes $2\mathrm{dh}(X_1, X_2, Y)$, given random $X_1, X_2, Y \in \mathbb{G}$, along with access to a *decision oracle* for the predicate $2\mathrm{dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, which on input $\hat{Y}, \hat{Z}_1, \hat{Z}_2$, returns $2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$.

Theorem 1 is a special case of the following:

Theorem 3. *Suppose $\mathcal{A}$ is a strong twin DH adversary that makes at most Q_d queries to its decision oracle, and runs in time at most τ. Then there exists a DH adversary $\mathcal{B}$ with the following properties: $\mathcal{B}$ runs in time at most τ, plus the time to perform $O(Q_\mathrm{d} \log q)$ group operations and some minor bookkeeping; moreover,*

$$\mathsf{Adv2DH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvDH}_{\mathcal{B},\mathbb{G}} + \frac{Q_\mathrm{d}}{q}.$$

In addition, if $\mathcal{B}$ does not output "failure," then its output is correct with probability at least $1 - 1/q$.

Proof. Our DH adversary $\mathcal{B}$ works as follows, given a challenge instance (X, Y) of the DH problem. First, $\mathcal{B}$ chooses $r, s \in \mathbb{Z}_q$ at random, sets $X_1 := X$ and $X_2 := g^s/X_1^r$, and gives $\mathcal{A}$ the challenge instance (X_1, X_2, Y). Second, $\mathcal{B}$ processes each decision query $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ by testing if $\hat{Z}_1\hat{Z}_2^r = \hat{Y}^s$ holds. Finally, if and when $\mathcal{A}$ outputs (Z_1, Z_2), $\mathcal{B}$ tests if this output is correct by testing if $Z_1 Z_2^r = Y^s$ holds; if this does not hold, then $\mathcal{B}$ outputs "failure," and otherwise, $\mathcal{B}$ outputs Z_1. The proof is easily completed using Theorem 2. □

3 Definitions

3.1 Public Key Encryption

We recall the usual definitions for chosen ciphertext security.

Let PKE be a public-key encryption scheme. Consider the usual chosen ciphertext attack game, played between a challenger and a adversary $\mathcal{A}$:

1. The challenger generates a public key/secret key pair, and gives the public key to $\mathcal{A}$;
2. $\mathcal{A}$ makes a number of *decryption queries* to the challenger; each such query is a ciphertext $\hat{C}$; the challenger decrypts $\hat{C}$, and sends the result to $\mathcal{A}$;
3. $\mathcal{A}$ makes one *challenge query*, which is a pair of messages (m_0, m_1); the challenger chooses $b \in \{0, 1\}$ at random, encrypts m_b, and sends the resulting ciphertext C to $\mathcal{A}$;
4. $\mathcal{A}$ makes more *decryption queries*, just as in step 2, but with the restriction that $\hat{C} \neq C$;
5. $\mathcal{A}$ outputs $\hat{b} \in \{0, 1\}$.

The advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}}$ is defined to be $|\Pr[\hat{b} = b] - 1/2|$. The scheme PKE is said to be secure against chosen ciphertext attack if for all efficient adversaries $\mathcal{A}$, the advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}}$ is negligible.

If we wish to analyze a scheme PKE in the random oracle model, then hash functions are replaced by random oracle queries as appropriate, and both challenger and adversary are given access to the random oracle in the above attack game. We write $\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{PKE}}$ for the corresponding advantage in the random oracle model.

3.2 Symmetric Encryption

If $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ is a symmetric cipher, then one defines security against chosen ciphertext attack in exactly the same way, except that in step 1 of the above attack game, the challenger simply generates a secret key (and no public key). The advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{SE}}$ is defined in exactly the same way, and PKE is said to be secure against chosen ciphertext attack if for all efficient adversaries $\mathcal{A}$, the advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{SE}}$ is negligible.

The usual construction of a chosen-ciphertext secure symmetric encryption scheme is to combine a one-time pad and a message-authentication code (MAC). We remark that such schemes do not necessarily add any redundancy to the symmetric ciphertext. In fact, Phan and Pointcheval [30] showed that a *strong PRP* [16] directly implies a length-preserving chosen-ciphertext secure symmetric encryption scheme that avoids the usual overhead due to the MAC. In practice, the modes of operation CMC [19], EME [20], and EME* [18] can be used to encrypt large messages. The

resulting scheme is chosen-ciphertext secure provided that the underlying block-cipher is a strong PRP.

3.3 Identity-Based Encryption

An IBE scheme consists of algorithms for master key generation, user key generation, encryption, and decryption. The master key generation algorithm outputs a random private/public master key pair. The user key generation algorithm uses the private master key and outputs a private user key for any identity. To encrypt a message for a user, one inputs the master public key and that user's identity to the encryption algorithm. The decryption then uses the user's private key to recover the message.

The concept of chosen ciphertext security naturally adapts to IBE. For an adversary $\mathcal{A}$ and IBE scheme IBE, the game is as follows:

1. The challenger generates a master public key/secret key pair, and gives the master public key to $\mathcal{A}$;

2. $\mathcal{A}$ makes *user secret key queries* and *decryption queries* to the challenger; each user secret key query is an identity $\hat{id}$; the challenger runs the user secret key generation on $\hat{id}$ and sends that key to $\mathcal{A}$; each decryption query is an identity $\hat{id}$ and ciphertext $\hat{c}$; the challenger decrypts $\hat{C}$ using the secret key for id and sends the result to $\mathcal{A}$;

3. $\mathcal{A}$ makes one *challenge query*, which is an identity id and a pair of messages (m_0, m_1); the challenger chooses $b \in \{0, 1\}$ at random, encrypts m_b for id, and sends the resulting ciphertext C to $\mathcal{A}$; $\mathcal{A}$ is not allowed to choose id after requesting the user private key for id in the previous step;

4. $\mathcal{A}$ makes more *user secret key queries* and *decryption queries*, just as in step 2, but with the restriction that $\hat{id} \neq id$ in user secret key queries and $(\hat{id}, \hat{C}) \neq (id, C)$ in decryption queries;

5. $\mathcal{A}$ outputs $\hat{b} \in \{0, 1\}$.

As before, we define the advantage $\mathsf{AdvCCA}_{\mathcal{A},\mathsf{IBE}}$ as $|\Pr[\hat{b} = b] - 1/2|$. When a hash function is modeled as a random oracle, we denote the advantage by $\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{IBE}}$.

4 Twin ElGamal encryption

We are now able to establish the security of the twin ElGamal encryption scheme described in §1.3.1, which we denote $\mathsf{PKE}_{2\mathrm{dh}}$. The security will be based on the strong twin DH assumption, of course, and this allows us to borrow the "oracle patching" technique from previous analyzes of hashed ElGamal encryption based on the strong DH assumption [11]. We stress, however, that unlike previous applications of this technique, the end result is a scheme based on the original DH assumption.

Theorem 4. *Suppose* H *is modeled as a random oracle and that the DH assumption holds. Then* $\mathsf{PKE}_{2\mathrm{dh}}$ *is secure against chosen ciphertext attack.*

In particular, suppose $\mathcal{A}$ *is an adversary that carries out a chosen ciphertext attack against* $\mathsf{PKE}_{2\mathrm{dh}}$ *in the random oracle model, and that* $\mathcal{A}$ *runs in time* τ*, and makes at most* Q_{h} *hash*

queries and Q_{d} decryption queries. Then there exists a DH adversary $\mathcal{B}_{\mathrm{dh}}$ and an adversary $\mathcal{B}_{\mathrm{sym}}$ that carries out a chosen ciphertext attack against SE, such that both $\mathcal{B}_{\mathrm{dh}}$ and $\mathcal{B}_{\mathrm{sym}}$ run in time at most τ, plus the time to perform $O((Q_{\mathrm{h}} + Q_{\mathrm{d}}) \log q)$ group operations; moreover,

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{PKE}_{2\mathrm{dh}}} \leq \mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}} + \frac{Q_{\mathrm{h}}}{q}.$$

Proof. In light of Theorem 1, the proof is fairly standard. We proceed with a sequence of games.

Game 0. Let Game 0 be the original chosen ciphertext attack game, and let S_0 be the event that $\hat{b} = b$ in this game.

In this game, the challenger generates the secret key (x_1, x_2) and computes the corresponding public key (X_1, X_2). We have to describe how the random oracle is implemented by the challenger. This is done in a special way to facilitate the proof. The challenger implements the random oracle using an associative array L, indexed by elements of $\mathbb{G}^3$, where each element in the array takes an initial, default value of $\perp$, indicating that it is undefined. In addition, the challenger prepares some values in advance, to be used later as part of the ciphertext generates in response to the adversary's challenge query. Namely, the challenger chooses a random symmetric key k, and a random $y \in \mathbb{Z}_q$, sets $Y := g^y$, $Z_1 := X_1^y$, and $Z_2 := X_2^y$. The challenger also sets $L[Y, Z_1, Z_2] := k$, which intuitively represents the fact that $\mathsf{H}(Y, Z_1, Z_2) = k$.

Now, the challenger sends the public key to the adversary. Whenever the adversary makes a random oracle query, the challenger sends the corresponding entry in L to the adversary, initializing it, if necessary, to a random symmetric key if it is currently $\perp$.

To process decryption queries in step 2 of the chosen ciphertext attack game, suppose the ciphertext is $(\hat{Y}, \hat{c})$. If $\hat{Y} = Y$, then the challenger simply responds with $\mathsf{D}_k(\hat{c})$. Otherwise, the challenger decrypts as usual, using the secret key (x_1, x_2), and processing its own random oracle queries using L, just as above.

To process the challenge query in step 3, the challenger uses the values Y, Z_1, Z_2, k generated in the initialization step, and computes $c := \mathsf{E}_k(m_b)$. The ciphertext (Y, c) is given to the adversary.

Decryption queries in step 4 are processed just as in step 2.

That finishes the description of Game 0. Despite the syntactic differences, it is clear that

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{PKE}_{2\mathrm{dh}}} = |\Pr[S_0] - 1/2|. \tag{5}$$

Game 1. We now describe Game 1, which is the same as Game 0, but with the following difference: in the initialization step, the challenger does not initialize $L[Y, Z_1, Z_2]$. Everything else remains *exactly* the same.

Let S_1 be the event that $\hat{b} = b$ in Game 1. Let F be the event that the adversary queries the random oracle at (Y, Z_1, Z_2) in Game 1. Note that the challenger never queries the random oracle at this point, due to the special way that decryption and challenge queries are processed. Since both Games 0 and 1 proceed identically unless F occurs, we have

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[F]. \tag{6}$$

We claim that

$$|\Pr[F]| \leq \mathsf{Adv2DH}_{\mathcal{B}_{2\mathrm{dh}},\mathbb{G}}, \tag{7}$$

where $\mathcal{B}_{2\mathrm{dh}}$ is an efficient strong twin DH adversary that makes at most Q_{h} decision oracle queries. We sketch at a very high level how $\mathcal{B}_{2\mathrm{dh}}$ works. Basically, $\mathcal{B}_{2\mathrm{dh}}$ runs just like the challenger in Game 1, but for every random oracle query $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$, $\mathcal{B}_{2\mathrm{dh}}$ sends this triple to its own decision oracle, and marks it "good" or "bad" accordingly. Using this information, $\mathcal{B}_{2\mathrm{dh}}$ can easily process decryption requests without using the secret key: given a ciphertext $(\hat{Y}, \hat{c})$ with $\hat{Y} \neq Y$, it checks if it has already seen a "good" triple of the form $(\hat{Y}, \cdot, \cdot)$ among the random oracle queries; if so, it uses the key associated with that triple; if not, it generates a random key, and it will stay on the lookout for a "good" triple of the form $(\hat{Y}, \cdot, \cdot)$ in future random oracle queries, associating this key with that triple to keep things consistent. At the end of the game, $\mathcal{B}_{2\mathrm{dh}}$ checks if it has seen a "good" triple of the form $(Y, \cdot, \cdot)$; if so, it outputs the last two components.

Of course, Theorem 1 gives us an efficient DH adversary $\mathcal{B}_{\mathrm{dh}}$ with

$$\mathsf{Adv2DH}_{\mathcal{B}_{2\mathrm{dh}},\mathbb{G}} \leq \mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + \frac{Q_{\mathrm{h}}}{q}. \tag{8}$$

Finally, it is easy to see that in Game 1, the adversary is essentially playing the chosen ciphertext attack game against SE. Thus, there is an efficient adversary $\mathcal{B}_{\mathrm{sym}}$ such that

$$|\Pr[S_1] - 1/2| = \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}}. \tag{9}$$

The theorem now follows by combining (5)–(9). □

Instantiating $\mathsf{PKE}_{2\mathrm{dh}}$ with a length-preserving chosen-ciphertext secure symmetric encryption scheme (see Appendix 3), we obtain a DH-based chosen-ciphertext secure encryption scheme with the following properties.

Optimal ciphertext overhead. The ciphertext overhead, i.e. ciphertext size minus plaintext size, is exactly one group element, which is optimal for Diffie-Hellman based schemes.

Encryption/decryption efficiency. Encryption needs three exponentiations in $\mathbb{G}$, one of which is to the fixed-base g (that can be shared among many public-keys). Decryption only needs one sequential exponentiation in $\mathbb{G}$ to compute Y^{x_1} and Y^{x_2} simultaneously, which is nearly as efficient as one single exponentiation (see, e.g., [27]).

5 Non-interactive key exchange

In this section we give a model and security definition for non-interactive key exchange and analyze the twin DH protocol from section §1.3.2. Strangely, after the seminal work of Diffie and Hellman on this subject, it does not seem to have been explored further in the literature, except in the identity-based setting.

5.1 Model and security

A non-interactive key exchange scheme KE consists of two algorithms: one for key generation and one for computing paired keys. The key generation algorithm is probabilistic and outputs a public

key/secret key pair. The paired key algorithm takes as input an identity and public key along with another identity and a secret key, and outputs a shared key for the two identities. Here, identities are arbitrary strings chosen by the users, and the key authority does not generate keys itself but acts only as a phone book.

For security we define an experiment between a challenger and an adversary $\mathcal{A}$. In this experiment, the challenger takes a random bit b as input and answers oracle queries for $\mathcal{A}$ until $\mathcal{A}$ outputs a bit $\hat{b}$. The challenger answers the following types of queries for $\mathcal{A}$:

Register honest user ID. $\mathcal{A}$ supplies a string id. The challenger runs the key generation algorithm to generate a public key/secret key pair $(\mathrm{pk}, \mathrm{sk})$ and records the tuple $(\mathrm{honest}, id, \mathrm{pk}, \mathrm{sk})$ for later. The challenger returns pk to $\mathcal{A}$.

Register corrupt user ID. In this type of query, $\mathcal{A}$ supplies both the string id and a public key pk. The challenger records the tuple $(\mathrm{corrupt}, id, \mathrm{pk})$ for later.

Get honest paired key. Here $\mathcal{A}$ supplies two identities id, id' that were registered as honest users. Now the challenger uses the bit b: if $b = 0$, the challenger runs the paired key algorithm using the public key for id and the secret key for id'. If $b = 1$, the challenger generates a random key, records it for later, and returns that to the adversary. To keep things consistent, the challenger returns the same random key for the set $\{id, id'\}$ every time $\mathcal{A}$ queries for their paired key (perhaps in reversed order).

Get corrupt paired key. Here $\mathcal{A}$ supplies two identities id, id', where id was registered as corrupt and id' was registered as honest. The challenger runs the paired key algorithm using the public key for id and the secret key for id' and returns the paired key.

When the adversary finally outputs $\hat{b}$, it wins the experiment if $\hat{b} = b$. For an adversary $\mathcal{A}$, we define its *active attack* advantage $\mathsf{AdvAA}_{\mathcal{A},\mathsf{KE}}$ to be $|\Pr[\hat{b} = b] - 1/2|$. When a hash function is modeled as a random oracle in the experiment, we denote the adversary's advantage by $\mathsf{AdvAA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{KE}}$.

We note that in the ideal version of the experiment above (when $b = 1$), the challenger returns the same random key for the honest paired key queries for (id, id') and (id', id). This essentially means that there should be no concept of "roles" in the model and that protocols should implement something like a canonical ordering of all the identities to implicitly define roles if needed.

5.2 Security of the twin DH protocol

As stated above, we can prove the twin DH protocol secure under the DH assumption using our trapdoor test. We denote the twin DH protocol by $\mathsf{KE}_{\mathrm{2dh}}$.

Theorem 5. *Suppose* H *is modeled as a random oracle and that the DH assumption holds. Then* $\mathsf{KE}_{\mathrm{2dh}}$ *is secure against active attacks.*

In particular, suppose $\mathcal{A}$ *is an adversary that attacks* $\mathsf{KE}_{\mathrm{2dh}}$ *in the random oracle model, and that* $\mathcal{A}$ *runs in time* τ*, and makes at most a total of* Q *oracle queries of all types. Then there exists a DH adversary* $\mathcal{B}_{\mathrm{dh}}$ *that runs in time at most* τ *plus the time to perform* $O(Q \log q)$ *group operations; moreover,*

$$\mathsf{AdvAA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{KE}_{\mathrm{2dh}}} \leq 2\mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + 4Q/q.$$

Proof. We proceed with a sequence of games.

Game 0. Let Game 0 be the original attack experiment and let S_0 be the event that $\hat{b} = b$ in this game. To help with the proof, we describe a specific way for the challenger to implement the experiment. The challenger uses two associative arrays, L and K, both initially empty. L will store random oracle responses, and K will store responses to paired key queries. As the adversary issues random oracle queries, the challenger stores them in L. Whenever the adversary issues a paired key query for some identities (id, id'), the challenger first uses the honest identity's secret key to compute the parties' inputs to the random oracle, say $(id, id', Z_1, Z_2, Z_3, Z_4)$. If $L[id, id', Z_1, Z_2, Z_3, Z_4] = k \neq \perp$, it stores $K[id, id'] := k$ and returns k to the adversary. Otherwise, the challenger generates a random k and stores it at both $K[id, id']$ and $L[id, id', Z_1, Z_2, Z_3, Z_4]$.

Since nothing significant has changed about the experiment,

$$\mathsf{AdvAA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{KE}_{2\mathrm{dh}}} = |\Pr[S_0] - 1/2|. \tag{10}$$

Game 1. In this game, when the adversary requests a paired key for honest identities id and id', the challenger processes the query normally, except it *only* stores the paired key in K, and not in L. Everything else is exactly the same. Thus, if the adversary happens to query the random oracle at the corresponding point for the paired key of id and id', then the challenger will check L, see that entry is uninitialized, and generate a fresh random response instead of the one stored in K.

Let S_1 be the event that $\hat{b} = b$ in Game 1. Let F be the event that the adversary queries random oracle H at a point $(id, id', \hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4)$ such that the point is the correct input to the random oracle for the paired key of some registered identities id, id'. By the construction of Game 1, it is clear that

$$|\Pr[S_1] - \Pr[S_0]| \leq \Pr[F]. \tag{11}$$

We claim that

$$\Pr[F] \leq 2(\mathsf{AdvDH}_{\mathcal{B}_{\mathrm{dh}},\mathbb{G}} + 2Q/q), \tag{12}$$

where $\mathcal{B}_{\mathrm{dh}}$ is an efficient DH adversary. We give a high-level description of $\mathcal{B}_{\mathrm{dh}}$.

$\mathcal{B}_{\mathrm{dh}}$ gets a DH instance (X, Y) as input and simulates the challenger's behavior in Game 1. It maintains the associative arrays $L[\cdot]$ and $K[\cdot, \cdot]$ to manage random oracle queries and paired keys respectively.

When registering an honest identity id, $\mathcal{B}_{\mathrm{dh}}$ generates a random bit b_{id} and random $r_{id}, s_{id}, t_{id} \in \mathbb{Z}_q$. If $b_{id} = 0$, $\mathcal{B}_{\mathrm{dh}}$ computes

$$X_1 := Xg^{t_{id}} \quad X_2 := X_1^{s_{id}}/g^{r_{id}}$$

and if $b_{id} = 1$, $\mathcal{B}_{\mathrm{dh}}$ instead computes

$$X_1 := Yg^{t_{id}} \quad X_2 := X_1^{s_{id}}/g^{r_{id}}.$$

The pair (X_1, X_2) is returned as the public key for id. $\mathcal{B}_{\mathrm{dh}}$ saves the bit b_{id}, trapdoor information r_{id}, s_{id}, and randomizing factor t_{id} for later in the simulation. $\mathcal{B}_{\mathrm{dh}}$ registers corrupt identities by simply saving them.

For honest and corrupt paired key queries with identities id, id', $\mathcal{B}_{\mathrm{dh}}$ returns $K[id, id']$ if it is initialized. Otherwise it generates a random key k, stores it in $K[id, id']$, and returns it.

On a random oracle query $\mathsf{H}(id, id', \hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4)$, $\mathcal{B}_{\text{dh}}$ returns the corresponding entry from L if it is already initialized. If not, $\mathcal{B}_{\text{dh}}$ must decide if this is the correct input to the random oracle for the paired key of id, id' in order to "patch" together queries and maintain consistency. We now describe how $\mathcal{B}_{\text{dh}}$ manages the patching.

If id, id' are not in the correct order in the query, then the query certainly won't correspond to their paired key. Otherwise, assume id is honest (a similar argument works for the case that id' is honest), and let the public keys of id, id' be $(X_1, X_2), (Y_1, Y_2)$ respectively. $\mathcal{B}_{\text{dh}}$ uses the trapdoor information for id to evaluate (with some error) the predicates

$$2\text{dhp}(X_1, X_2, Y_1, \hat{Z}_1, \hat{Z}_3) \quad \text{and} \quad 2\text{dhp}(X_1, X_2, Y_2, \hat{Z}_2, \hat{Z}_4)$$

as in Theorem 2. If both of these predicates evaluate to 1, then $\mathcal{B}_{\text{dh}}$ determines that the tuple $(id, id', \hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4)$ is the input for their paired key, it marks that tuple as "good" and stores k in $K[id, id']$.

$\mathcal{B}_{\text{dh}}$ runs until the adversary halts and then looks in L for a good tuple with honest identities id, id' such that $b_{id} \neq b_{id'}$. If $\mathcal{B}_{\text{dh}}$ finds such a good tuple $(id, id', \hat{Z}_1, \hat{Z}_2, \hat{Z}_3, \hat{Z}_4)$, it looks up the public keys $(X_1, X_2), (Y_1, Y_2)$ for id, id'. Suppose, for now, that $b_{id} = 0$ and $b'_{id} = 1$. Then $\mathcal{B}_{\text{dh}}$ computes an output for the DH problem by

$$Z := Z_1 / \left(X^{t_{id'}} Y^{t_{id}} g^{t_{id} t_{id'}} \right).$$

A similar computation works in the case that $b_{id} = 1$ and $b_{id} = 0$.

Finally we claim that $\mathcal{B}_{\text{dh}}$ produces the correct solution to its DH instance with probability at least $(1/2)(\Pr[F] - 2Q/q)$. First observe that all of the public keys given to the adversary are uniform and independent by Theorem 2. Also by Theorem 2, the probability that $\mathcal{B}_{\text{dh}}$ makes an error when patching together queries is at most $2Q/q$. It is also easy to check that the bits $b_{id}, b_{id'}$ are independent of the adversary's view. Thus, conditioned on the event that the adversary queries a good tuple, $\mathcal{B}_{\text{dh}}$ is able to compute the correct value Z with probability 1/2, and (12) follows.

The theorem is proved by combining (10)-(12). The bound on the number of group operations comes from the observation that $\mathcal{B}_{\text{dh}}$ can be implemented to never check a hashed tuple more than twice. □

6 A variant of the Cramer-Shoup encryption scheme

In this section we show how to apply our trapdoor test to construct public-key encryption schemes with security proofs in the standard model. We give a new assumption based on the *decisional* Diffie-Hellman problem and describe several schemes with varying efficiency and security properties.

6.1 The (twin) DDH assumption

Let $\mathbb{G}$ be a group of order q and let g be a random generator. Distinguishing the two distributions $(X, Y, \text{dh}(X, Y))$ and (X, Y, Z) for random $X, Y, Z \in \mathbb{G}$ is the *decision Diffie-Hellman* (DDH) problem. The *DDH assumption* states that the DDH problem is hard. As a natural decision variant of the twin DH problem, the *twin DDH problem* is distinguishing the two distributions $(X_1, X_2, Y, \text{dh}(X_1, Y))$ and (X_1, X_2, Y, Z) for random $X_1, X_2, Y, Z \in \mathbb{G}$. The *strong twin DDH*

assumption states that the twin DDH problem is hard, even given access to a decision oracle for the predicate for $2\mathrm{dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, which on input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2)$ returns $2\mathrm{dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$. (Note the value $\mathrm{dh}(X_2, Y)$ is never provided as input to the distinguisher since otherwise the strong twin DDH assumption could be trivially broken using the 2dhp oracle.)

We also consider potentially weaker "hashed" variants of the above two assumptions. For a hash function $\mathsf{H} : \mathbb{G} \to \{0,1\}^\kappa$, the *hashed DDH* problem is to distinguish the two distributions $(X, Y, \mathsf{H}(\mathrm{dh}(X, Y))$ and (X, Y, k), for random $X, Y \in \mathbb{G}$ and $k \in \{0,1\}^\kappa$. The *hashed DDH assumption* states that the hashed DDH problem is hard. Finally, the *strong twin hashed DDH assumption* states that it is hard to distinguish the distributions $(X_1, X_2, Y, \mathsf{H}(\mathrm{dh}(X, Y))$ and (X_1, X_2, Y, k), even with access to an oracle computing $2\mathrm{dhp}(X_1, X_2, \cdot, \cdot, \cdot)$, where $X_1, X_2, Y \in$ GG and $k \in \{0,1\}^\kappa$ are random.

We note that the (strong twin) hashed DDH assumption simplifies to the (strong twin) DDH assumption when H is the identity. Furthermore, there are natural groups (such as non-prime-order groups) where the DDH problem is known to be easy yet the hashed DDH problem is still assumed to be hard for a reasonable choice of the hash function [14]. If H is modeled as random oracle then the hashed DDH and the DH assumption become equivalent.

Using the trapdoor test in Theorem 2, we can prove an analogue of Theorem 3.

Theorem 6. *The (hashed) DDH assumption holds if and only if the strong twin (hashed) DDH assumption holds. In particular, suppose $\mathcal{A}$ is a strong twin (hashed) DDH adversary that makes at most Q_d queries to its decision oracle, and runs in time at most τ. Then there exists a (hashed) DDH adversary $\mathcal{B}$ with the following properties: $\mathcal{B}$ runs in time at most τ, plus the time to perform $O(Q_\mathrm{d} \log q)$ group operations and some minor bookkeeping; moreover,*

$$\mathsf{Adv2DDH}_{\mathcal{A},\mathbb{G}} \leq \mathsf{AdvDDH}_{\mathcal{B},\mathbb{G}} + \frac{Q_\mathrm{d}}{q}.$$

6.2 A variant of the Cramer-Shoup scheme

We now can consider the following encryption scheme which we call $\mathsf{PKE}_{\widetilde{\mathrm{CS}}}$. This scheme makes use of a symmetric cipher (E, D) and a hash function $\mathsf{T} : \mathbb{G} \to \mathbb{Z}_q$ which we assume to be target collision-resistant [11]. A public key for this scheme is a tuple of random group elements $(X_1, \tilde{X}_1, X_2, \tilde{X}_2)$, with corresponding secret key $(x_1, \tilde{x}_1, x_2, \tilde{x}_2)$, where $X_i = g^{x_i}$ and $\tilde{X}_i = g^{\tilde{x}_i}$ for $i = 1, 2$. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y, \quad t := \mathsf{T}(Y), \quad Z_1 := (X_1^t \tilde{X}_1)^y, \quad Z_2 := (X_2^t \tilde{X}_2)^y, \quad k := \mathsf{H}(X_1^y), \quad c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, Z_1, Z_2, c). Decryption works as follows: given the ciphertext (Y, Z_1, Z_2, c) and secret key $(x_1, \tilde{x}_1, x_2, \tilde{x}_2)$, one computes $t := \mathsf{T}(Y)$ and checks if

$$Y^{x_1 t + \tilde{x}_1} = Z_1 \text{ and } Y^{x_2 t + \tilde{x}_2} = Z_2. \tag{13}$$

If not (we say the ciphertext is *not consistent*), reject; otherwise, compute

$$k := \mathsf{H}(Y^{x_1}), \quad m := \mathsf{D}_k(c).$$

We remark that since $|\mathbb{G}| = |\mathbb{Z}_q| = q$, hash function T could be a bijection. See [8] for efficient constructions for certain groups $\mathbb{G}$.

Relation to Cramer-Shoup. Our scheme is very similar to the one by Cramer and Shoup [10]. Syntactically, the difference is that in Cramer-Shoup the value Z_1 is computed as $Z_1 = X_3^y$ (where

X_3 is another random group element in the public key) and t is computed as $t = \mathsf{T}(Y, Z_1)$. However, our variant allows for a *simple security proof* based on the *hashed* DDH assumption whereas for the Cramer-Shoup scheme only a proof based on the DDH assumption is known (and the currently known proofs do not allow for it). In Appendix A, we review the proof of the original Cramer-Shoup scheme and show how a portion of the scheme can be analyzed using the twin DDH assumption. The purpose of this appendix is primarily expository.

We now show that, using the trapdoor test, $\mathsf{PKE}_{\widetilde{\mathrm{cs}}}$ allows for a very elementary proof under the hashed DDH assumption. We stress that are security proof is not in the random oracle model.

Theorem 7. *Suppose* T *is a target collision resistant hash function. Further, suppose the hashed DDH assumption holds, and that the symmetric cipher* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is secure against chosen ciphertext attack. Then* $\mathsf{PKE}_{\widetilde{\mathrm{cs}}}$ *is secure against chosen ciphertext attack.*

In particular, suppose $\mathcal{A}$ *is an adversary that carries out a chosen ciphertext attack against* $\mathsf{PKE}_{\widetilde{\mathrm{cs}}}$ *and that* $\mathcal{A}$ *runs in time* τ*, and makes at most* Q_{d} *decryption queries. Then there exists a hashed DDH adversary* $\mathcal{B}_{\mathrm{ddh}}$*, an adversary* $\mathcal{B}_{\mathrm{sym}}$ *that carries out a chosen ciphertext attack against* SE*, and a TCR adversary* $\mathcal{B}_{\mathrm{tcr}}$ *such that both* $\mathcal{B}_{\mathrm{ddh}}$*,* $\mathcal{B}_{\mathrm{sym}}$ *and* $\mathcal{B}_{\mathrm{tcr}}$ *run in time at most* τ*, plus the time to perform* $O(Q_{\mathrm{d}} \log q)$ *group operations; moreover,*

$$\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\widetilde{\mathrm{cs}}}} \leq \mathsf{AdvDDH}_{\mathcal{B}_{\mathrm{ddh}},\mathbb{G},\mathsf{H}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}} + \mathsf{AdvTCR}_{\mathcal{B}_{\mathrm{tcr}},\mathsf{T}} + \frac{Q_{\mathrm{d}}}{q}.$$

Proof. We proceed with a sequence of games.

Game 0. Let Game 0 be the original chosen ciphertext attack game, and let S_0 be the event that $\hat{b} = b$ in this game.

$$\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\widetilde{\mathrm{cs}}}} = |\Pr[S_0] - 1/2|. \tag{14}$$

Game 1 Let Game 1 be like Game 0, but with the following difference. Game 1 aborts if the adversary, at any time, makes a decryption query containing a $\hat{Y}$ such that $\hat{Y} \neq Y$ and $\mathsf{T}(\hat{Y}) = \mathsf{T}(Y)$ where Y comes from the challenge ciphertext. Using a standard argument from [11] it is easy to show that

$$|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{AdvTCR}_{\mathcal{B}_{\mathrm{tcr}},\mathsf{T}}. \tag{15}$$

Game 2. Let Game 2 be as Game 1 with the following differences. For computing the public-key the experiment picks $x_1, x_2, y, a_1, a_2 \in \mathbb{Z}_q$ at random and computes $X_1 = g^{x_1}$, $X_2 = g^{x_2}$, and $Y = g^y$. Next, it computes $t := \mathsf{T}(Y)$ and

$$\tilde{X}_1 := X_1^{-t} g^{a_1}, \quad \tilde{X}_2 := X_2^{-t} g^{a_2}.$$

Note that the way the public-key is setup uses a technique to prove selective-ID security for IBE schemes [4].

The challenge ciphertext (Y, Z_1, Z_2, c) for message m_b is computed as

$$t := \mathsf{T}(Y), \quad Z_1 := Y^{a_1}, \quad Z_2 := Y^{a_2}, \quad k := \mathsf{H}(X_1^y), \quad c := \mathsf{E}_k(m_b). \tag{16}$$

This is a correctly distributed ciphertext for m_b and randomness $y = \log_g(Y)$ since, for $i = 1, 2$, $(X_i^t \tilde{X}_i)^y = (X_i^{t-t} g^{a_i})^y = (g^{a_i})^y = Y^{a_i} = Z_i$. We can assume (Y, Z_1, Z_2, k) to be computed in the beginning of the experiment since they are independent of m_0, m_1.

A decryption query for ciphertext $(\hat{Y}, \hat{Z}_1, \hat{Z}_2, \hat{c})$ is answered as follows. Compute $\hat{t} = \mathsf{T}(\hat{Y})$. If $t = \hat{t}$ then verify consistency by checking if $Z_1 = \hat{Z}_1$ and $Z_2 = \hat{Z}_2$. If the ciphertext is consistent then use the challenge key k defined in (16) to decrypt $\hat{c}$. If $t \neq \hat{t}$ then proceed as follows. For $i = 1, 2$, compute $\bar{Z}_i = (\hat{Z}_i / \hat{Y}^{a_i})^{1/(\hat{t}-t)}$. Consistency of the ciphertext is verified by checking if

$$\hat{Y}^{x_1} = \bar{Z}_1 \text{ and } \hat{Y}^{x_2} = \bar{Z}_2. \tag{17}$$

Let $\hat{y} = \log_g \hat{Y}$. The value $\hat{Z}_i$ was correctly generated iff $\hat{Z}_i = (X_i^{\hat{t}} \tilde{X}_i)^{\hat{y}} = (X_i^{\hat{t}-t} g^{a_i})^{\hat{y}} = (\hat{Y}^{x_i})^{\hat{t}-t} \cdot \hat{Y}^{a_i}$ which is equivalent to $\bar{Z}_i = \hat{Y}^{x_i}$. Hence, (17) is equivalent to the test from the original scheme (13). If the ciphertext is consistent then one can use the symmetric key $\hat{k} = \mathsf{H}(\bar{Z}_1) = \mathsf{H}(\hat{Y}^{x_1})$ to decrypt $\hat{c}$ and return $\hat{m} = \mathsf{D}_{\hat{k}}(\hat{c})$.

Let S_2 be the event that $\hat{b} = b$ in this game. As we have seen,

$$\Pr[S_2] = \Pr[S_1]. \tag{18}$$

Game 3. Let Game 3 be as Game 2 with the only difference that the value k to compute the challenge ciphertext is now chosen at random. We claim that

$$|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{Adv2DDH}_{\mathcal{B}_{2\mathrm{ddh}}, \mathbb{G}, \mathsf{H}}, \tag{19}$$

where $\mathcal{B}_{2\mathrm{ddh}}$ is an efficient strong twin hashed DDH adversary that makes at most Q_{d} queries to the decision oracle. $\mathcal{B}_{2\mathrm{ddh}}$ is defined as follows. Using the values (X_1, X_2, Y, k) from its challenge (where either $k = \mathsf{H}(\mathrm{dh}(X_1, Y))$ or k is random), adversary $\mathcal{B}_{2\mathrm{ddh}}$ runs (without knowing x_1, x_2, y) the experiment as described in Game 2 using k as the challenge key in (16) to encrypt m_b. Note that the only point where Games 2 and 3 make use of x_1 and x_2 is the consistency check (17) which $\mathcal{B}_{2\mathrm{ddh}}$ equivalently implements using the 2dhp oracle, i.e. by checking if

$$2\mathrm{dhp}(X_1, X_2, \hat{Y}, \bar{Z}_1, \bar{Z}_2)$$

holds. We have that if $k = \mathsf{H}(\mathrm{dh}(X_1, Y)) \in \{0,1\}^\kappa$, this perfectly simulates Game 2, whereas if $k \in \{0,1\}^\kappa$ is random this perfectly simulates Game 3. This proves (19).

Finally, it is easy to see that in Game 3, the adversary is essentially playing the chosen ciphertext attack game against SE. Thus, there is an efficient adversary $\mathcal{B}_{\mathrm{sym}}$ such that

$$|\Pr[S_3] - 1/2| = \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}}, \mathsf{SE}}. \tag{20}$$

The theorem now follows by combining (14)–(20) with Theorem 6. □

6.3 A variant with shorter ciphertexts

Consider a ciphertext in the above scheme that is of the form $(Y = g^r, Z_1 = (X_1^t \tilde{X}_1)^{r'}, Z_2, c)$ with $r \neq r'$. Such a ciphertext is inconsistent and should therefore be rejected by (13) in the decryption algorithm. Essentially, the trapdoor test says that in the view of the adversary, the unique value Z_2 that leads the simulation (as described in the proof of Theorem 7) to *falsely accept* such ciphertexts is a uniformly distributed group element. Therefore, the adversary can never guess this "bad Z_2" and, with high probability, the simulation of the CCA experiment is correct.

With this intuition it is easy to see that one can as well replace $Z_2 \in \mathbb{G}$ in the ciphertext by $Z_2' = \mathsf{KDF}(Z_2) \in \{0,1\}^k$, where $\mathsf{KDF} : \mathbb{G} \to \{0,1\}^k$ is a secure key-derivation function. (For uniform $X \in \mathbb{G}$, $\mathsf{KDF}(X)$ is computationally indistinguishable from an uniform bitstring in $\{0,1\}^k$.) Accordingly, decryption is modified to check $Z_2' = \mathsf{KDF}(Y^{x_2 t + \tilde{x}_2})$. This variant shortens the ciphertexts by replacing a group element by a bitstring in $\{0,1\}^k$.

Yet another variant uses the value Z_2 directly as a source for an integrity check of the symmetric cipher. Here we assume that symmetric encryption satisfies the stronger notion of (one-time) authenticated encryption [22]. Such a ciphertext can, for example, be obtained by combining a one-time pad with a message authenticated code (MAC). The idea is to move the value Z_2 from the ciphertext into the symmetric key which we re-define as $k = \mathsf{H}(X_1^y \cdot Z_2) = \mathsf{H}(X_1^y \cdot (X_2^t \tilde{X}_2)^y)$. Now, if (Y, Z_1) is inconsistent (in the above sense that $r \neq r'$) then the value for Z_2 used in the simulation is random and will make the symmetric key k essentially look random (from the adversary's view). Consequently, the authenticity property of the symmetric cipher makes the simulated decryption algorithm reject this ciphertext. After applying one more simplification (defining $Z_2 = X_2^y$) we get the following scheme which we call $\mathsf{PKE}_{\widetilde{\mathrm{kd}}}$.

Public and secret keys as the same as in $\mathsf{PKE}_{\widetilde{\mathrm{cs}}}$ with the difference that the element $\tilde{X}_2$ is no longer needed in the public-key. To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$, computes

$$Y := g^y, \quad t := \mathsf{T}(Y), \quad Z_1 := (X_1^t \tilde{X}_1)^y, \quad k := \mathsf{H}(X_2^y), \quad c := \mathsf{E}_k(m),$$

and the ciphertext is (Y, Z_1, c). Decryption works as follows: given the ciphertext (Y, Z_1, c), and secret key $(x_1, \tilde{x}_1, x_2)$, one computes $t := \mathsf{T}(Y)$ and checks if

$$Y^{x_1 t + \tilde{x}_1} = Z_1.$$

If not, reject; otherwise, compute

$$k := \mathsf{H}(Y^{x_2}), \quad m := \mathsf{D}_k(c).$$

This scheme is essentially the public-key encryption scheme presented in [22]. Here, using the trapdoor test we offer a different and maybe simpler interpretation of its security.

Theorem 8. *Suppose* T *is a target collision resistant hash function. Further, suppose the hashed DDH assumption holds, and that the symmetric cipher* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is secure in the sense of authenticated encryption. Then* $\mathsf{PKE}_{\widetilde{\mathrm{kd}}}$ *is secure against chosen ciphertext attack.*

A proof and a more precise security statement can be looked up in [22] or can alternatively be obtained by modifying the proof of Theorem 7 as described above. We remark that even though it is not explicitly mentioned in [22] their original proof already implies security of the $\mathsf{PKE}_{\widetilde{\mathrm{kd}}}$ scheme based on the *Hashed* DDH assumption.

6.4 A variant with security from the DH assumption

We now consider an extension of $\mathsf{PKE}_{\widetilde{\mathrm{cs}}}$ that achieves security based on the (computational) DH assumption. The idea is to first extend the public keys and ciphertexts to have several X_i and Z_i terms, respectively, and then use the Goldreich-Levin hard-core function [17, 16] as the hash function to extract symmetric key bits. Because security depends on the reduction to the hard-coreness of the function, the reduction is not very tight, and so we carry out the analysis in asymptotic terms. Below, we denote by f_{gl} the Goldreich-Levin hard-core function for $\mathrm{dh'}(X, Y, R) := (\mathrm{dh}(X, Y), R)$.

Let κ be the security parameter, and for simplicity we assume that it is also the length (in bits) of the symmetric keys for (E, D). Let $\nu = O(\log \kappa)$ be some integer that divides κ, and $\ell := \kappa/\nu$. In this scheme, the secret key now consists of $2(\ell+1)$ random elements of $\mathbb{Z}_p$, denoted $x_i, \tilde{x}_i$ for $i = 1, \ldots, \ell+1$. The public key contains the $2(\ell+1)$ corresponding group elements $X_i = g^{x_i}, \tilde{X}_i = g^{\tilde{x}_i}$, for $i = 1, \ldots, \ell+1$, along with a random bit string R of length long enough to evaluate a hard-core function with ν output bits ($u := 2\log|\mathbb{G}|$ bits are sufficient). To encrypt a message m, one chooses a random $y \in \mathbb{Z}_q$ and computes

$$Y := g^y, \quad t := \mathsf{T}(Y), \quad Z_i := (X_i^t \tilde{X}_i)^y \quad \text{for } i = 1, \ldots, \ell+1.$$

Then one sets $k_i := f_{\mathrm{gl}}(X_i^y, R) \in \{0,1\}^\nu$ $(i = 1, \ldots, \ell)$. Note that $X_{\ell+1}, \tilde{X}_{\ell+1}$, and $Z_{\ell+1}$ are not used for key derivation. Finally, a concatenation of all k_i yields a symmetric key $k \in \{0,1\}^\kappa$ that is used to encrypt m as $c := \mathsf{E}_k(m)$. The ciphertext is $(Y, Z_1, \ldots, Z_{\ell+1}, c)$. Decryption first verifies the consistency of $(Y, Z_1, \ldots, Z_{\ell+1}, c)$ by checking if $Y^{x_i t + \tilde{x}_i} = Z_i$ for all $i = 1, \ldots, \ell+1$. Then the key k is reconstructed as the concatenation of $k_i = f_{\mathrm{gl}}(Y^{x_i}, R)$ for $i = 1, \ldots, \ell$, and finally m is recovered by computing $m := \mathsf{D}_k(c)$.

In order to analyze this scheme we will need the following version of the Goldreich-Levin theorem.

Theorem 9. *Suppose that $\mathcal{A}_{\mathrm{gl}}$ is a probabilistic poly-time algorithm such that $\mathcal{A}_{\mathrm{gl}}(X, Y, R, k)$ distinguishes $k = f_{\mathrm{gl}}(\mathrm{dh}(X, Y), R)$ from a uniform string with non-negligible advantage, for random $X, Y \in \mathbb{G}$ and random $R \in \{0,1\}^u$. Then there exists a probabilistic poly-time algorithm $\mathcal{A}_{\mathrm{dh}}$ that computes $\mathrm{dh}(X, Y)$ with non-negligible probability for random X, Y.*

Using our techniques, it also not hard to show that this theorem still holds if we assume that $\mathcal{A}_{\mathrm{gl}}$ additionally gets as input a random $X' \in \mathbb{G}$ and has access to an oracle computing $2\mathrm{dhp}(X, X', \cdot, \cdot, \cdot)$. We will use this augmented version in our analysis below.

Theorem 10. *Suppose T is a target collision resistant hash function. Further, suppose the DH assumption holds, and that the symmetric cipher $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ is secure against chosen ciphertext attack. Then $\mathsf{PKE}_{\mathrm{dh}}$ is secure against chosen ciphertext attack.*

Proof. We proceed with a sequence of games. For each i, let S_i be the event that $\hat{b} = b$ in Game i.

Game 0. We define Game 0 to be the original CCA game that $\mathcal{A}$ plays against $\mathsf{PKE}_{\mathrm{dh}}$. By definition,

$$|\Pr[S_0] - 1/2| = \mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\mathrm{dh}}} \tag{21}$$

Game 1. Game 1 is the same as Game 0, except now if the adversary asks for a decryption of a ciphertext containing $\hat{Y} \neq Y$ but $\mathsf{T}(\hat{Y}) = \mathsf{T}(Y)$, where Y is from the challenge ciphertext, then the game aborts. By the target-collision resistance of T, we have that

$$|\Pr[S_1] - \Pr[S_0]| \leq \mathrm{negl}(\kappa). \tag{22}$$

Games 2. Game 2 is the same as Game 1, except that k is set to a uniform and independent bit string. We claim that

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathrm{negl}(\kappa). \tag{23}$$

We will prove this by a hybrid argument. For $j = 0, \ldots, \ell$ and $d = 0, 1$, we define the hybrid games $\mathsf{H}_{j,d}$. Intuitively, in $\mathsf{H}_{j,0}$, $k_1, \ldots, k_j$ will be uniformly random strings, while $k_{j+1}, \ldots, k_\ell$ will be computed normally. For each j, the hybrids will be defined so that $\mathsf{H}_{j,0}$ and $\mathsf{H}_{j,1}$ have exactly the same output distribution, but are run in a slightly different way to facilitate the proof. In addition, $\mathsf{H}_{0,1}$ will have the same output distribution as Game 1, and $\mathsf{H}_{\ell,0}$ will have that of Game 2.

We now describe the hybrid games. Fix some $j \in \{0, \ldots, \ell\}$. We start with $\mathsf{H}_{j,0}$ and show how to define $\mathsf{H}_{j,1}$ afterwards. In $\mathsf{H}_{j,0}$, the public key is generated as follows. It samples $y \leftarrow_R \mathbb{Z}_q$ and sets $Y := g^y$, $t := \mathsf{T}(Y)$. For $i = 1, \ldots \ell$, $i \neq j$, $X_i, \widetilde{X}_i$ are generated normally. The game then samples $x_j, x_{\ell+1}, a_j, a_{\ell+1} \leftarrow_R \mathbb{Z}_q$ and computes

$$X_j := g^{x_j}, \quad X_{\ell+1} := g^{x_{\ell+1}}, \quad \widetilde{X}_j := X_j^{-t} g^{a_j}, \quad \widetilde{X}_{\ell+1} := X_{\ell+1}^{-t} g^{a_{\ell+1}}. \tag{24}$$

Finally, it samples $R \leftarrow_R \{0,1\}^u$ and sets the public key to $(X_1, \widetilde{X}_1, \ldots, X_{\ell+1}, \widetilde{X}_{\ell+1}, R)$.

To compute the challenge ciphertext, $\mathsf{H}_{j,0}$ does the following. It uses the Y that it computed at the start of the game, and sets $Z_i := Y^{x_i}$ for $i \neq j, \ell+1$. It sets $Z_j := Y^{a_j}$ and $Z_{\ell+1} := Y^{a_{\ell+1}}$. For $i = 1, \ldots, j$, it sets $k_i \leftarrow_R \{0,1\}^\nu$. For $i = j+1, \ldots, \ell$ it computes $k_i := f_{\mathrm{gl}}(Y^{x_i}, R)$. It uses $k := k_1 \ldots k_\ell$ and computes $c := \mathsf{E}_k(m_b)$, and the challenge ciphertext is $(Y, Z_1, \ldots, Z_{\ell+1}, c)$.

To respond to a decryption query for the ciphertext $(\hat{Y}, \hat{Z}_1, \ldots, \hat{Z}_{\ell+1}, \hat{c})$, $\mathsf{H}_{j,0}$ first computes $\hat{t} := \mathsf{T}(\hat{Y})$. If $\hat{t} = t$, it checks if $\hat{Z}_i = Z_i$ for all i. If this holds it decrypts $\hat{c}$ using k. If $\hat{t} \neq t$, it verifies the consistency of Z_i for $i \neq j, \ell+1$, normally. It then computes

$$\bar{Z}_j := (\hat{Z}_j / \hat{Y}_j^{a_j})^{1/(t-\hat{t})}, \qquad \bar{Z}_{\ell+1} := (\hat{Z}_{\ell+1} / \hat{Y}_{\ell+1}^{a_{\ell+1}})^{1/(t-\hat{t})}$$

and tests if

$$\hat{Y}^{x_j} = \bar{Z}_j \ \text{ and } \ \hat{Y}^{x_{\ell+1}} = \bar{Z}_{\ell+1}. \tag{25}$$

If this holds, it computes $\hat{k}_j := f_{\mathrm{gl}}(\bar{Z}_j, R)$ and then computes the rest of the $\hat{k}_i$ normally (as $f_{\mathrm{gl}}(Y^{x_i}, R)$) and decrypts $\hat{c}$ using $\hat{k} := \hat{k}_1 \ldots \hat{k}_\ell$. This completes the decryption of $\mathsf{H}_{j,0}$.

We let the hybrid game $\mathsf{H}_{j,1}$ be *exactly* like $\mathsf{H}_{j+1,0}$, except that k_{j+1} is computed as $k_{j+1} := f_{\mathrm{gl}}(Y^{x_{j+1}}, R)$ instead of being set to a random string. Note that in both $\mathsf{H}_{j,0}$ and $\mathsf{H}_{j,1}$, $k_1, \ldots, k_j$ are set to random strings and $k_{j+1}, \ldots, k_\ell$ are computed normally. The only difference between them is them is the way in which the games are "managed," but the output distributions are exactly the same (the change between $\mathsf{H}_{j,0}$ and $\mathsf{H}_{j,1}$ is essentially like the change between Games 1 and 2 in the proof of Theorem 7, and the argument there can be applied here).

We are now ready to describe our adversary that breaks the hardcore-ness of f_{gl}. Let $\mathcal{B}_{\mathrm{gl}}$ be an adversary that gets (X, X', Y, R, s) as input, where either $s = f_{\mathrm{gl}}(\mathrm{dh}(X, Y), R)$ or s is a random string. In addition, $\mathcal{B}_{\mathrm{gl}}$ has access to an oracle computing $2\mathrm{dhp}(X, X', \hat{Y}, \hat{Z}, \hat{Z}')$.

$\mathcal{B}_{\mathrm{gl}}$ does the following. It selects $j \leftarrow_R \{1, \ldots, \ell\}$, sets $X_j := X$, and $X_{\ell+1} := X'$. It proceeds to simulate $\mathsf{H}_{j-1,1}$ for $\mathcal{A}$, except that it sets $k_j := s$. The only point where x_j and $x_{\ell+1}$ are used is in the consistency check in (25), which $\mathcal{B}_{\mathrm{gl}}$ can perform by using an oracle query as in the proof of Theorem 7. When $\mathcal{A}$ outputs $\hat{b}$, $\mathcal{B}_{\mathrm{gl}}$ checks if $\hat{b} = b$, and outputs 1 if this holds and 0 otherwise.

It is not hard to check that, conditioned on $s = f_{\mathrm{gl}}(\mathrm{dh}(X, Y), R)$, $\mathcal{B}_{\mathrm{gl}}$ simulates $\mathsf{H}_{j-1,1}$ for $\mathcal{A}$, and conditioned on the event that s was random, $\mathcal{B}_{\mathrm{gl}}$ simulates $\mathsf{H}_{j,0}$. Then the following standard

hybrid argument applies.

$$
\begin{aligned}
&\Pr\left[\mathcal{B}_{\mathrm{gl}}(X,Y,R,s)=1 \,|\, s=f_{\mathrm{gl}}(\mathrm{dh}(X,Y),R)\right]-\Pr\left[\mathcal{B}_{\mathrm{gl}}(X,Y,R,s)=1 \,|\, s \text{ is random}\right]\\
&=\frac{1}{\ell}\sum_{j=1}^{\ell}\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{j-1,1}\right]-\frac{1}{\ell}\sum_{j=1}^{\ell}\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{j,0}\right]\\
&=\frac{1}{\ell}\sum_{j=1}^{\ell}\left(\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{j-1,1}\right]-\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{j,0}\right]\right)\\
&=\frac{1}{\ell}\left(\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{0,1}\right]-\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{\ell,0}\right]\right) \qquad (26)\\
&=\frac{1}{\ell}\left(\Pr\left[S_1\right]-\Pr\left[S_0\right]\right)
\end{aligned}
$$

We get (26) by observing that $\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{j,0}\right]=\Pr\left[\hat{b}=b \text{ in } \mathsf{H}_{j,1}\right]$ for $j=1,\ldots,\ell-1$.

Finally, if the advantage of $\mathcal{B}_{\mathrm{gl}}$ is non-negligible, then by Theorem 9 (augmented with our trapdoor test), we get an adversary $\mathcal{B}_{\mathrm{dh}}$ that solves the DH problem with non-negligible advantage. Then, by the DH assumption, (23) follows.

Returning to the proof of the theorem, in Game 2 the adversary is mounting a chosen-ciphertext attack against the symmetric encryption scheme. Thus, by the CCA security of SE,

$$\Pr[S_2]=\mathrm{negl}(k). \qquad (27)$$

The proof is completed by combining (21), (22), (23), and (27). □

7 Identity based encryption

In this section we show how to apply the trapdoor test in Theorem 2 to identity-based encryption. We give a bilinear version of the strong twin DH problem and show that it can be reduced to the standard bilinear DH problem. We then use this assumption to construct a new IBE scheme that we call twin Boneh-Franklin. While our scheme is not as computationally efficient as some other CCA secure schemes, it only incures one group element of overhead in the ciphertexts and has tighter reduction to the BDH assumption than the original (CPA) scheme on which it is based.

7.1 The (twin) BDH assumption

In groups equipped with a pairing $\hat{e}:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$, we can define the function

$$\mathrm{bdh}(X,Y,W):=Z, \quad \text{where } X=g^x,\ Y=g^y,\ W=g^w, \text{ and } Z=\hat{e}(g,g)^{wxy}.$$

Computing $\mathrm{bdh}(X,Y,W)$ for random $X,Y,W\in\mathbb{G}$ is the *bilinear DH (or BDH) problem.* The *BDH assumption* states that solving the BDH problem is hard. Next we define a predicate

$$\mathrm{bdhp}(X,\hat{Y},\hat{W},\hat{Z}):= \quad \mathrm{bdh}(X,\hat{Y},\hat{W})\stackrel{?}{=}\hat{Z}.$$

We can also consider the BDH problem where, in addition to random (X,Y,W), one is also given access to an oracle that on input $(\hat{Y},\hat{W},\hat{Z})$ returns $\mathrm{bdhp}(X,\hat{Y},\hat{W},\hat{Z})$. The *strong BDH assumption* [25] states that the BDH problem remains hard even with the help of the oracle.

For reasons similar to the issue with hashed ElGamal encryption, the strong BDH assumption seems necessary to prove the CCA security of the basic version [25] of the original Boneh-Franklin IBE [6]. We can repeat the "twinning" idea and define the *twin BDH problem*, where one must compute $2\mathrm{bdh}(X_1, X_2, Y, W)$ for random X_1, X_2, Y, W, where we define

$$2\mathrm{bdh}(X_1, X_2, Y, W) := (\mathrm{bdh}(X_1, Y, W), \mathrm{bdh}(X_2, Y, W)).$$

The *strong twin BDH problem* is the same as the twin BDH problem, but the adversary has access to an oracle computing the predicate

$$2\mathrm{bdhp}(X_1, X_2, \hat{Y}, \hat{W}, \hat{Z}_1, \hat{Z}_2) := \quad 2\mathrm{bdh}(X_1, X_2, \hat{Y}, \hat{W}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2),$$

for $, \hat{Y}, \hat{W}, \hat{Z}_1, \hat{Z}_2$ of its choice. The *strong twin BDH assumption* states that the BDH problem is still hard, even with access to the decision oracle.

We will need a slight generalization of the trapdoor test in Theorem 2 to prove the following theorem. It is easy to check that Theorem 2 is still true if the elements $\hat{Y}, \hat{Z}_1, \hat{Z}_2$ are in a *different* cyclic group of the same order (we will take them in the range group of the pairing). With this observation, we can prove an analogue of Theorem 3.

Theorem 11. *Suppose $\mathcal{B}_{2\mathrm{bdh}}$ is a strong twin BDH adversary that makes at most Q_{d} queries to its decision oracle, and runs in time at most τ. Then there exists a BDH adversary $\mathcal{B}_{\mathrm{bdh}}$ with the following properties: $\mathcal{B}_{\mathrm{bdh}}$ runs in time at most τ, plus the time to perform $O(Q_{\mathrm{d}} \log q)$ group operations and some minor bookkeeping; moreover,*

$$\mathsf{Adv2BDH}_{\mathcal{B}_{2\mathrm{bdh}},\mathbb{G}} \leq \mathsf{AdvBDH}_{\mathcal{B}_{\mathrm{bdh}},\mathbb{G}} + \frac{Q_{\mathrm{d}}}{q}.$$

In addition, if $\mathcal{B}_{\mathrm{bdh}}$ does not output "failure," then its output is correct with probability at least $1 - 1/q$.

7.2 Twin Boneh-Franklin

Theorem 11 admits a simple analysis of the following IBE scheme, which we call the *twin Boneh-Franklin IBE scheme*. This scheme will use two hash functions, H (which outputs symmetric keys) and G (which outputs group elements), and a symmetric cipher (E, D). A master public key is a pair of group elements (X_1, X_2), where $X_i = g^{x_i}$ for $i = 1, 2$. The master private key is (x_1, x_2), which are selected at random from $\mathbb{Z}_q$ by the setup algorithm. The secret key for an identity $id \in \{0,1\}^*$ is $(S_1, S_2) = (\mathsf{G}(id)^{x_1}, \mathsf{G}(id)^{x_2})$. To encrypt a message m for identity id, one chooses $y \in \mathbb{Z}_q$ at random and sets

$$Y := g^y, \quad Z_1 := \hat{e}(\mathsf{G}(id), X_1)^y, \quad Z_2 := \hat{e}(\mathsf{G}(id), X_2)^y, \quad k := \mathsf{H}(id, Y, Z_1, Z_2), \quad c := \mathsf{E}_k(m).$$

The ciphertext is (Y, c). To decrypt using the secret key (S_1, S_2) for id, one computes

$$Z_1 := \hat{e}(S_1, Y), \quad Z_2 := \hat{e}(S_2, Y), \quad k := \mathsf{H}(id, Y, Z_1, Z_2), \quad m := \mathsf{D}_k(c).$$

We shall denote this scheme $\mathsf{IBE}_{2\mathrm{bdh}}$. Now we can essentially borrow the analysis of the original Boneh-Franklin scheme under the strong BDH assumption [25], except now we prove that the scheme is secure against chosen ciphertext attacks under the strong twin BDH assumption. By Theorem 11, we get that the above IBE scheme is CCA secure under the BDH assumption if the symmetric cipher is secure and the hash functions are treated as random oracles. This is captured in the following theorem.

Theorem 12. *Suppose* H *and* G *are modeled as random oracles. Further, suppose the DH assumption holds, and that the symmetric cipher* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is secure against chosen ciphertext attack. Then* $\mathsf{IBE}_{\mathrm{2bdh}}$ *is secure against chosen ciphertext attack.*

In particular, suppose $\mathcal{A}$ *is an adversary that carries out a chosen ciphertext attack against* $\mathsf{IBE}_{\mathrm{2bdh}}$ *in the random oracle model, and that* $\mathcal{A}$ *runs in time* τ*, and makes at most* Q_{h} *hash queries,* Q_{d} *decryption queries, and* Q_{id} *user secret key queries. Then there exists a BDH adversary* $\mathcal{B}_{\mathrm{bdh}}$ *and an adversary* $\mathcal{B}_{\mathrm{sym}}$ *that carries out a chosen ciphertext attack against* SE*, such that both* $\mathcal{B}_{\mathrm{bdh}}$ *and* $\mathcal{B}_{\mathrm{sym}}$ *run in time at most* τ*, plus the time to perform* $O((Q_{\mathrm{id}}+Q_{\mathrm{h}}+Q_{\mathrm{d}})\log q)$ *group operations; moreover,*

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{IBE}_{\mathrm{2bdh}}} \leq e \cdot (Q_{\mathrm{id}}+1) \cdot \left(\frac{2Q_{\mathrm{h}}+Q_{\mathrm{d}}}{q} + \mathsf{AdvBDH}_{\mathcal{B}_{\mathrm{bdh}},\mathbb{G}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}} \right).$$

Proof. As with our other proofs, we proceed with a sequence of games.

Game 0. Let Game 0 be the original IBE chosen ciphertext attack game, and let S_0 be the event that $\hat{b} = b$ in this game.

The challenger chooses the master private key (x_1, x_2) and gives the adversary the corresponding master public key (X_1, X_2) as normal. To track random oracle responses, the challenger uses two associative arrays L and K. L will store responses for G and K will store responses for H, and both will initially have all entries set to $\perp$. When processing a random oracle response, the adversary returns the corresponding entry if it is defined, and otherwise initializes it with an appropriate random value and returns that. Apart from this book keeping, the challenger runs Game 0 exactly as specified in the definition, and we have

$$\mathsf{AdvCCA}^{\mathrm{ro}}_{\mathcal{A},\mathsf{IBE}_{\mathrm{2bdh}}} = |\Pr[S_0] - 1/2|. \tag{28}$$

Game 1. Game 1 will be like Game 0, but now we change how the challenger processes queries to G. Now, in addition to inserting oracle responses into L, the challenger also "marks" some entries in the L array used to store G responses. On query $\mathsf{G}(\hat{id})$, in addition to the normal processing, with probability δ the challenger marks $L[\hat{id}]$. The challenger completely hides the marks from the adversary.

At the end of the game, the challenger looks at L and decides if it should abort the game. For each user secret key query that the adversary issued during the game, the challenger checks if the entry in L for that identity is marked. If any of them are marked, the challenger aborts the game. Finally it checks the entry $L[id]$, where id is the identity from the challenge query. If that entry is *not* marked, then the challenger aborts. Otherwise it proceeds normally.

Let S_1 be the event that $\hat{b} = b$ in Game 1 and F_1 be the event that the challenger aborts. Since the coins that determine F_1 are independent of the rest of the game, it follows that

$$\Pr[S_1] = \delta \cdot (1-\delta)^{Q_{\mathrm{id}}} \Pr[S_0],$$

and if we set $\delta = 1/(1+Q_{\mathrm{id}})$,

$$\Pr[S_1] \geq (e(1+Q_{\mathrm{id}}))^{-1} \Pr[S_0]. \tag{29}$$

Game 2 Game 2 will be like Game 1, except that now the challenger sets up some of the challenge ciphertext in advance. Before starting the game, it chooses a random symmetric key k, random $y \in \mathbb{Z}_q$ and random $W \in \mathbb{G}$, sets $Y := g^y$, $Z_1 := \hat{e}(W, X_1)^y$ and $Z_2 := \hat{e}(W, X_2)^y$.

Now the challenger uses these values in the rest of the game. When creating the challenge ciphertext, the challenger sets $K[id, Y, Z_1, Z_2] := k$ (overwriting the entry if it is already defined), computes $c := \mathsf{E}(k, m_b)$, and returns (Y, c).

For decryption queries, when the adversary asks for the decryption of $(\hat{Y}, \hat{c})$ under identity $\hat{id}$, if $\hat{id} = id$, $L[id]$ is marked, and $\hat{Y} = Y$, then the challenger uses k to decrypt $\hat{c}$. Otherwise, the challenger decrypts normally.

For the challenge query, the challenger uses k to compute $c := \mathsf{E}(k, m_b)$ and returns (Y, c).

Let S_2 be the event that $\hat{b} = b$ in Game 2. Since Game 2 and Game 1 only differ when the adversary manages to query $\mathsf{H}(id, Y, Z_1, Z_2)$ *before* the challenge query, and this event only happens if the adversary can guess Y, an independently chosen group element. Thus

$$|\Pr[S_2] - \Pr[S_1]| \leq Q_{\mathrm{H}}/q. \tag{30}$$

Game 3 Game 3 will include one simple change from Game 2: it no longer immediately stores the value k in K as described in Game 2. Instead, it leaves that entry unchanged, but still uses the k, Y, Z_1, Z_2 generated at the beginning of the game.

Let S_3 be the event that $\hat{b} = b$ in Game 3. Let F_{2bdh} be the event that the adversary queries H at (id, Y, Z_1, Z_2), where id is the identity used in the challenge ciphertext. Since Game 2 and Game 3 are exactly the same when F_{2bdh} does not occur, it follows that

$$|\Pr[S_3] - \Pr[S_2]| \leq \Pr[F_{\mathrm{2bdh}}]. \tag{31}$$

We claim that

$$\Pr[F_{\mathrm{2bdh}}] \leq \mathsf{Adv2BDH}_{\mathcal{B}_{\mathrm{2bdh}},\mathbb{G}}, \tag{32}$$

for an efficient strong twin BDH adversary $\mathcal{B}_{\mathrm{2bdh}}$ that makes $Q_{\mathrm{h}} + Q_{\mathrm{d}}$ decision oracle queries. We give a high level description of $\mathcal{B}_{\mathrm{2bdh}}$. $\mathcal{B}_{\mathrm{2bdh}}$ gets (X_1, X_2, Y, W) as input and begins to run Game 3, acting as the challenger for the adversary. Of course, it sets the master public key to (X_1, X_2) and uses (Y, c) as challenge ciphertext, where $c := \mathsf{E}_k(m_b)$, as in Game 3.

We need to describe how $\mathcal{B}_{\mathrm{2bdh}}$ answers queries for the random oracles and user secret keys. When the adversary requests $\mathsf{G}(\hat{id})$, if that entry gets marked, $\mathcal{B}_{\mathrm{2bdh}}$ chooses a new random $r \in \mathbb{Z}_q$, sets $L[\hat{id}] := Wg^r$, and gives Wg^r to the adversary. If the entry does not get marked, $\mathcal{B}_{\mathrm{2bdh}}$ returns g^r instead. (Note that $\mathcal{B}_{\mathrm{2bdh}}$ can respond with the corresponding user secret key for unmarked identities.) In either case, r is remembered for later.

When the adversary requests the user secret key for an unmarked identity $\hat{id}$, $\mathcal{B}_{\mathrm{2bdh}}$ retrieves the r used to generate the entry g^r in $L[\hat{id}]$, and returns (X_1^r, X_2^r). If the adversary requests the user secret key for a marked identity, $\mathcal{B}_{\mathrm{2bdh}}$ immediately aborts.

For H queries, $\mathcal{B}_{\mathrm{2bdh}}$ implements the same oracle patching idea used in the proof of Theorem 4. On query $\mathsf{H}(\hat{id}, \hat{Y}, \hat{Z}_1, \hat{Z}_2)$, $\mathcal{B}_{\mathrm{2bdh}}$ looks up $\hat{W}$ stored at $L[\hat{id}]$ and queries its decision oracle with $(\hat{Y}, \hat{W}, \hat{Z}_1, \hat{Z}_2)$, and marks the tuple as "good" or "bad" depending on the answer. If it finds a good tuple, it uses the corresponding key to decrypt ciphertexts with $\hat{Y}$. Otherwise, it generates a random symmetric key to use with those ciphertexts, and watches for a good tuple to come up as a hash query. When it sees one, it "patches" that query by returning the symmetric key generated earlier.

After the game ends, $\mathcal{B}_{2\text{bdh}}$ checks that the identity from the test query was unmarked. If not, $\mathcal{B}_{2\text{bdh}}$ aborts. Otherwise, it examines K and looks for a good entry of the form $K[id, Y, Z_1, Z_2]$ (where id and Y are from the test query). If it finds one, it looks up the $\hat{W} = Wg^r$ and corresponding r and outputs $(Z_1/\hat{e}(X_1, Y)^r, Z_2/\hat{e}(X_2, Y)^r)$. It is straightforward to check that $\mathcal{B}_{2\text{bdh}}$ solves the strong twin BDH problem whenever the event $F_{2\text{bdh}}$ would happen in Game 3.

Finally, in Game 3 the adversary is essentially playing the chosen ciphertext game against SE. Thus there is an adversary $\mathcal{B}_{\text{sym}}$ such that

$$|\Pr[S_1] - 1/2| = \mathsf{AdvCCA}_{\mathcal{B}_{\text{sym}},\mathsf{SE}}. \tag{33}$$

The theorem follows by combining (28)–(33). □

We remark that our ideas can also be applied to the IBE scheme from Sakai-Kasahara [31]. The resulting IBE scheme is more efficient, but its security can only be proved based on the (computational) q-BDHI assumption [5].

8 Relation to Shoup's DH self-corrector

In [32], Shoup presented a simple DH self-corrector, which implicitly contained our trapdoor test (our Theorem 2).[2] In this section, we describe Shoup's DH self-corrector, using the high-level notion of our trapdoor test.[3]

Let $\mathbb{G}$ be a group of prime order q with generator $g \in \mathbb{G}$. In general, a DH self-corrector works as follows. Suppose A is a probabilistic, polynomial-time algorithm that on a random input $(X, Y) \in \mathbb{G} \times \mathbb{G}$, outputs a list L of group elements, such that L contains $\text{dh}(X, Y)$ with non-negligible probability. A self-corrector C is a probabilistic, polynomial-time algorithm that uses A as a subroutine, so that for all inputs $(X, Y) \in \mathbb{G} \times \mathbb{G}$, it correctly computes $\text{dh}(X, Y)$ with all but negligible probability; that is, the output of C is a single group element (or possibly "failure"), which is an incorrect solution to the given instance of the DH problem with negligible probability.

Here is how we can construct C, using the trapdoor test, and an algorithm A, as above, as a subroutine. First, using the well-known random self reducibility property of the DH problem, along with standard amplification techniques, we can convert A into a probabilistic, polynomial-time algorithm A' that for all inputs (X, Y), computes a list L' of group elements, such that L' does not contain $\text{dh}(X, Y)$ with negligible probability. On input (X, Y), the self-corrector C runs as follows:

```
initialize the trapdoor test with X_1 := X,
      obtaining X_2 and a corresponding trapdoor
L_1 ← A'(X_1, Y)
L_2 ← A'(X_2, Y)
for each Z_1 in L_1 and each Z_2 in L_2 do
      if 2dhp(X_1, X_2, Y, Z_1, Z_2) then
            output Z_1 and halt
output "failure"
```

[2] Maurer and Wolf [26] also present a DH self-corrector, based, however, on completely different principles.

[3] Actually, we present a slightly less efficient, less general, but simpler, version of Shoup's corrector.

public system parameters: $g, U, V \in \mathbb{G}$
shared secret password: pw

P		Q
$x \leftarrow_R \mathbb{Z}_q, X \leftarrow g^x U^{\mathrm{pw}}$	$\xrightarrow{\quad X \quad}$	
		$y \leftarrow_R \mathbb{Z}_q,\ Y \leftarrow g^y V^{\mathrm{pw}}$ $Z \leftarrow (X/U^{\mathrm{pw}})^y$ $k \leftarrow \mathsf{H}(\mathrm{pw}, id_P, id_Q, X, Y, Z)$
$Z \leftarrow (Y/V^{\mathrm{pw}})^x$ $k \leftarrow \mathsf{H}(\mathrm{pw}, id_P, id_Q, X, Y, Z)$	$\xleftarrow{\quad Y \quad}$	
	session key: k	

Figure 1: Protocol SPAKE$_2$

If $2\mathrm{dhp}(X1, X_2, \cdot, \cdot, \cdot)$ is implemented using the trapdoor test, and q is large (which is the interesting case, of course), then it is clear that C makes a mistake with negligible probability.

9 Password authenticated Key Exchange

Abdalla and Pointcheval [2] presented a very efficient and elegant protocol for password authenticated key exchange (PAKE), called SPAKE$_2$. If users have weak passwords, it prevents offline dictionary attacks. Security is proved in the random oracle model, under the DH assumption. The protocol makes use of a group $\mathbb{G}$ of prime order q, a generator $g \in \mathbb{G}$, and a hash function H, which we model as a random oracle. The protocol has additional system parameters U and V, which are randomly chosen elements of $\mathbb{G}$. Furthermore, passwords pw are viewed as elements of $\mathbb{Z}_q$. Protocol SPAKE$_2$ is described in Figure 1. Both users compute the value $Z = \mathrm{dh}(X, Y)$, and then compute the session key as $k = \mathsf{H}(\mathrm{pw}, id_P, id_Q, X, Y, Z)$.

Often, users play very distinct roles. One user may be a *client*, which obtains the password by keyboard entry, while the other is a *server*, which is a machine that keeps a *password file*, containing information for each client who is authorized to access the server. A type of attack that we would like to provide some defense against is a *server compromise*, in which an adversary obtains the server's password file. Given the password file, the adversary can certainly impersonate the server; however, we would like to make it as hard as possible for the adversary to impersonate a client, and gain unauthorized access to the server.

Given the password file, an adversary can always mount an offline dictionary attack to recover a given client's password; ideally, this would be all the adversary could do; in particular, it should be infeasible to recover a strong password.

Consider again protocol SPAKE$_2$. The roles of the two users in that protocol are quite symmetric, but for concreteness, let us say that P is the client, and Q is the server. In the most obvious implementation, Q would explicitly store the password pw in the password file. Clearly, this implementation is undesirable, as an adversary that compromises the server immediately recovers the password.

public system parameters: $g, U, V \in \mathbb{G}$
password: pw, $(\pi_0, \pi_1) := \mathsf{G}(\text{pw}, id_P, id_Q)$

P		Q
secret: π_0, π_1		secret: $\pi_0, L := g^{\pi_1}$
$x \leftarrow_R \mathbb{Z}_q, X \leftarrow g^x U^{\pi_0}$	$\xrightarrow{\quad X \quad}$	
		$y \leftarrow_R \mathbb{Z}_q,\ Y \leftarrow g^y V^{\pi_0}$ $Z \leftarrow (X/U^{\pi_0})^y,\ N \leftarrow L^y$ $k \leftarrow \mathsf{H}(\pi_0, X, Y, Z, N)$
$Z \leftarrow (Y/V^{\pi_0})^x,\ N \leftarrow (Y/V^{\pi_0})^{\pi_1}$ $k \leftarrow \mathsf{H}(\pi_0, X, Y, Z, N)$	$\xleftarrow{\quad Y \quad}$	
	session key: k	

Figure 2: Protocol SPAKE_2^+

While there generic transformations that can transform any PAKE protocol into a PAKE protocol that provides protection against security compromise (see [15]), we present a protocol, SPAKE_2^+, which does so more directly. With this protocol, if the server is compromised, the best an adversary can do to impersonate a client is an offline dictionary attack.

In addition to SPAKE_2, protocol SPAKE_2^+ employs another hash function G, which has range $\mathbb{Z}_q \times \mathbb{Z}_q$, and which we also model as a random oracle. Let pw be the password shared between client P and server Q, which is an arbitrary bit string. The protocol is described in Figure 2. Here, the client stores (π_0, π_1), while the server stores (π_0, L), where $L := g^{\pi_1}$ and

$$(\pi_0, \pi_1) := \mathsf{G}(\text{pw}, id_P, id_Q).$$

Of course, the client can derive (π_0, π_1) from pw. Both users compute the values $Z = \text{dh}(X, Y) = g^{xy}$ and $N = g^{\pi_1 y}$, and then compute the session key as $k = \mathsf{H}(\pi_0, X, Y, Z, N)$.

It is not hard to argue that protocol SPAKE_2^+ offers the same level of security as protocol SPAKE_2 under normal conditions, when the server is not compromised. However, consider what happens if the server Q is compromised in protocol SPAKE_2^+, and the adversary obtains π_0 and L. At this point, the adversary could attempt an offline dictionary attack, as follows: evaluate G at points (pw', id_P, id_Q) for various passwords pw', trying to find pw' such that $\mathsf{G}(\text{pw}', id_P, id_Q) = (\pi_0, \cdot)$. If this succeeds, then with high probability, $\text{pw}' = \text{pw}$, and the adversary can easily impersonate the client.

The key property we want to prove is the following: if the above dictionary attack fails, then under the DH assumption, the adversary cannot impersonate the client. Intuitively, to impersonate the client, the adversary will have to compute $\text{dh}(L, Y')$, where L is the value g^{π_1} stored on the server, and $Y' := g^y$ is a random group element generated by the server. If the dictionary attack fails, then the adversary does not see π_1. However, he may also interact with the client, who uses the value π_1 in its calculation of N. To prove that the adversary cannot compute $\text{dh}(L, Y')$, one would normally have to appeal to the strong DH assumption. However, because the hash happens to already include Z in addition to N, it is not hard to prove, using Theorem 2, that the (ordinary)

DH assumption suffices.

References

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer-Verlag, Berlin, Germany, April 2001.

[2] Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 191–208. Springer-Verlag, Berlin, Germany, February 2005.

[3] Joonsang Baek, Byoungcheon Lee, and Kwangjo Kim. Secure length-saving ElGamal encryption under the computational Diffie-Hellman assumption. In *ACISP 2000*, pages 49–58, 2000.

[4] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, Berlin, Germany, May 2004.

[5] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer-Verlag, Berlin, Germany, May 2004.

[6] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, Berlin, Germany, August 2001.

[7] Xavier Boyen. Miniature CCA2 PK encryption : Tight security without redundancy. In *Advances in Cryptology—ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 485–501. Berlin: Springer-Verlag, 2007.

[8] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In *ACM CCS 05*, pages 320–329. ACM Press, November 2005.

[9] Jean-Sébastien Coron, Helena Handschuh, Marc Joye, Pascal Paillier, David Pointcheval, and Christophe Tymen. GEM: A generic chosen-ciphertext secure encryption method. In Bart Preneel, editor, *CT-RSA 2002*, volume 2271 of *LNCS*, pages 263–276. Springer-Verlag, Berlin, Germany, February 2002.

[10] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, Berlin, Germany, August 1998.

[11] 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.

[12] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer-Verlag, Berlin, Germany, August 1999.

[14] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure Hashed Diffie-Hellman over non-DDH groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 361–381. Springer-Verlag, Berlin, Germany, May 2004.

[15] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, *CRYPTO 2006*, LNCS, pages 142–159. Springer-Verlag, Berlin, Germany, August 2006.

[16] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[17] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.

[18] Shai Halevi. EME*: Extending EME to handle arbitrary-length messages with associated data. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 315–327. Springer-Verlag, Berlin, Germany, December 2004.

[19] Shai Halevi and Phillip Rogaway. A tweakable enciphering mode. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 482–499. Springer-Verlag, Berlin, Germany, August 2003.

[20] Shai Halevi and Phillip Rogaway. A parallelizable enciphering mode. In Tatsuaki Okamoto, editor, *CT-RSA 2004*, volume 2964 of *LNCS*, pages 292–304. Springer-Verlag, Berlin, Germany, February 2004.

[21] Goichiro Hanaoka and Kaoru Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. Cryptology ePrint Archive, Report 2008/094, 2008. `http://eprint.iacr.org/`.

[22] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In Alfred Menezes, editor, *Advances in Cryptology, Proceedings of CRYPTO 2007*, Lecture Notes in Computer Science, pages 553–571. Springer-Verlag, 2007. Full version available from `http://eprint.iacr.org/2007/288`.

[23] Caroline Kudla and Kenneth G. Paterson. Modular security proofs for key agreement protocols. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 549–565. Springer-Verlag, Berlin, Germany, December 2005.

[24] Kaoru Kurosawa and Toshihiko Matsuo. How to remove MAC from DHIES. In *ACISP 2004*, pages 236–247, 2004.

[25] Benoît Libert and Jean-Jacques Quisquater. Identity based encryption without redundancy. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 285–300. Springer-Verlag, Berlin, Germany, June 2005.

[26] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 268–282. Springer-Verlag, Berlin, Germany, August 1996.

[27] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[28] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, Berlin, Germany, February 2001.

[29] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–175. Springer-Verlag, Berlin, Germany, April 2001.

[30] Duong Hieu Phan and David Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In Helena Handschuh and Anwar Hasan, editors, *SAC 2004*, volume 3357 of *LNCS*, pages 182–197. Springer-Verlag, Berlin, Germany, August 2004.

[31] Ryuichi Sakai and Masao Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. `http://eprint.iacr.org/`.

[32] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer-Verlag, Berlin, Germany, May 1997.

[33] R. Steinfeld, J. Baek, and Y. Zheng. On the necessity of strong assumptions for the security of a class of asymmetric encryption schemes. In *ACISP 2002*, volume 2384 of *LNCS*, pages 241–256. Springer-Verlag, Berlin, Germany, 2002.

A The proof of security for $\mathsf{PKE}_{\mathrm{cs}}$

In this section we show how the proof of the original analysis of $\mathsf{PKE}_{\mathrm{cs}}$ can be viewed in terminology.

We first recall the original scheme given by Cramer and Shoup, which we will denote $\mathsf{PKE}_{\mathrm{cs}}$. The schemes uses a hash function $\mathsf{T} : \mathbb{G} \to \mathbb{Z}_q$ and a symmetric cipher $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$. For simplicity we assume that the cipher's secret key consists of a random group member in $\mathbb{G}$, but this assumption can be removed using standard techniques, c.f. [11].

A secret key consists of four random elements of $\mathbb{Z}_q$, denoted $x_1, x_2, \widetilde{x}_2, x_3$, and the corresponding public key consists of four group elements $X_1 = g^{x_1}, X_2 = g^{x_2}, \widetilde{X}_2 = g^{\widetilde{x}_2}, X_3 = g^{x_3}$. To encrypt a message m, one chooses y at random from $\mathbb{Z}_q$, and computes

$$Y := g^y, \quad Z_1 := X_1^y, \quad t := \mathsf{T}(Y, Z_1), \quad Z_2 := (X_2^t \widetilde{X}_2)^y, \quad k := X_3^y, \quad c := \mathsf{E}_k(m).$$

The ciphertext is (Y, Z_1, Z_2, c). To decrypt $(\hat{Y}, \hat{Z}_1, \hat{Z}_2, \hat{c})$, one computes $\hat{t} := \mathsf{T}(\hat{Y}, \hat{Z}_1)$ and tests if

$$\hat{Y}^{x_1} \stackrel{?}{=} \hat{Z}_1 \quad \text{and} \quad \hat{Y}^{\hat{t}x_2 + \widetilde{x}_2} \stackrel{?}{=} \hat{Z}_2.$$

If not, reject. Otherwise, compute $\hat{k} := Y^{x_3}$ and output $\mathsf{D}_{\hat{k}}(\hat{c})$.

Theorem 13 ([11]). *Suppose* T *is a target collision resistant hash function. Further, suppose the DDH assumption holds, and that the symmetric cipher* $\mathsf{SE} = (\mathsf{E}, \mathsf{D})$ *is secure against chosen ciphertext attack. Then* $\mathsf{PKE}_{\mathrm{cs}}$ *is secure against chosen ciphertext attack.*

In particular, suppose $\mathcal{A}$ *is an adversary that carries out a chosen ciphertext attack against* $\mathsf{PKE}_{\mathrm{cs}}$ *and that* $\mathcal{A}$ *runs in time* τ*, and makes at most* Q_{d} *decryption queries. Then there exists a DDH adversary* $\mathcal{B}_{\mathrm{ddh}}$*, an adversary* $\mathcal{B}_{\mathrm{sym}}$ *that carries out a chosen ciphertext attack against* SE*, and a TCR adversary* $\mathcal{B}_{\mathrm{tcr}}$ *such that both* $\mathcal{B}_{\mathrm{ddh}}$*,* $\mathcal{B}_{\mathrm{sym}}$ *and* $\mathcal{B}_{\mathrm{tcr}}$ *run in time at most* τ*, plus the time to perform* $O(Q_{\mathrm{d}} \log q)$ *group operations; moreover,*

$$\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\widetilde{\mathrm{cs}}}} \leq \mathsf{AdvDDH}_{\mathcal{B}_{\mathrm{ddh}},\mathbb{G}} + \mathsf{AdvCCA}_{\mathcal{B}_{\mathrm{sym}},\mathsf{SE}} + \mathsf{AdvTCR}_{\mathcal{B}_{\mathrm{tcr}},\mathsf{T}} + \frac{Q_{\mathrm{d}}}{q}.$$

Proof. As usual, our proof consists of a sequence of games. For each i, let S_i be the event that $\hat{b} = b$ in Game i.

Game 0. Let Game 0 be the chosen ciphertext game played by $\mathcal{A}$ against $\mathsf{PKE}_{\mathrm{cs}}$. Then

$$\mathsf{AdvCCA}_{\mathcal{A},\mathsf{PKE}_{\mathrm{cs}}} = |\Pr[S_0] - 1/2|. \tag{34}$$

Game 1. Game 1 is like Game 0, except that if the adversary issues a decryption query containing $(\hat{Y}, \hat{Z}_1) \neq (Y, Z_1)$ such that $\mathsf{T}(\hat{Y}, \hat{Z}_1) = t$, then the game aborts. It is standard to show (since Y and Z_1 can be chosen ahead of time) that there exists an adversary $\mathcal{B}_{\mathrm{tcr}}$ such that

$$|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{AdvTCR}_{\mathcal{B}_{\mathrm{tcr}},\mathsf{T}}. \tag{35}$$

Game 2. Let Game 2 is like Game 1, except that now the challenger sets up some values ahead of time and uses them during the game, but does not change the distribution of the game at all. At the start of the game, the challenger chooses $y, x_1 \leftarrow_R \mathbb{Z}_q$ and computes

$$Y := g^y, \quad X_1 := g^{x_1}, \quad Z_1 := X_1^y, \quad t := \mathsf{T}(Y, Z_1).$$

It then chooses $x_2, r \leftarrow_R \mathbb{Z}_q$ and computes

$$X_2 := g^{x_2}, \quad \widetilde{X}_2 := g^r X_2^{-t}.$$

It chooses $x_3 \leftarrow_R \mathbb{Z}_q$ and computes $X_3 := g^{x_3}$ normally, and sets the public key to $(X_1, X_2, \widetilde{X}_2, X_3)$. To compute the challenge ciphertext, the challenger sets

$$Z_2 := Y^r, \quad k := Y^{x_3} \quad c := \mathsf{E}_k(m_b)$$

and returns (Y, Z_1, Z_2, c).

We also change the way the challenger performs the consistency check in the decryption oracle. On input $(\hat{Y}, \hat{Z}_1, \hat{Z}_2, \hat{c})$, if $(\hat{Y}, \hat{Z}_1) = (Y, Z_1)$, then the challenger further checks if $\hat{Z}_2 = Z_2$. If so, it uses k to decrypt $\hat{c}$; otherwise it rejects the query. If $(\hat{Y}, \hat{Z}_1) \neq (Y, Z_1)$, it computes $\hat{t} := \mathsf{T}(\hat{Y}, \hat{Z}_1)$ and $\bar{Z}_2 := (\hat{Z}_2/g^r)^{\frac{1}{\hat{t}-t}}$. Then it tests if

$$\hat{Z}_1 \stackrel{?}{=} \hat{Y}^{x_1} \quad \text{and} \quad \bar{Z}_2 \stackrel{?}{=} \hat{Y}^{x_2}. \tag{36}$$

If this does not hold, it rejects. Otherwise it computes $\hat{k} := \hat{Y}^{x_3}$ and outputs $\mathsf{D}_{\hat{k}}(\hat{c})$.

We claim that the distribution of Game 2 is exactly the same as the distribution of Game 1. This follows by observing that the public key and challenge ciphertext are computed correctly, and that the decryption consistency check works as before. Then we have that

$$\Pr[S_2] = \Pr[S_1]. \tag{37}$$

Game 3. Let Game 3 is like Game 2, except we change how the challenger computes X_3 and the values that depend on X_3. The challenger now picks $x_3, u_3 \leftarrow_R \mathbb{Z}_q$ and sets $X_3 := g^{x_3} X_1^{u_3}$. In the challenge ciphertext it computes $k := Y^{x_3} Z_1^{u_3}$, and in decryption queries it computes $\hat{k} := \hat{Y}^{x_3} \hat{Z}_1^{u_3}$. These changes do not affect the distribution of the game because X_3 is still independent of X_1, and the rest of the values are computed correctly.

$$\Pr[S_3] = \Pr[S_2]. \tag{38}$$

Game 4. Let Game 4 be exactly like Game 3, except that Z_1 is set to a random element other than X_1. We claim that there exists an efficient adversary $\mathcal{B}_{\text{2ddh}}$ such that

$$|\Pr[S_4] - \Pr[S_3]| \leq \mathsf{Adv2DDH}_{\mathcal{B}_{\text{2ddh}},\mathbb{G}} + 1/q. \tag{39}$$

$\mathcal{B}_{\text{2ddh}}$ gets (X_1, X_2, Y, Z_1) as input and simply simulates Game 3 for $\mathcal{A}$. It selects x_3, u_3 itself, but the rest of the discrete logs are not necessary for the experiment. For the consistency check in equation (36), $\mathcal{B}_{\text{2ddh}}$ uses its 2dhp oracle. The claim follows by observing that $\mathcal{B}_{\text{2ddh}}$ exactly simulates Game 3 if $Z_1 = \text{dh}(X_1, Y)$ or Game 4 if Z_1 is random.

Game 5. Let Game 5 be exactly like Game 4, except that k is set to a random group element. We claim that

$$\Pr[S_5] = \Pr[S_4]. \tag{40}$$

To prove this, it is sufficient to show that in Game 3 k is uniform when conditioned on the the values in the the public key and the challenge ciphertext, which determine behavior of the decryption oracle. This argument is exactly as in the original proof.

In Game 5, $\mathcal{A}$ is playing a chosen-ciphertext game against SE, and hence there exists an adversary $\mathcal{B}_{\text{sym}}$ such that

$$|\Pr[S_5] - 1/2| \leq \mathsf{AdvCCA}_{\mathcal{B}_{\text{sym}},\mathsf{SE}}. \tag{41}$$

The theorem follows by collecting equations (34), (35), (37), (38), (39), (40), and (41). □