Lecture Notes for CS-578 Cryptography (SS2006) Prof. Michael Backes

8. The Cramer-Shoup Public-key Encryption System

Lecture 11 Saarland University

This chapter is dedicated to the Cramer-Shoup public-key encryption scheme, which constitutes the first (and essentially still only) efficient public-key encryption scheme that is provably secure under chosen-ciphertext attack under standard cryptographic assumptions.

8.1 Keyed Hash Functions

In order to present the Cramer-Shoup encryption scheme, we first have to briefly introduce keyed hash functions. In contrast to their unkeyed counterparts, keyed hash functions have a rigorous security definition that can in fact be fulfilled by candidate functions. Let Gen denote a probabilistic algorithm called the key generation algorithm. Then a family $H = (\mathsf{hash}(pk, \cdot))_{pk \in [\mathsf{Gen}(n)]}$ (here $[\mathsf{Gen}(n)]$ denotes the carrier set of the probabilistic algorithm $\mathsf{Gen}(n)$, i.e., the set of all values that can be output by $\mathsf{Gen}(n)$ with non-zero probability) is called a keyed collision-resistant family of hash functions iff for all efficient algorithms A, the probability

$$\mathsf{Pr}(m \neq m^* \wedge \mathsf{hash}(pk, m) = \mathsf{hash}(pk, m^*); pk \leftarrow \mathsf{Gen}(n), (m, m^*) \leftarrow \mathsf{A}(n, pk))$$

is negligible in n.

8.2 Definition of the Cramer-Shoup Public-key Encryption Scheme

The Cramer-Shoup encryption scheme is quite similar to ElGamal encryption in subgroups G_q of $\mathbb{Z}_p^*$ of prime order, but the ciphertext contains additional fields. Intuitively, the purpose of these fields is that only someone who knows the plaintext can construct them correctly. However, there is no notion of "proof of knowledge" in the security proof, nor an intuitive explanation yet why exactly this choice of fields is good; they were simply chosen so that the proof worked. The key is also extended corresponding to the additional fields in the ciphertext.

8.2.1 Key Generation

Choose a subgroup G_q of prime order q of $\mathbb{Z}_p^*$ as for ElGamal encryption in subgroups of $\mathbb{Z}_p^*$, i.e., randomly choose an n-bit prime q and an $n_p(n)$-bit prime p such that $q | p-1$ (recall that $n_p(\cdot)$ was the polynomial function that relates the lengths of p and q). Choose a random generator g_1 of G_q. (Alternatively, these parameters can be fixed in advance and publicly known, cf. the description of Elgamal.) Choose a second generator g_2 of G_q. Choose five elements x_1, x_2, y_1, y_2, z randomly from $\mathbb{Z}_q$. Here z will play the role of the secret ElGamal key. Then compute

$$s := g_1^{x_1} \cdot g_2^{x_2}, \quad t := g_1^{y_1} \cdot g_2^{y_2}, \quad h := g_1^z.$$

Here h will play the role of the public key of ElGamal, and thus g_1 plays the role of g in ElGamal. Finally generate a key pk_{hash} for a keyed collision-resistant family of hash functions. The resulting actual hash function is abbreviated as

$$\mathsf{H}(\cdot) := \mathsf{hash}(pk_{\mathsf{hash}}, \cdot).$$

The complete public key is then

$$pk := (q, p, g_1, g_2, s, t, h, pk_{\mathsf{hash}})$$

and the secret key

$$sk := (pk, x_1, x_2, y_1, y_2, z).$$

8.2.2 Encryption

Choose r randomly from $\mathbb{Z}_q$. This corresponds to y in ElGamal encryption. Still as in ElGamal encryption, the ciphertext c contains components

$$i_1 := g_1^r, \quad c^* := h^r \cdot m.$$

In addition, compute

$$i_2 := g_2^r, \quad \alpha := \mathsf{H}(i_1, i_2, c^*), \quad v := s^r \cdot t^{r\alpha}.$$

The ciphertext is

$$c := (i_1, i_2, c^*, v).$$

Let us start with some intuition on this construction: In some way, you would expect that an active attacker who tries to reuse or blind i_1 as with ElGamal must reuse or blind i_2 in the same way so that they remain consistent; then α will be more or less fixed. (He may be able to choose c^* freely, but it seems he cannot do much better than choose some values c^* and get a list of values α to choose from.) None of these values α will be that from the original ciphertext. Hence it seems he cannot reuse or blind the given v, but must compute it by knowing r. Then, however, he would also know the ciphertext. However, all this was not a proof of security (nor, e.g., does it really explain the need for i_2).

8.2.3 Decryption

Given pk and $c = (i_1, i_2, c^*, v)$, the recipient recomputes $\alpha := \mathsf{H}(i_1, i_2, c^*)$. Now he must verify v. However, even with the secret key, he cannot retrieve r. Instead, he tests whether

$$i_1^{x_1 + y_1\alpha} \cdot i_2^{x_2 + y_2\alpha} = v.$$

If this is true, he decrypts m as with ElGamal, i.e., by computing

$$k := i_1^z, \quad m := c^*/k.$$

8.2.4 Correctness of Decryption

If a value m is output at all, its correctness follows exactly as with ElGamal: $k = i_1^z = g_1^{rz} = h^r$ and therefore c^*/k is the original m.

It remains to be shown that any correctly constructed v passes the test. Here we have

$$v = s^r \cdot t^{r\alpha} = (g_1^{x_1} \cdot g_2^{x_2})^r \cdot (g_1^{y_1} \cdot g_2^{y_2})^{r\alpha} = g_1^{rx_1 + ry_1\alpha} \cdot g_2^{rx_2 + ry_2\alpha} = i_1^{x_1 + y_1\alpha} \cdot i_2^{x_2 + y_2\alpha}. \tag{8.1}$$

The central question we have to ask ourselves here is the following: Does this test reject all incorrect quadruples? In other words, for any quadruple (i_1, i_2, c^*, v) that passes the test, is there an r and m such that $i_1 = g_1^r$, $i_2 = g_2^r$, $c^* = h^r \cdot m$, and $v = s^r \cdot t^{r\alpha}$ for $\alpha := \mathsf{H}(i_1, i_2, c^*)$? Equivalently, is there an r such that $i_1 = g_1^r$, $i_2 = g_2^r$, and $v = s^r \cdot t^{r\alpha}$ (then m is defined uniquely as c^*/h^r)?

The only possible value for r (modulo q) is already fixed by i_1. (This is so since $g_1^r = g_1^{r'}$ if and only if $r = r' \bmod q$.) We now distinguish two cases:

- If $i_2 = g_2^r$, then it is clear that v must also be of the required form (to see this, just write "$= v$" at the end instead of the beginning of Equation 8.1 and read it backwards).

- For any other i_2 (i.e., $i_2 \neq g_2^r$), a suitable value

  $$v := i_1^{x_1 + y_1\alpha} \cdot i_2^{x_2 + y_2\alpha}$$

  also exists. However, we will see in a lemma below that an attacker who does not know the secret key will not be able to find such a v except with negligible probability.

8.3 Security Proof

Theorem 8.1 *The Cramer-Shoup encryption scheme is secure under chosen-ciphertext attack (CCA2) under the Decisional Diffie-Hellman assumption.* □

Proof. The proof constitutes a complex reduction showing that if a successful attacker $\mathsf{A_{CS}}$ against the Cramer-Shoup system exist, i.e., $\mathsf{A_{CS}}$ succeeds in an game against the CCA2 challenger, we can construct a successful attacker $\mathsf{A_{DDH}}$ against the Decisional Diffie-Hellman assumption. For readability, we assume that $\mathsf{A_{CS}}$ consists of four algorithms $\mathsf{A}_1, \ldots, \mathsf{A}_4$ for the respective phases of the CCA2 experiment. (A_1 outputs ciphertexts and expects the corresponding plaintexts, A_2 outputs two challenge messages m_0^*, m_1^* and expects the ciphertext c of one of the messages, A_3 outputs ciphertexts different from c and expects the corresponding plaintexts, while A_4 finally outputs the guess b^*. The structure of $\mathsf{A_{CS}}$, of $\mathsf{A_{DDH}}$, and of the overall reduction is depicted in Figure 8.1. More precisely, the figure shows the desired input/output behavior of $\mathsf{A_{CS}}$ and the given input/output behavior of $\mathsf{A_{CS}}$. What remains to be done is to define the actual adversary $\mathsf{A_{DDH}}$ and prove that it wins against the DDH challenger with not-negligible probability, provided that $\mathsf{A_{DDH}}$ would win its game against the CCA2 challenger with not-negligible probability. The actual proof has to fill in how $\mathsf{A_{DDH}}$ treats incoming messages and computes outgoing messages.

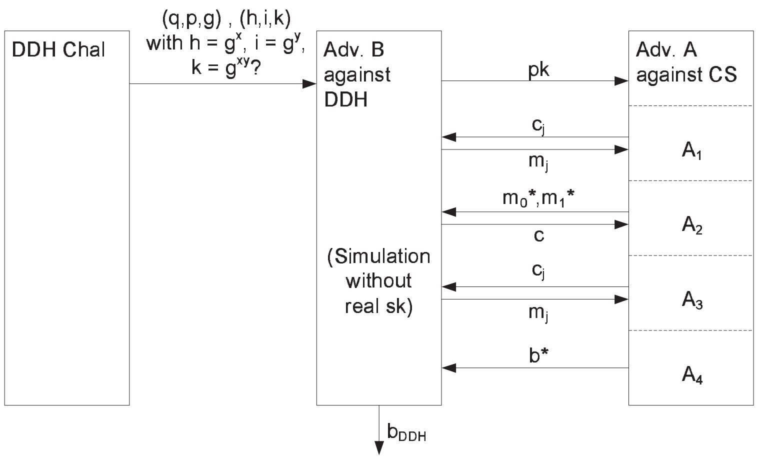

Figure 8.1: Structure of the Security Proof of the Cramer-Shoup Encryption Scheme

8.3.1 Intuition on the Reduction

Basic idea: The basic idea of the actual proof is the following: The given Diffie-Hellman tuple (h, i, k) (which is either (g^x, g^y, g^{xy}) or (g^x, g^y, g^a) for some random a) is used in the public key and ciphertext in a way so that everything is essentially correct if $k = g^{xy}$ in fact holds. In the following we use the notation $h = g^x$ and $i = g^y$ because such values x and y have to exist. (Note, however, that $\mathsf{A}_{\mathsf{DDH}}$ does not know them.)

- Hence, if $k = g^{xy}$, we give a correct simulation to A_{CS} and thus the success probability of A_{CS} is significantly better than $\frac{1}{2}$. This however also means that the probability that $b^* = b$ is significantly better than $\frac{1}{2}$.
- In the other case, where k is random, it will be shown that everything A_{CS} sees is essentially independent of b. Hence $b^* = b$ will hold with probability almost exactly $\frac{1}{2}$.

Thus the final step of $\mathsf{A}_{\mathsf{DDH}}$ is to set $b_{\mathsf{DDH}} = 0$ if and only if $b^* = b$. Thus $b_{\mathsf{DDH}} = 0$ can be interpreted as a guess that $k = g^{xy}$ is given, and in fact $b_{\mathsf{DDH}} = 0$ will be output with significantly larger probability in this case than in the other case.

Where to use the Diffie-Hellman Triple: The main decision is now where $\mathsf{A}_{\mathsf{DDH}}$ can reasonably use the given input, i.e., in what places A_{CS} expects a Diffie-Hellman triple among its inputs. The choice made here is to use

$$g_1 := g \text{ and } (g_2, i_1, i_2) := (h, i, k),$$

where g_1, g_2 are the two generators in the public key, and i_1 and i_2 the first two components of the ciphertext c that has to be produced in the challenge phase (against adversary A_2).

Using g and h as g_1 and g_2 is no problem: g and h are two random generators of G_q just as g_1 and g_2 should be.[1]

[1]Recall that $\mathsf{A}_{\mathsf{DDH}}$ has to work in the context of the Decisional Diffie-Hellman Assumption, i.e., its parameters are chosen according to the definition of the DDH assumption. Thus g is a random generator by construction, and $h = g^x$ for a random exponent x and thus also a random generator.

If $k = g^{xy}$ (i.e., the case that we would like to give a good simulation for), then i_1, i_2 are also chosen with the correct probability distribution because then $i_1 = i = g^y = g_1^y$ and $i_2 = k = g^{xy} = h^y = g_2^y$. In other words, y plays the role of r in a correct ciphertext. However, $\mathsf{A}_{\mathsf{DDH}}$ does not know r and therefore has to compute the remaining components of the ciphertext c in another way than usual.

We now show the entire construction of $\mathsf{A}_{\mathsf{DDH}}$. For the moment, we could think that $\mathsf{A}_{\mathsf{DDH}}$ could choose sk in a perfectly correct way starting from (p, q, g_1, g_2), because none of the remaining components was used for the outside inputs i and k. Nevertheless, for reasons that will be seen in the case "k random", $\mathsf{A}_{\mathsf{DDH}}$ chooses sk differently so that it has a bit more flexibility.

8.3.2 Construction of $\mathsf{A}_{\mathsf{DDH}}$

In addition to the construction, we immediately show that everything is correct in the case $k = g^{xy}$.

1. Actions before A_{CS} is called:

   - The group and the first generator in pk are those given from the outside, i.e., p and q are the same for A_{CS} as for $\mathsf{A}_{\mathsf{DDH}}$, and $g_1 := g$.
   - The second generator, g_2, is h from the input.
   - Now, instead of five, six elements x_1, x_2, y_1, y_2, z_1, z_2 are chosen randomly from $\mathbb{Z}_q$. The components s and t of the public key are computed in the correct way, but the last component is now
   $$h_{CS} := g_1^{z_1} \cdot g_2^{z_2}.$$
   - The hash key pk_{hash} is generated as usual.

   The change in the choice of h_{CS} does not alter the probability distribution of pk: Both the correct g_1^z and the simulated $g_1^{z_1} \cdot g_2^{z_2}$ are uniformly random elements of G_q. In fact, we can define the appropriate z as $z := z_1 + x \cdot z_2$ because $g_2 = g_1^x$. However, $\mathsf{A}_{\mathsf{DDH}}$ does not know x and thus z.

2. Interaction with A_1: Now $\mathsf{A}_{\mathsf{DDH}}$ gives pk to A_{CS}, so that the first active attack phase, A_1, starts. We have to show how $\mathsf{A}_{\mathsf{DDH}}$ decrypts the ciphertexts $c_j = (i_{j,1}, i_{j,2}, c_j^*, v_j)$ that A_1 sends. As $\mathsf{A}_{\mathsf{DDH}}$ knows the entire secret key, this is quite easy; we only have to adapt the decryption to the changed value h_{CS}: The verification of v_j is exactly as before (it uses neither z nor h). Now, in a normal decryption, k_j would be computed as $i_{j,1}^z$. If we define r_j such that $i_{j,1} = g_1^{r_j}$ (although $\mathsf{A}_{\mathsf{DDH}}$ does not know this r_j), we can rewrite this as
   $$k_j = i_{j,1}^z = g_1^{r_j z} = (g_1^{z_1} \cdot g_2^{z_2})^{r_j} = g_1^{r_j z_1} \cdot g_2^{r_j z_2}.$$
   If the ciphertext is correct, this is $i_{j,1}^{z_1} \cdot i_{j,2}^{z_2}$. Thus $\mathsf{A}_{\mathsf{DDH}}$ sets
   $$k_j := i_{j,1}^{z_1} \cdot i_{j,2}^{z_2}$$
   and then decrypts $m_j := c_j^*/k_j$ in the normal way.

   However, if the ciphertext is not correct (and it was sent by the attacker algorithm A_1, so nobody guarantees that it is correct), then $i_{j,2}$ need not be $g_2^{r_j}$, and thus $\mathsf{A}_{\mathsf{DDH}}$ gives A_1 another answer than an actual CCA2 challenger would. This might alter the success probability of A_1 and thus destroy our proof. Fortunately, we can show in Lemma 8.1 below that this case only occurs with negligible probability.

3. Interaction with A_2: When A_1 stops and A_2 sends m_0^* and m_1^*, then $\mathsf{A}_{\mathsf{DDH}}$ first chooses $b \in \{0,1\}$ as usual. Then it puts the given i and k into the ciphertext as described above, i.e., it uses them as i_1 and i_2. Recall that this means that r is the unknown y if $k = g^{xy}$.

   Now $\mathsf{A}_{\mathsf{DDH}}$ has to compute the remaining components c^* and v of this ciphertext, so that A_{CS} gets an input with the correct distribution in the case $k = g^{xy}$. Here c^* should be $h_{CS}^y \cdot m_b^*$. As $\mathsf{A}_{\mathsf{DDH}}$ does not know r, but it does know the secret key (in contrast to a normal CCA2 challenger who encrypts), it encrypts more or less as it would usually decrypt, i.e., it computes h_{CS}^y as $(g_1^{z_1} \cdot g_2^{z_2})^y = g_1^{yz_1} \cdot g_2^{yz_2} = i^{z_1} \cdot k^{z_2}$. This means it actually sets

   $$c^* := i^{z_1} \cdot k^{z_2} \cdot m_b^*.$$

   Finally, the correct v certainly fulfills the verification. Hence the only possible choice is $i^{x_1+y_1\alpha} \cdot k^{x_2+y_2\alpha}$, where $\alpha = \mathsf{H}(i,k,c^*)$. This computation is easy for $\mathsf{A}_{\mathsf{DDH}}$ because it knows the secret key.

4. Interaction with A_3: Finally, $\mathsf{A}_{\mathsf{DDH}}$ has to interact with A_3, i.e., to perform decryptions again. This is exactly as in the interaction with A_1, except that $\mathsf{A}_{\mathsf{DDH}}$ compares each ciphertext c_j with the computed challenge ciphertext c first, so that it does not decrypt the secret.

5. Final computation: After A_3 stopped, A_4 starts and outputs a bit b^*. We already described above that $\mathsf{A}_{\mathsf{DDH}}$ outputs $b_{\mathsf{DDH}} = 0$ iff $b^* = b$, i.e., if A_4 guesses correctly.

This finishes the construction of $\mathsf{A}_{\mathsf{DDH}}$. We summarize it in Figure 8.2.

8.3.3 Proof of Correct Simulation

We now have to show how $\mathsf{A}_{\mathsf{DDH}}$ behaves in the two cases.

Behavior if $k = g^{xy}$. For this case, we have, with one exception, already shown during the construction that $\mathsf{A}_{\mathsf{DDH}}$ gives A_{CS} all its inputs with the correct respective probabilities. Once we have filled in the exception by Lemma 8.1, this means that A_{CS} will output a correct guess $b^* = b$ with probability significantly larger than $\frac{1}{2}$, and thus $\mathsf{A}_{\mathsf{DDH}}$ will output $b_{\mathsf{DDH}} = 0$ with that same probability.

Lemma 8.1 *If $k = g^{xy}$, the probability that A_{CS} succeeds in sending any ciphertext c_j whose first two components are not of the form $i_{j,1} = g_1^{r_j}$, $i_{j,2} = g_2^{r_j}$ for some r_j, but which nevertheless passes the verification, is exponentially small. This holds both with interacting with $\mathsf{A}_{\mathsf{DDH}}$ and in a normal game against a CCA2 challenger.* □

Proof. Actually, we can even show this property information-theoretically, i.e., even if A_{CS} were computationally unrestricted.[2] We show that for *every* wrong ciphertext $(i_{j,1}, i_{j,2}, c_j^*, v_j)$, the probability that it passes the verification is small. This probability is over the possible secret keys, more precisely the conditional probability over those secret keys that are still possible given everything that the attacker has seen. (In other words, not even exhaustive search helps because there is no

[2]This may sound like an unnecessary complication, but in fact information-theoretic proofs are usually simpler than computational ones.

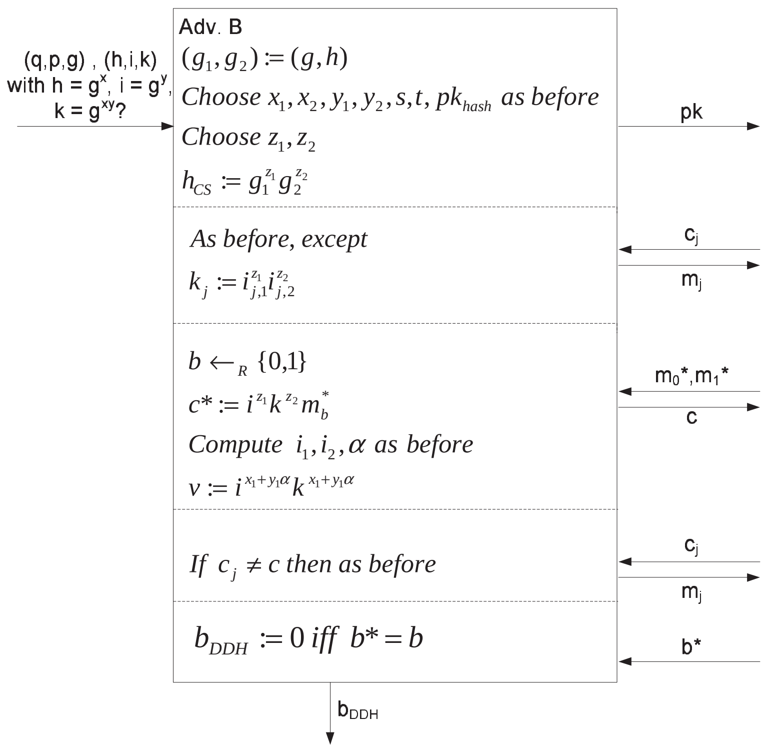
Adv. B
(q,p,g), (h,i,k) with h = g^x, i = g^y, k = g^{xy}?
$(g_1, g_2) := (g, h)$
Choose $x_1, x_2, y_1, y_2, s, t, pk_{hash}$ as before
Choose z_1, z_2
$h_{CS} := g_1^{z_1} g_2^{z_2}$
pk
As before, except
$k_j := i_{j,1}^{z_1} i_{j,2}^{z_2}$
c_j
m_j
$b \leftarrow_R \{0,1\}$
$c^* := i^{z_1} k^{z_2} m_b^*$
Compute i_1, i_2, α as before
$v := i^{x_1+y_1\alpha} k^{x_1+y_1\alpha}$
m_0^*, m_1^*
c
If $c_j \neq c$ then as before
c_j
m_j
$b_{DDH} := 0$ iff $b^* = b$
b^*
b_{DDH}

Figure 8.2: Summary of the Construction of $\mathsf{A}_{\mathsf{DDH}}$

good solution. The point here is that the verification equation is secret; otherwise this could not possibly work.)

Thus let us fix an arbitrary ciphertext $(i_{j,1}, i_{j,2}, c_j^*, v_j)$ where $i_{j,1} = g_1^{r_j}$ and $i_{j,2} = g_2^{r_j^*}$ with $r_j^* \neq r_j \bmod q$. Let $\alpha_j = \mathsf{H}(i_{j,1}, i_{j,2}, c_j^*)$ as usual. The ciphertext passes the verification if and only if

$$v_j = i_{j,1}^{x_1+y_1\alpha_j} \cdot i_{j,2}^{x_2+y_2\alpha_j} = g_1^{r_j(x_1+y_1\alpha_j)} \cdot g_2^{r_j^*(x_2+y_2\alpha_j)}.$$

Recall that $g_2 = h = g_1^x$. It will be helpful to compute entirely in the exponents (exploiting that $g_1^r = g_1^{r'}$ if and only if $r = r' \bmod q$). Hence we define β_j such that $v_j = g_1^{\beta_j}$. Then the verification is true if and only if (modulo q)

$$\beta_j = r_j(x_1 + y_1\alpha_j) + xr_j^*(x_2 + y_2\alpha_j). \tag{8.2}$$

If the attacker knew nothing at all about the secret key, it would be clear that he also had only an exponentially small chance to guess β_j, and thus v_j, correctly. However, the attacker gains knowledge about the secret key from several sources, and we have to show that this knowledge still leaves β_j unclear.

The secrets in Equation 8.2 are x_1, x_2, y_1, and y_2. (An unrestricted attacker could compute x from g_2.) The attacker gets the following information about them (follow Figure 8.2) to see that nothing is forgotten):

- The components $s = g_1^{x_1} \cdot g_2^{x_2}$ and $t = g_1^{y_1} \cdot g_2^{y_2}$ of pk. With unlimited computing power, he could compute their discrete logarithms and thus get two linear equations

$$\sigma = x_1 + x \cdot x_2, \quad \tau = y_1 + x \cdot y_2. \tag{8.3}$$

- Decryption of correct ciphertexts. However, here the attacker only learns $k_j = i_{j,1}^{z_1} \cdot i_{j,2}^{z_2}$, which tells him nothing about (x_1, x_2, y_1, y_2).
- The encryption. Here c^* only depends on the values z, but $v = i^{x_1 + y_1\alpha} \cdot k^{x_2 + y_2\alpha}$. However, this is simply $s^y \cdot t^{y\alpha}$ in this correct case, where s and t are already known, and thus no new information about (x_1, x_2, y_1, y_2).
- Answers on incorrect ciphertexts. This is the case that we are just treating, i.e., we are showing that almost certainly, all answers will be the fixed error message (and that getting the fixed error message gives almost no information).

In summary, except for the error messages, the attacker obtains at most the two linear equations (Equation 8.3) in the four variables. Thus q^2 quadruples (x_1, x_2, y_1, y_2) are still possible from the attacker's point of view. We now show that Equation 8.2 for β_j is linearly independent of the two equations in Equation 8.3: The matrix (in the order x_1, x_2, y_1, y_2) is

$$\begin{pmatrix} 1 & x & 0 & 0 \\ 0 & 0 & 1 & x \\ r_j & xr_j^* & r_j\alpha_j & xr_j^*\alpha_j \end{pmatrix}$$

This can be transformed into

$$\begin{pmatrix} 1 & x & 0 & 0 \\ 0 & 0 & 1 & x \\ 0 & x(r_j^* - r_j) & 0 & x\alpha_j(r_j^* - r_j) \end{pmatrix}$$

We now exclude the case $x \neq 0$, which only occurs with exponentially small probability. We also know $r_j^* \neq r_j$. Thus $x(r_j^* - r_j) \neq 0$, and we can divide it out (recall that this is modulo q and thus in a field where every non-zero element has an inverse, hence division is well-defined). Exchanging Rows 2 and 3, we get

$$\begin{pmatrix} 1 & x & 0 & 0 \\ 0 & 1 & 0 & \alpha_j \\ 0 & 0 & 1 & x \end{pmatrix}$$

The rank of this matrix is 3, and thus it has exactly q solutions. This means that every β_j is correct for exactly q secret key quadruples (x_1, x_2, y_1, y_2) out of the q^2 that are possible from the point of view of the attacker. Thus any guess at β_j, and thus v_j, is correct with probability $\frac{1}{q}$. This is exponentially small (in the binary size of the group order G_q, i.e., in $\log q$). With each error message the attacker gets, one β_j and thus q keys are excluded. However, only a polynomial number of attempts is possible, and thus even at the end of the attack, the success probability is not significantly larger than $\frac{1}{q}$. ∎

Behavior if k randomly chosen in G_q: For this case, we want to show that the view of A_{CS}, i.e., everything it sees, is essentially independent of b. Then it is clear that the output b^* of A_{CS} is also independent of b. Thus the probability of $b^* = b$ is almost exactly $\frac{1}{2}$, and thus also the probability that $b_{\mathsf{DDH}} = 0$. This will be the significant difference between the two cases.

Intuitively, we show that for this wrong scenario with a random k, the encryption is good even without the Diffie-Hellman assumption.

The primary source for A_{CS} to learn anything about b is the ciphertext c, which $\mathsf{A}_{\mathsf{DDH}}$ constructs as

$$(i, k, c^*, v)$$

with

$$c^* = i^{z_1} \cdot k^{z_2} \cdot m_b^*, \quad v = i^{x_1+y_1\alpha} \cdot k^{x_2+y_2\alpha},$$

where $\alpha := \mathsf{H}(i, k, c^*)$ as usual. To show that this is independent of b, we show that $k^* = i^{z_1} \cdot k^{z_2}$ is as good as a multiplicative one-time pad for m_b^*. (It is easy to see that one-time pads can be made in any group; here is such a surrounding protocol where it makes sense not to use XOR.) In other words, we have to show that with all the knowledge the attacker has, all values of k^* are still equally likely. This can only be because z_1 and z_2 are secret. An attacker might obtain information about them from the following sources:

- The public key contains $h_{CS} = g_1^{z_1} \cdot g_2^{z_2}$. For a computationally unrestricted attacker, this means a linear equation
$$z = z_1 + xz_2. \tag{8.4}$$
- Decryption of correct ciphertexts. Here the attacker learns $k_j = i_{j,1}^{z_1} \cdot i_{j,2}^{z_2} = (g_1^{z_1} \cdot g_2^{z_2})^{r_j} = h_{CS}^{r_j}$. As he already knows h_{CS}, this is no new information about (z_1, z_2).
- The encryption is what we are just considering.
- Answers on incorrect ciphertexts. In Lemma 8.2, we will show that almost certainly all these answers will be the fixed error message. (Lemma 8.1 does not imply this because there we used that $k = g^{xy}$.)

In summary, except for the error messages, the attacker obtains at most one linear equation (Equation 8.4) about the two secret variables. Thus q pairs (z_1, z_2) are still possible from his point of view. The equation $k^* = i^{z_1} \cdot k^{z_2}$ can be rewritten as

$$\gamma = y \cdot z_1 + \delta \cdot z_2, \tag{8.5}$$

where γ and δ are defined such that $k^* = g_1^{\gamma}$ and $k = g_1^{\delta}$. Except in the one case where k, in spite of being randomly chosen, happens to be the correct Diffie-Hellman key g^{xy}, this equation is linearly independent from the one above. Hence, each $k^* \in G_q$ occurs for exactly one choice of (z_1, z_2). In other words, each k^* is equally probable from the point of view of the attacker, and thus it is a One-time Pad that perfectly hides m_b^* and thus b.

Lemma 8.2 *If k is randomly chosen from G_q, the probability that A_{CS} succeeds in sending any ciphertext c_j whose first two components are not of the form $i_{j,1} = g_1^{r_j}$, $i_{j,2} = g_2^{r_j}$ for some r_j, but which nevertheless passes the verification, is negligibly small.* $\square$

Proof. We start the proof as for Lemma 8.1: We fix an arbitrary ciphertext $(i_{j,1}, i_{j,2}, c_j^*, v_j)$ where $i_{j,1} = g_1^{r_j}$ and $i_{j,2} = g_2^{r_j^*}$ with $r_j^* \neq r_j \bmod q$. It passes the verification if and only if

$$\beta_j = r_j(x_1 + y_1\alpha_j) + xr_j^*(x_2 + y_2\alpha_j), \tag{8.6}$$

where $v_j = g_1^{\beta_j}$. Again, the attacker $\mathsf{A_{CS}}$ can only obtain information about (x_1, x_2, y_1, y_2) from the following sources:

- The components s and t of pk, which give linear equations

$$\sigma = x_1 + xx_2, \quad \tau = y_1 + xy_2. \tag{8.7}$$

- The encryption, and only its component $v = i^{x_1+y_1\alpha} \cdot k^{x_2+y_2\alpha}$. Here is the difference from Lemma 8.1, because now this is not simply $s^y \cdot t^{y\alpha}$ and can therefore contain no new information about (x_1, x_2, y_1, y_2). We can rewrite this as

$$\epsilon = y(x_1 + y_1\alpha) + \delta(x_2 + y_2\alpha), \tag{8.8}$$

  where $k = g_1^\delta$ as in Equation 8.5 and $v = g_1^\epsilon$.

- Answers on incorrect ciphertexts. For these, we are just proving that they are almost certainly only the fixed error message.

In summary, except for the error messages, the attacker now obtains three linear equations about the four secret variables. Again we write them as a matrix, and in the last row we write the equation for β_j:

$$\begin{pmatrix} 1 & x & 0 & 0 \\ 0 & 0 & 1 & x \\ y & \delta & y\alpha & \delta\alpha \\ r_j & xr_j^* & r_j\alpha_j & xr_j^*\alpha_j \end{pmatrix}$$

We transform Rows 1, 2, and 4 as in Lemma 8.1 and move Row 3 down:

$$\begin{pmatrix} 1 & x & 0 & 0 \\ 0 & 1 & 0 & \alpha_j \\ 0 & 0 & 1 & x \\ y & \delta & y\alpha & \delta\alpha \end{pmatrix}$$

If we subtract Rows 1 and 3 from 4 with the appropriate coefficients, Row 4 becomes

$$\begin{pmatrix} 0 & \delta - xy & 0 & \alpha(\delta - xy) \end{pmatrix}$$

and finally, subtracting Row 2,

$$\begin{pmatrix} 0 & 0 & 0 & (\alpha - \alpha_j)(\delta - xy) \end{pmatrix}$$

If $(\alpha - \alpha_j)(\delta - xy) \neq 0$, the rank of the matrix is 4. This also implies that any three of the original equations have rank 3 and thus q solutions. Thus q secret key quadruples (x_1, x_2, y_1, y_2) are possible from the point of view of the attacker, and only for one of them, β_j is correct. This is an exponentially small probability $\frac{1}{q}$.

Now we have to show whether the attacker can achieve either $\alpha = \alpha_j$ or $\delta = xy$. (As to an intuitive idea why the case $\alpha = \alpha_j$ is special, recall the paragraph "some intuition" we gave you earlier in this chapter in Section 8.2.2.)

- $\delta = xy$ would mean that $k = g_1^{xy}$, i.e., k happens to be the correct Diffie-Hellman key. This only happens with exponentially small probability.
- $\alpha = \alpha_j$ would mean $\mathsf{H}(i, k, c^*) = \mathsf{H}(i_{j,1}, i_{j,2}, c_j^*)$. If $(i_{j,1}, i_{j,2}, c_j^*) \neq (i, k, c^*)$, then the attacker would have found a collision of the hash function, which is assumed to be infeasible.[3]

  Otherwise, the components i, k, c^* determine the only correct v uniquely, and thus $c_j = c$. If c_j is sent after c, $\mathsf{A_{DDH}}$ would refuse to answer. And if c_j is sent before c, this would mean that the attacker $\mathsf{A_{CS}}$ guessed the random values i and k correctly although so far he obtained no information at all about them.

Thus in fact, at the beginning any β_j is only correct for one out of q pairs (z_1, z_2) that are possible from the point of view of the attacker, and each error message excludes only one such pair, so that the success probability of the attacker remains exponentially small. ■

This finishes the proof that b is well hidden from $\mathsf{A_{CS}}$ in the case where k is random, and thus the overall proof of the security of the Cramer-Shoup system according to the basic idea explained at the beginning. ■

[3] You might ask whether it was really the attacker, because (i, k, c^*) came from the (simulated) CCA2 challenger. However, the CCA2 challenger never exploited the fact that he himself chose pk_{hash}. Hence if $\mathsf{A_{CS}}$ could achieve this case, one could easily construct a successful attacker against the hash functions from it by letting that attacker play the roles of both $\mathsf{A_{DDH}}$ with random i, k and $\mathsf{A_{CS}}$.