An Efficient Two-Party Public Key Cryptosystem Secure against Adaptive Chosen Ciphertext Attack

Philip MacKenzie

Bell Laboratories, Lucent Technologies
Murray Hill, NJ 07974, USA
philmac@lucent.com

Abstract. We propose an efficient two-party public key cryptosystem that is secure against adaptive chosen ciphertext attack, based on the hardness of Decision Diffie-Hellman (DDH). Specifically, we show that the two parties together can decrypt ciphertexts, but neither can alone. Our system is based on the Cramer-Shoup cryptosystem. Previous results on efficient threshold cryptosystems secure against adaptive chosen ciphertext attack required either (1) a strict majority of uncorrupted decryption servers, and thus do not apply to the two-party scenario, or (2) the random oracle assumption, and thus were not proven secure in the "standard" model.

1 Introduction

In this paper we present an efficient and provably secure protocol by which alice and bob, each holding a share of a Cramer-Shoup [15] private key, can jointly decrypt a ciphertext, but such that neither alice nor bob can decrypt a ciphertext alone. Of course, protocols for generic secure two-party computation (e.g., [43]) could be used to perform this decryption operation, but here we present a more efficient protocol to solve this particular problem. To our knowledge, this is the first practical and provably secure protocol for a two-party Cramer-Shoup cryptosystem.

In addition to being an important theoretical question, our interest in a two-party Cramer-Shoup cryptosystem is motivated by some very practical applications. One is that it could be used for a secure distributed third-party decryption service, which requires the joint agreement by two parties to decrypt a ciphertext. For example, this may be used to provide added security to (1) a key recovery system by law enforcement (e.g., [37]), or (2) an "offline trusted third party" system in a fair exchange protocol (e.g., [2]).

Another application (and our main motivation) is related to the work of [35] on techniques by which a device that performs private key operations (signatures or decryptions) in networked applications, and whose local private key is activated with a password or PIN, can be immunized against offline dictionary

Y.G. Desmedt (Ed.): PKC 2003, LNCS 2567, pp. 47–61, 2003.

attacks in case the device is captured. Briefly, this goal is achieved by involving a remote server in the device's private key computations, essentially sharing the cryptographic computation between the device and the server. The work of [35] showed how to accomplish this for the case of RSA functions and certain discrete-log-based functions, with DSA signatures and Cramer-Shoup decryptions being notable exceptions. These exceptions were due to the fact that there were no known provably secure protocols for a two-party DSA signature system or a two-party Cramer-Shoup cryptosystem. The work of [36] on two-party DSA signatures filled this gap for DSA signatures, although provable security was only obtained in the random oracle model based on the hardness of Decision Composite Residuosity [38] and Strong RSA [5], along with the standard security of DSA signatures.

The work in this paper fills this gap with respect to the Cramer-Shoup cryptosystem. Moreover, the cryptosystem presented here requires no extra assumptions beyond those required for the original Cramer-Shoup cryptosystem, and it is proven secure in the standard model (without random oracles). To achieve this we introduce some novel techniques, including the use of homomorphic encryptions of partial Cramer-Shoup decryption subcomputations (rather than encryptions of partial keys as in the two-party DSA signature system), and special three-move Σ-protocols for proving consistency (rather than non-interactive zero-knowledge proofs using random oracles as in the two-party DSA signature system). These Σ-protocols are especially noteworthy in that (1) they are not required to be (fully) zero-knowledge, and are used as proofs of consistency rather than proofs of knowledge, and thus they can be used in a concurrent setting (since neither simulation of provers nor extraction of witnesses is needed), and (2) their secure *use* relies in a fundamental way on the hardness of DDH, though their soundness and (honest-verifier) zero-knowledge properties do not. We will further discuss these technical issues in Section 5.

2 Related Work

The two-party Cramer-Shoup cryptosystem falls into the general category of threshold cryptography. Early work in the field is due to Boyd [7], Desmedt [19], Croft and Harris [17], Frankel [24], and Desmedt and Frankel [20]. Work in threshold cryptography for discrete-log based cryptosystems includes, for example, Desmedt and Frankel [20], Hwang [32], Pedersen [40], Cerecedo *et al.* [12], Harn [30], Langford [34], Gennaro *et al.* [29], Park and Kurosawa [39], Herzberg *et al.* [31], and Frankel *et al.* [26].

There have been previous proposals for threshold cryptosystems secure against adaptive chosen ciphertext attack, namely, Shoup and Gennaro [42], Canetti and Goldwasser [10], Abe [1], Jarecki and Lysyanskaya [33], and Fouque and Pointcheval [23]. They all assume the adversary corrupts t out of n decryption servers. Both the Shoup and Gennaro scheme and the Fouque and Pointcheval scheme may be used in the two-party case ($t = 1$, $n = 2$) if one is only concerned with security and not robustness, but they also both use the

non-standard assumption that hashes are modeled as random oracles ([8, 11] give arguments for and against this assumption). In this paper we are concerned with schemes that may be proven secure without using random oracles.

The remaining proposals are all based on the Cramer-Shoup cryptosystem. Canetti and Goldwasser assume there are users that wish to have messages decrypted, and that the servers do not communicate with each other, but only with the users. Then they show a secure system for $n > 2t$, a secure and robust system for $n > t^2$, and a secure and robust system for $n > 2t$ if the users are given extra per-ciphertext robustness information. Abe, and Jarecki and Lysyanskaya allow the servers to communicate with each other and present secure and robust systems for $n > 2t$. Note that none of these results apply to the scenario of this paper, i.e., $t = 1$ and $n = 2$. In fact, it is often the case that threshold cryptosystems (assuming a strict minority of corrupted players) are developed before the corresponding two-party cryptosystems. For example, threshold DSA [12, 34, 29] was developed before two-party DSA [36], and threshold RSA key generation [4, 25] was developed before two-party RSA key generation [28, 41].

3 Preliminaries

Security parameter Let κ be the cryptographic security parameter used for, e.g., hash functions and discrete log group orders; reasonable values today may be $\kappa = 160$ or $\kappa = 256$.

Notation and definitions We use $(a, b) \times (c, d)$ to mean elementwise multiplication, i.e., (ac, bd). We use $(a, b)^r$ to mean elementwise exponentiation, i.e., (a^r, b^r). For a tuple V, the notation $V[j]$ means the jth element of V.

Let G_q denote a finite (cyclic) group of prime order q, where $|q| = \kappa$. Let g be a generator of G_q, and assume it is included in the description of G_q. Note that in the following definitions and descriptions, we will abuse notation slightly and let G_q denote its own description. For instance, when we say the input to a function is G_q, we mean that the input is the description of the group G_q.

Encryption schemes An *encryption scheme* $\mathcal{E}$ is a triple $(G_{\mathcal{E}}, E, D)$ of algorithms, the first two being probabilistic polynomial-time, and the last being deterministic polynomial time. $G_{\mathcal{E}}$ takes as input[1] G_q and outputs a public key pair (pk, sk), i.e., $(pk, sk) \leftarrow G_{\mathcal{E}}(G_q)$. E takes a public key pk and a message m as input and outputs an encryption c for m; we denote this $c \leftarrow E_{pk}(m)$. D takes a ciphertext c and a secret key sk as input and returns either a message m such that c is a valid encryption of m under the corresponding public key, if such an m exists, and otherwise returns an arbitrary value.

[1] For convenience, instead of using input 1^κ, we will simply use a fixed group G_q with $|q| = \kappa$ and define encryption schemes over this group.

Now we define the Cramer-Shoup encryption scheme [15, 16] using a fixed universal one-way hash function H and over a fixed group G_q, in which solving DDH is difficult.[2]

$G_{CS}(G_q)$**:** Let g be the generator of G_q (implicitly included in the description of G_q). Generate $g_2 \stackrel{R}{\leftarrow} G_q$ and $a, b, c, d, e \stackrel{R}{\leftarrow} \mathbb{Z}_q$, and set $U \leftarrow g^a(g_2)^b$, $V \leftarrow g^c(g_2)^d$, and $W \leftarrow g^e$. Let the public key be $<g, g_2, U, V, W>$ and the secret key be $<a, b, c, d, e>$.

$E_{<g,g_2,U,V,W>}(m)$**:** Generate $r \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and compute $x \leftarrow g^r$, $y \leftarrow (g_2)^r$, $w \leftarrow W^r m$, $\sigma \leftarrow H(x, y, w)$, and $v \leftarrow U^r V^{r\sigma}$. Return $<x, y, w, v>$ as the ciphertext.

$D_{<a,b,c,d,e>}(<x, y, w, v>)$**:** Generate $\sigma \leftarrow H(x, y, w)$. If $v \neq x^{a+c\sigma} y^{b+d\sigma}$, return $\perp$, else return w/x^e.

Canetti and Goldwasser [10] give a variation of this protocol in which the decryption algorithm $D_{<a,b,c,d,e>}(<x, y, w, v>)$ generates σ as above, but then generates $s \stackrel{R}{\leftarrow} \mathbb{Z}_q$ and returns $w/(x^e(v/v')^s)$, where $v' = x^{a+c\sigma} y^{b+d\sigma}$. One can see that for invalid encryptions (those in which the original D function returns $\perp$) the new decryption function will return a completely random value, and for all other encryptions, the new decryption function returns the same value as the original. Our two-party protocol will actually perform the Canetti-Goldwasser variation of the decryption procedure.

System model Our system includes two parties, alice and bob, who obtain public and secret data through a trusted initialization procedure. Here we will simply assume alice and bob receive all their public and secret data from a trusted party. After initialization, communication between alice and bob occurs in *sessions* (or decryption protocol runs), one per ciphertext that they decrypt together. alice plays the role of session initiator in our decryption protocol. That is, alice receives requests to decrypt ciphertexts, and communicates with bob to decrypt these ciphertexts. We presume that each message between alice and bob is implicitly labeled with an identifier for the session to which it belongs. Multiple decryption sessions may be executed concurrently.

The adversary in our protocol controls the network, inserting and manipulating communication as it chooses. In addition, it takes one of two forms: an alice-compromising adversary that has perpetual read access to the private storage and computation of alice, and a bob-compromising adversary that has perpetual read access to the private storage and computation of bob.

We note that a proof of security in this two-party system extends to a proof of security in an n-party system in a natural way, assuming the adversary decides which parties to compromise before any session begins. The basic idea is to guess for which pair of parties the adversary decrypts a ciphertext without the assistance of the non-corrupted party, and focus the simulation proof on those

[2] Note that one possible group G_q may be found by generating a large prime p such that q divides $p - 1$, and letting G_q be the subgroup of order q in $\mathbb{Z}_p^*$.

two parties, running all other parties as in the real protocol. The only consequence is a factor of roughly n^2 lost in the reduction argument from the security of the encryption scheme.

Labeled ciphertexts Note that our scenario in which alice decides on which ciphertexts to decrypt is motivated by the systems in [35]. This removes the need to include in our model separate users that communicate with alice and bob to obtain decryptions, and allows us not to have to change the encryption scheme to use explicit labels (see [42]). Of course, the Cramer-Shoup encryption scheme does allow an easy way to introduce labels, and this could be done in our protocol also.

4 Definition and Basic Theory of Σ-Protocols

Our two-party decryption system in Section 5 uses special types of Σ-protocols to deal with malicious adversaries, so here we overview the basic definitions and properties of Σ-protocols [14, 13]. (This section may be skipped if one is only interested in the high-level design of our two-party decryption system, and in particular, a design that is only secure against so-called "honest-but-curious" adversaries.)

First we start with some definitions and notation. Let $R = \{(x, w)\}$ be a binary relation and assume that for some given polynomial $p(\cdot)$ it holds that $|w| \leq p(|x|)$ for all $(x, w) \in R$. Furthermore, let R be testable in polynomial time. Let $L_R = \{x : \exists w, (x, w) \in R\}$ be the *language* defined by the relation, and for all $x \in L_R$, let $W_R(x) = \{w : (x, w) \in R\}$ be the *witness set* for x. For any NP language L, note that there is a natural *witness relation* R containing pairs (x, w) where w is the witness for the membership of x in L, and that $L_R = L$.

Now we define a Σ-protocol (A, B) to be a three move interactive protocol between a probabilistic polynomial-time prover A and a probabilistic polynomial-time verifier B, where the prover acts first. The verifier is only required to send random bits as a challenge to the prover. For some $(x, w) \in R$, the common input to both players is x while w is private input to the prover. For such given x, let (a, c, z) denote the conversation between the prover and the verifier. To compute the first and final messages, the prover invokes efficient algorithms $a(\cdot)$ and $z(\cdot)$, respectively, using (x, w) and random bits as input. Using an efficient predicate $\phi(\cdot)$, the verifier decides whether the conversation is accepting with respect to x. The relation R, the algorithms $a(\cdot)$, $z(\cdot)$ and $\phi(\cdot)$ are public. The length of the challenges is denoted t_B, and we assume that t_B only depends on the length of the common string x. (In the following, we will always use challenges randomly drawn from $\mathbb{Z}_q$.)

We will need to broaden this definition slightly, to deal with cheating provers. We will define $\hat{L}_R$ to be the input language, with the properties that $L_R \subseteq \hat{L}_R$, and that membership in $\hat{L}_R$ may be tested in polynomial time. We implicitly assume B only executes the protocol if the common input $x \in \hat{L}_R$.

All Σ-protocols presented here will satisfy the following security properties:

1. *weak special soundness*: Let (a, c, z) and (a, c', z') be two conversations, that are accepting for some given x. If $c \neq c'$, then $x \in L_R$.[3] The pair of accepting conversations (a, c, z) and (a, c', z') with $c \neq c'$ is called a *collision* .
2. *special honest verifier zero knowledge* (special HVZK): There is a (probabilistic polynomial time) simulator M that on input $x \in L_R$ generates accepting conversations with the exact same distribution as when A and B execute the protocol on common input x (and A is given a witness w for x), and B indeed honestly chooses its challenges uniformly at random. The simulator is special in the sense that it can additionally take a random string c as input, and output an accepting conversation for x where c is the challenge. In fact, we will require the simulator to have this special property for not only $x \in L_R$, but also any $x \in \hat{L}_R$.

A simple but important fact (see [14]) is that if a Σ-protocol is HVZK, the protocol is perfectly *witness indistinguishable* (WI) [22]. Although HVZK by itself is defined with respect to a very much restricted verifier, i.e. an honest one, this means that if for a given instance x there are at least two witnesses w, then even an arbitrarily powerful and malicious verifier cannot distinguish which witness the prover uses.

In our results to follow, we need a particular, simple instance of the main theorem from [14]. Specifically, we use a slight generalization of a corollary in [14] which enables a prover, given two relations (R_1, R_2), values $(x_1, x_2) \in \hat{L}_{R_1} \times \hat{L}_{R_2}$, and corresponding 3-move Σ-protocols $((A_1, B_1), (A_2, B_2))$, to present a 3-move Σ-protocol (A_{or}, B_{or}) for proving the existence of a w such that either $(x_1, w) \in R_1$ or $(x_2, w) \in R_2$. We call this the "OR" protocol for $((A_1, B_1), (A_2, B_2))$,

For a relation R, let $\Sigma[R]$ denote a Σ-protocol over R. For a predicate P, let $\Sigma[P]$ denote $\Sigma[R]$ for the relation R defined by P, with public values defined by P. Furthermore, let $L_P = L_R$ and $\hat{L}_P = \hat{L}_R$, for the relation R defined by P. Let $\Sigma[X, Y]$ denote the "OR" protocol for $(\Sigma[X], \Sigma[Y])$.

5 S-CS System

In this section we present a new system called S-CS by which alice and bob can jointly decrypt Cramer-Shoup ciphertexts.

Our main motivating application naturally admits a trusted party for initializing the system (see [35]), so we will focus on that case.[4] Specifically, we

[3] Often these protocols are assumed to satisfy *special soundness*: On input x and those two conversations, a witness w such that $(x, w) \in R$ can be computed efficiently. We do not need special soundness for our results.

[4] Alternatively, one could build a distributed initialization protocol involving only alice and bob, and no trusted center. To achieve provable security, this initialization would have to be executed in a sequential manner prior to any decryption sessions, even though the decryption sessions themselves may be executed concurrently with respect to each other. Details are beyond the scope of this paper.

assume a trusted party is given a (public) group G_q with generator g and generates a Cramer-Shoup public key along with secret values for alice and bob to allow decryption:

$$\begin{aligned} g_2 &\stackrel{R}{\leftarrow} G_q, \\ a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, e_1, e_2 &\stackrel{R}{\leftarrow} \mathbb{Z}_q, \\ <U_1, U_2> &\leftarrow <g^{a_1}(g_2)^{b_2}, g^{a_2}(g_2)^{b_2}>, \\ <V_1, V_2> &\leftarrow <g^{c_1}(g_2)^{d_2}, g^{c_2}(g_2)^{d_2}>, \\ <W_1, W_2> &\leftarrow <g^{e_1}, g^{e_2}>, \\ \beta_1, \beta_2 &\stackrel{R}{\leftarrow} \mathbb{Z}_q, \\ <h_1, h_2> &\leftarrow <g^{\beta_1}, g^{\beta_2}>, \\ D_1, D_2, D_3, D'_1, D'_2, D'_3 &\stackrel{R}{\leftarrow} G_q. \end{aligned}$$

The trusted party gives alice the values $<a_1, b_1, c_1, d_1, e_1, \beta_1>$, gives bob the values $<a_2, b_2, c_2, d_2, e_2>$, and gives both alice and bob the values

$$<g, g_2, U_1, U_2, V_1, V_2, W_1, W_2, h_1, h_2, D_1, D_2, D_3, D'_1, D'_2, D'_3>.$$

Letting $U \leftarrow U_1U_2$, $V \leftarrow V_1V_2$ and $W \leftarrow W_1W_2$, the Cramer-Shoup public key is $<g, g_2, U, V, W>$. Note that this public key is drawn from the same distribution as in the standard Cramer-Shoup key generation. Also note that only this public key is necessary for encryption, and not the partial public key values U_1, U_2, etc.

Here we give some intuition for this initialization. First, it is easy to see how the standard Cramer-Shoup private keys are split between alice and bob, with their associated public values. Next, the h_1 and h_2 values will be used as ElGamal [21] public keys for alice and bob, respectively. Note that it is not necessary for bob to receive β_2, since bob does not need to decrypt anything encrypted with h_2. Encryptions using h_2 will simply be used for consistency checking, as described below. Finally, the $D_1, D_2, D_3, D'_1, D'_2, D'_3$ values are used in order to make our consistency proofs work in the concurrent setting based on DDH, as explained later.

5.1 Decryption Protocol

The protocol by which alice and bob cooperate to decrypt ciphertexts with respect to the public key $<g, g_2, U, V, W>$ is shown in Figure 1. As input to this protocol, alice receives a ciphertext c to be decrypted. bob receives no input (but receives $c = <x, y, w, v>$ from alice in the first message). The predicates Ψ, Ψ', Γ, and Γ' used for consistency checking are displayed without their parameters in the figure for readability. We give their full descriptions below, with parameter names that correspond to the parameters in the S-CS protocol.

The decryption protocol proceeds as follows. Upon receiving c to decrypt, alice first generates a share s_1 of a random secret s as used in a Canetti-Goldwasser variant of Cramer-Shoup decryption. Then alice generates ElGamal encryptions

of x^{s_1}, y^{s_1}, v^{s_1}, and $x^{-(a_1+c_1\sigma)}y^{-(b_1+d_1\sigma)}$. All of these values except v^{s_1} are needed by Bob to be able to perform his part of the decryption, but it is necessary to include v^{s_1} for consistency checking, and more specifically, for the protocol's proof of security. She generates these encryptions under the public key h_1, for which she knows the secret key. Finally, alice proves that she has generated these encryptions consistently.

Once bob receives c and the four encryptions from alice and accepts the proof, bob generates his own share s_2 of s. (Note that this is an intuitive description - s itself is actually determined by s_1 and s_2.) Next bob uses the homomorphic properties of the ElGamal encryption scheme used by alice to compute an encryption (still under the public key for which alice knows the secret key) of a partial decryption of c, using the first, second, and fourth encryptions sent by alice. Then bob generates ElGamal encryptions of x^{s_2}, y^{s_2}, v^{s_2}, and $x^{-s_2(a_2+c_2\sigma)}y^{-s_2(b_2+d_2\sigma)}$ under the public key h_2, for which the secret key is not known to alice. Finally, bob proves that he has generated these encryptions consistently. Note that the extra encryptions are not necessary for any computations of alice, but are used for consistency checking, and more specifically, for the protocol's proof of security.

When alice receives the encryptions from bob and accepts the proof, she decrypts the encryption containing bob's partial decryption of c, and then finishes the decryption of c using her local values.

Given g, g_2, $c = <x, y, w, v>$, and $\sigma = H(x, y, w)$, the predicates Ψ, Ψ', Γ, and Γ' are defined as follows.

$$\Psi[U_1, V_1, E_1, E_2, E_3, E_4] \stackrel{\text{def}}{=} \left[\begin{array}{l} \exists r_1, r_2, r_3, r_4, a_1, b_1, c_1, d_1, s_1 : \\ \qquad\qquad U_1 = g^{a_1}(g_2)^{b_1} \\ \wedge \qquad\qquad V_1 = g^{c_1}(g_2)^{d_1} \\ \wedge \qquad\qquad E_1 = (g^{r_1}, (h_1)^{r_1}x^{s_1}) \\ \wedge \qquad\qquad E_2 = (g^{r_2}, (h_1)^{r_2}y^{s_1}) \\ \wedge \qquad\qquad E_3 = (g^{r_3}, (h_1)^{r_3}v^{s_1}) \\ \wedge\, E_4 = (g^{r_4}, (h_1)^{r_4}x^{-(a_1+c_1\sigma)}y^{-(b_1+d_1\sigma)}) \end{array}\right]$$

$$\Psi'[U_2, V_2, W_2, E_5, E'_1, E'_2, E'_3, E'_4]$$
$$\stackrel{\text{def}}{=} \left[\begin{array}{l} \exists r_5, r'_1, r'_2, r'_3, r'_4, a_2, b_2, c_2, d_2, e_2, s_2 : \\ \qquad\qquad U_2 = g^{a_2}(g_2)^{b_2} \\ \wedge \qquad\qquad V_2 = g^{c_2}(g_2)^{d_2} \\ \wedge \qquad\qquad W_2 = g^{e_2} \\ \wedge\, E_5 = (g^{r_5}, (h_1)^{r_5}x^{e_2}(vx^{-(a_2+c_2\sigma)}y^{-(b_2+d_2\sigma)})^{s_2}) \\ \qquad\quad \times (E_1)^{-(a_2+c_2\sigma)} \times (E_2)^{-(b_2+d_2\sigma)} \times (E_4)^{s_2} \\ \wedge \qquad\qquad E'_1 = (g^{r'_1}, (h_2)^{r'_1}x^{s_2}) \\ \wedge \qquad\qquad E'_2 = (g^{r'_2}, (h_2)^{r'_2}y^{s_2}) \\ \wedge \qquad\qquad E'_3 = (g^{r'_3}, (h_2)^{r'_3}v^{s_2}) \\ \wedge \qquad\qquad E'_4 = (g^{r'_4}, (h_2)^{r'_4}) \times (E'_1)^{-(a_2+c_2\sigma)} \\ \qquad\qquad\qquad\qquad \times (E'_2)^{-(b_2+d_2\sigma)} \end{array}\right]$$

$$\Gamma[D_1, D_2, D_3] \stackrel{\text{def}}{=} [\exists r : D_1 = g^r \wedge D_3 = (D_2)^r]$$

$$\Gamma'[D'_1, D'_2, D'_3] \stackrel{\text{def}}{=} [\exists r : D'_1 = g^r \wedge D'_3 = (D'_2)^r]$$

The encryptions of alice are defined to be consistent if Ψ holds, but instead of simply constructing $\Sigma[\Psi]$ to prove consistency, alice constructs $\Sigma[\Psi, \Gamma]$, proving that either Ψ holds, or the triple (D_1, D_2, D_3) is a Diffie-Hellman triple. Obviously, since (D_1, D_2, D_3) was chosen randomly in initialization, most likely it will not be a Diffie-Hellman triple, and thus alice will essentially be proving that Ψ holds. The reason for including Γ is that we will be able to use it to simulate the Σ-protocols for alice, by having our simulator set (D_1, D_2, D_3) to be a Diffie-Hellman triple in the initialization protocol. By the hardness of DDH, this should not noticeably affect the adversary. Note that this technique only works in the case of static adversaries, and in particular, bob-compromising adversaries, since setting (D_1, D_2, D_3) to be a Diffie-Hellman triple may also allow an adversary to give a valid proof $\Sigma[\Psi, \Gamma]$ without Ψ holding. However, it is easy

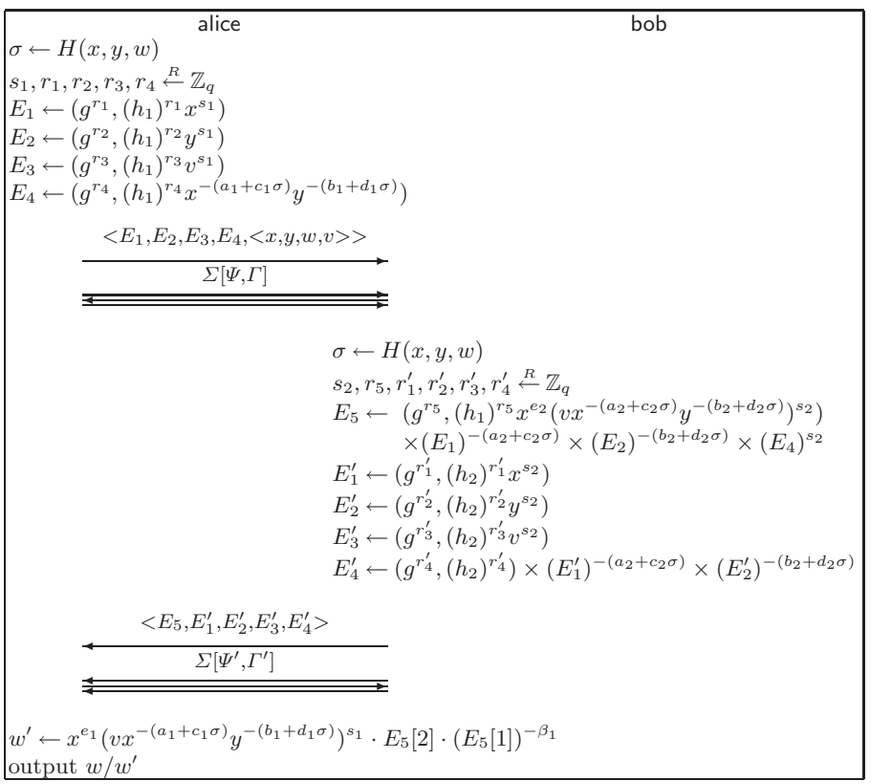

Fig. 1. S-CS shared decryption protocol: alice receives a ciphertext $c = <x, y, w, v>$ as input

to see (and follows from the proof) that a bob-compromising adversary gains no advantage from this.

The encryptions of bob are defined to be consistent if Ψ' holds, and the reasoning behind the $\Sigma[\Psi', \Gamma']$ construction is similar to the reasoning behind the $\Sigma[\Psi, \Gamma]$ construction of alice. $\Sigma[\Psi, \Gamma]$ and $\Sigma[\Psi', \Gamma']$ are similar to other protocols for proving relations among discrete logs, e.g., [9], and are omitted due to space limitations.

At this point, we have stated that E_3 and E'_i for $1 \leq i \leq 4$, as well as the two Σ-protocols, are used for consistency checking, and thus it may be tempting to believe that they could all be removed from the protocol if one were only to consider security against "honest-but-curious" adversaries. However, this does not seem to be true. The Σ-protocols and E'_4 could in fact be removed, but the other values serve another purpose in our security proofs, namely, to allow a simulator for one of the parties to obtain the results of partial decryption computations from the other party. Thus if one were to consider the "simplified" protocol for honest-but-curious adversaries, only E'_4 and the Σ-protocols would be removed, leaving alice and bob to send values to each other that are never actually used.[5]

As a final remark, and relating to the preceding discussion, our simulator does not require knowledge of the other party's share of the decryption randomizer s, but only the results of partial decryption computations. These can be encrypted and checked for consistency easily, using techniques that rely solely on the hardness of DDH. This is one of the important technical contributions of this paper, since having the simulator obtain s itself, although not difficult to achieve in threshold Cramer-Shoup protocols [10, 33] assuming an honest majority, seems to require a much more complicated two-party protocol, and in fact may not admit a protocol whose security relies solely on the hardness of DDH. For instance, it may requires techniques such as those in [36].

Efficiency As shown, our protocol requires 6 messages. This could possibly be improved to 4 messages by using the "committed proof" technique of Jarecki and Lysyanskaya [33] or Damgård [18]. In particular, one could replace bob's proof $\Sigma[\Psi', \Gamma']$ by a committed proof of $\Sigma[\Psi', \Gamma']$, where, in particular, E_5 is kept secret until the third message of the committed proof. This would allow the proofs by alice and bob to be interleaved, since E_5 would not be revealed until after bob verifies that Ψ holds. This would be a novel application of the committed proof technique, i.e., it would be used not for the purpose of obtaining security against adaptive adversaries, but for improving efficiency. However, the security reduction involved in the committed proof technique is neither as straightforward nor as efficient as the security reduction in our proofs. Nevertheless, we will provide the analysis of this variation in the full paper.

We should also note that our protocol could be reduced to two messages using the standard Fiat-Shamir technique [27] for making proofs non-interactive

[5] This would indeed be a "curious" protocol.

using a hash function to calculate a challenge, but then a proof of security would require the random oracle assumption, which we specifically want to avoid.

Turning to computational complexity, one can see that each party must perform roughly 90 exponentiations.[6] By comparison, the protocol of Shoup and Gennaro [42] only requires each party to perform about 7 exponentiations. However, the security of their protocol relies on the random oracle assumption, which, as stated above, we specifically want to avoid.

6 Security for S-CS

Due to space limitations, we will present our theorems, and briefly sketch our proofs. Details will be presented in the full paper.

First we informally define some notation. $\mathsf{Adv}^{\mathrm{DDH}}_{G_q}(t)$ is the maximum advantage of distinguishing a DH triple (g^x, g^y, g^{xy}) from a random triple (g^x, g^y, g^z), where the maximum is taken over all adversaries that run in time t. See [3] for details. For an encryption scheme $\mathcal{E} = (G_\mathcal{E}, E, D)$, $\mathsf{Adv}^{\text{ind-cca2}}_{\mathcal{E},G_q}(t,u)$ is the maximum advantage of distinguishing which of two messages was encrypted by a test oracle, where the maximum is taken over all adversaries that run in time t and make u queries to a decryption oracle (but not on the ciphertext returned by the test oracle). See [6, Property IND-CCA2] for details.

Finally, for an adversary $\mathcal{A}$ in the model described in Section 3 where two parties, alice and bob are running the S-CS protocol, $\mathsf{Adv}^{\text{ind-cca2}}_{\text{S-CS},G_q}(\mathcal{A})$ is the advantage of $\mathcal{A}$ in distinguishing which of two messages was encrypted by a test oracle, given that $\mathcal{A}$ cannot start an alice or bob session using the ciphertext returned by the test oracle. An "alice-compromising attacker" is additionally given perpetual read access to the private storage and computation of alice. A "bob-compromising attacker" has perpetual read access to the private storage and computation of bob. Recall that $\mathcal{A}$ is either an alice-compromising attacker or a bob-compromising attacker, but not both, and we assume this is statically fixed before initialization.

Now we state our theorems and give sketches of our proofs. The idea behind each proof is to construct a series of systems S-CS$_0$, S-CS$_1$, ..., related to S-CS, with S-CS$_0$ = S-CS, and such that we eventually come to a system S-CS$_i$ such that breaking S-CS$_i$ implies breaking the original Cramer-Shoup cryptosystem. We then show that for any attacker, the difference in the advantage of the attacker in breaking S-CS$_{i-1}$ and S-CS$_i$ is related to the maximum advantage of breaking DDH. For the following theorems, let t_{exp} be the time to perform an exponentiation in G_q.

[6] Although this number is somewhat high, most of the exponentiations are performed over one of a small number of bases, and thus preprocessing can be used to greatly reduce the computation time. Also, assuming that the group is of size q where $|q| = 160$, the exponents are reasonably small (roughly 160 bits each).

Theorem 1. *Fix an* alice*-compromising adversary $\mathcal{A}$ that runs in time t. Then for $t' = O(t + (q_{\mathsf{alice}} + q_{\mathsf{bob}})t_{exp})$:*

$$\mathsf{Adv}^{\text{ind-cca2}}_{\text{S-CS},G_q}(\mathcal{A}) \leq 4 \cdot \mathsf{Adv}^{\text{DDH}}_{G_q}(t') + \mathsf{Adv}^{\text{ind-cca2}}_{\text{CS},G_q}(t', q_{\mathsf{bob}}) + \frac{2(q_{\mathsf{bob}} + 2)}{q}.$$

This is proven as follows. First simulate the initialization so that (D'_1, D'_2, D'_3) is a DH triple. The advantage gained by $\mathcal{A}$ is at most $O(\mathsf{Adv}^{\text{DDH}}_{G_q}(t'))$. Then have bob instances generate E'_1, E'_2, E'_3, E'_4 randomly, and prove $\Sigma[\Psi', \Gamma']$ using Γ' (since (D'_1, D'_2, D'_3) is a DH triple). One can show that the advantage gained by $\mathcal{A}$ is $O(\mathsf{Adv}^{\text{DDH}}_{G_q}(t'))$ plus a small amount related to Σ-protocols. Next set up a Cramer-Shoup decryption oracle (using the key from the simulated initialization) and have bob compute the encryption E_5 by taking the result of calling the decryption oracle on the input ciphertext and modifying it appropriately using E_1, E_2, E_3 (which can be decrypted with β_1 obtained from the simulated initialization). As long as E_1, E_2, E_3, E_4 are consistent, this does affect $\mathcal{A}$. Breaking this scheme can now be reduced to breaking the Cramer-Shoup scheme by taking a Cramer-Shoup key, along with a decryption and test oracle, simulating the private key shares of alice along with all public key shares, and having bob use the given decryption oracle to decrypt the input ciphertexts.

Theorem 2. *Fix a* bob*-compromising adversary $\mathcal{A}$ that runs in time t. Then for $t' = O(t + (q_{\mathsf{alice}} + q_{\mathsf{bob}})t_{exp})$:*

$$\mathsf{Adv}^{\text{ind-cca2}}_{\text{S-CS},G_q}(\mathcal{A}) \leq 4 \cdot \mathsf{Adv}^{\text{DDH}}_{G_q}(t') + \mathsf{Adv}^{\text{ind-cca2}}_{\text{CS},G_q}(t', q_{\mathsf{bob}}) + \frac{6(q_{\mathsf{alice}} + 1) + 2}{q}.$$

This is proven as follows. First simulate the initialization so that (D_1, D_2, D_3) is a DH triple. The advantage gained by $\mathcal{A}$ is at most $O(\mathsf{Adv}^{\text{DDH}}_{G_q}(t'))$. Then have alice instances generate E_1, E_2, E_3, E_4 randomly, prove $\Sigma[\Psi, \Gamma]$ using Γ (since (D_1, D_2, D_3) is a DH triple), and compute the decryption of the input ciphertext using E'_1, E'_2, E'_3 (which can be decrypted with β_2 obtained from the simulated initialization). One can show that the advantage gained by $\mathcal{A}$ is $O(\mathsf{Adv}^{\text{DDH}}_{G_q}(t'))$ plus a small amount related to Σ-protocols. Next set up a Cramer-Shoup decryption oracle (using the key from the simulated initialization) and have alice output the result of calling the decryption oracle. Since E_1, E_2, E_3, E_4 are random, if $E_5, E'_1, E'_2, E'_3, E'_4$ are consistent, this does affect $\mathcal{A}$. Breaking this scheme can now be reduced to breaking the Cramer-Shoup scheme by taking a Cramer-Shoup key, along with a decryption and test oracle, simulating the private key shares of bob along with all public key shares, and having alice use the decryption oracle to decrypt the input ciphertexts.

References

[1] M. Abe. Robust distributed multiplication without interaction. In *CRYPTO '99* (LNCS 1666), pages 130–147, 1999. 48

[2] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In 3rd *ACM Conference on Computer and Communications Security*, pages 6–17, 1996. 47

[3] D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium* (LNCS 1423), pp. 48–63, 1998. 57

[4] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *CRYPTO '97* (LNCS 1294), pages 425–439, 1997. 49

[5] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97* (LNCS 1233), pages 480–494, 1997. 48

[6] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO '98* (LNCS 1462), pp. 26–45, 1998. 57

[7] C. Boyd. Digital multisignatures. In H. J. Beker and F. C. Piper, editors, *Cryptography and Coding*, pages 241–246. Clarendon Press, 1986. 48

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In 1st *ACM Conference on Computer and Communications Security*, pages 62–73, November 1993. 49

[9] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Department of Computer Science, ETH Zurich, March 1997. 56

[10] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT '99* (LNCS 1592), pages 90–106, 1999. 48, 50, 56

[11] R. Canetti, O. Goldreich, and S. Halevi. Random oracle methodology, revisited. In 30th ACM Symposium on Theory of Computing, pages 209–218, 1998. 49

[12] M. Cerecedo, T. Matsumoto, H. Imai. Efficient and secure multiparty generation of digital signatures based on discrete logarithms. *IEICE Trans. Fundamentals of Electronics Communications and Computer Sciences*, E76A(4):532–545, April 1993. 48, 49

[13] R. Cramer. Modular Design of Secure yet Practical Cryptographic Protocols. Ph.D. Thesis. CWI and University of Amsterdam, 1997. 51

[14] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94* (LNCS 839), pages 174–187, 1994. 51, 52

[15] R. Cramer and V. Shoup. A practical public-key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98* (LNCS 1462), pages 13–25, 1998. 47, 50

[16] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT 2002* (LNCS 2332), pages 45–64, 2002. 50

[17] R. A. Croft and S. P. Harris. Public-key cryptography and reusable shared secrets. In H. Baker and F. Piper, editors, *Cryptography and Coding*, pages 189–201, 1989. 48

[18] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000* (LNCS 1807), pages 418–430, 2000. 56
[19] Y. Desmedt. Society and group oriented cryptography: a new concept. In *CRYPTO '87* (LNCS 293), pages 120–127, 1987. 48
[20] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO '89* (LNCS 435), pages 307–315, 1989. 48
[21] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. 53
[22] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM Symposium on Theory of Computing*, pp. 416–426, 1990. 52
[23] P. Fouque and D. Pointcheval. Threshold Cryptosystems secure against Chosen-Ciphertext Attack. In *ASIACRYPT '01* (LNCS 2248), pages 351–368, 2001. 48
[24] Y. Frankel. A practical protocol for large group oriented networks. In *EUROCRYPT '89* (LNCS 434), pages 56–61, 1989. 48
[25] Y. Frankel, P. MacKenzie, and M. Yung. Robust efficient distributed RSA-key generation. In 30$^{\text{th}}$ *ACM Symposium on Theory of Computing*, pages 663–672, 1998. 49
[26] Y. Frankel, P. MacKenzie, and M. Yung. Adaptively-secure distributed threshold public key systems. In *European Symposium on Algorithms* (LNCS 1643), pages 4–27, 1999. 48
[27] A. Fiat and A.Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO '86* (LNCS 263), pages 186–194, 1987. 56
[28] N. Gilboa. Two party RSA key generation. In *CRYPTO '99* (LNCS 1666), pages 116–129, 1999. 49
[29] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *EUROCRYPT '96* (LNCS 1070), pages 354–371, 1996. 48, 49
[30] L. Harn. Group oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proc.-Comput. Digit. Tech.* 141(5):307–313, 1994. 48
[31] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public-key and signature schemes. In 4$^{\text{th}}$ *ACM Conference on Computer and Communications Security*, pages 100–110, 1997. 48
[32] T. Hwang. Cryptosystem for group oriented cryptography. In *EUROCRYPT '90* (LNCS 473), pages 352–360, 1990. 48
[33] S. Jarecki and A. Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In *EUROCRYPT 2000* (LNCS 1807), pages 221–242, 2000. 48, 56
[34] S. Langford. Threshold DSS signatures without a trusted party. In *CRYPTO '95* (LNCS 963), pages 397–409, 1995. 48, 49
[35] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. DIMACS Technical Report 2001-19, May 2001. Extended abstract in *2001 IEEE Symposium on Security and Privacy*, May 2001. 47, 48, 51, 52
[36] P. MacKenzie and M. K. Reiter. Two-party generation of DSA signatures. In *CRYPTO 2001* (LNCS 2139), pages 137–154, 2001. 48, 49, 56
[37] S. Micali. Fair public-key cryptosystems. In *CRYPTO '92* (LNCS 740), pages 113–138, 1992. 47
[38] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT '99* (LNCS 1592), pages 223–238, 1999. 48
[39] C. Park and K. Kurosawa. New ElGamal type threshold digital signature scheme. *IEICE Trans. Fundamentals of Electronics Communications and Computer Sciences*, E79A(1):86–93, January, 1996. 48

[40] T. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT '91* (LNCS 547), pages 522–526, 1991. 48

[41] G. Poupard and J. Stern. Generation of shared RSA keys by two parties. In *ASIACRYPT '98*, LNCS 1514, pages 11–24, 1998. 49

[42] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *EUROCRYPT '98*, pp. 1–16, 1998. 48, 51, 57

[43] A. Yao. Protocols for secure computation. In 23rd *IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982. 47