A Note on an Encryption Scheme of Kurosawa and Desmedt

Rosario Gennaro* Victor Shoup†

August 10, 2004

Abstract

Recently Kurosawa and Desmedt presented a new hybrid encryption scheme which is secure against adaptive chosen-ciphertext attack. Their scheme is a modification of the Cramer-Shoup encryption scheme. Its major advantage with respect to Cramer-Shoup is that it saves the computation of one exponentiation and produces shorter ciphertexts. However, the proof presented by Kurosawa and Desmedt relies on the use of information-theoretic key derivation and message authentication functions.

In this note we present a different proof of security which shows that the Kurosawa-Desmedt scheme can be instantiated with any computationally secure key derivation and message authentication functions, thus extending the applicability of their paradigm, and improving its efficiency.

1 Introduction

The notion of *chosen-ciphertext security* was introduced by Naor and Yung [5] and developed by Rackoff and Simon [6], and Dolev, Dwork, and Naor [3].

In a chosen ciphertext attack, the adversary is given access to a *decryption oracle* that allows him to obtain the decryptions of ciphertexts of his choosing. Intuitively, security in this setting means that an adversary obtains (effectively) no information about encrypted messages, provided the corresponding ciphertexts are never submitted to the decryption oracle.

*IBM T.J.Watson Research Center, Yorktown Heights, NY, USA. rosario@watson.ibm.com

†Computer Science Dept. NYU. shoup@cs.nyu.edu

As shown in [3], security against chosen-ciphertext attack is equivalent to the notion of *non-malleability*. An encryption scheme is said to be non-malleable if given a ciphertext c, it is infeasible to compute a ciphertext c' whose decryption is somewhat related to the decryption of c.

For these reasons, the notion of chosen-ciphertext security has emerged as the "right" notion of security for encryption schemes. Indeed it can be shown that in order to model encryption as a "secure envelope", then the encryption scheme used must be chosen-ciphertext secure.

A number of chosen ciphertext secure cryptosystems have been proposed in the literature. The first schemes were presented in [5, 6, 3], but they were quite impractical. The first truly practical cryptosystem that is provably secure against chosen ciphertext attack was discovered by Cramer and Shoup [1]. The security of this scheme is based on the hardness of the decisional Diffie-Hellman problem. In [2] Cramer and Shoup show that their original scheme is an instance of a more generic paradigm, which can be also instantiated with the Quadratic Residuosity and N-Residuosity assumptions.

In [7] Shoup presents an *hybrid* variant of the Cramer-Shoup cryptosystem. This scheme uses the original public-key scheme to generate an encryption of a random group element κ. Then a key derivation function (KDF) is applied to κ to compute two keys k, K which are used to encrypt the actual message with a chosen-ciphertext secure symmetric encryption scheme (recalled below).

Differently than in the public-key case, symmetric encryption schemes which are secure against a chosen-ciphertext attack can be easily built out of weaker primitives. It is indeed well known that all you need is a symmetric encryption scheme E which is secure against passive adversaries, and a secure message authentication code (MAC). To encrypt a message m with keys k, K it is sufficient to encrypt m with K, i.e. compute $e = E_K(m)$, and then compute a message authentication tag for e using k, i.e. compute $t = MAC_k(e)$. The final ciphertext is (e, t). The receiver, who also holds k, K, first checks that the tag t is correct and only in that case decrypts e.

Recently Kurosawa and Desmedt [4] modified the hybrid scheme presented in [7]. The advantage of their modification is that the computation of a ciphertext in their scheme requires one less exponentiation and produces shorter ciphertexts.

However their proof of security relies on the use of information theoretically secure KDF and MAC functions in the symmetric step of the hybrid construction. There are several reasons why this is not desirable, among them:

efficiency The proof in [4] requires the key k to be statistically close to a random key. This means that we cannot use a pseudo-random generator to derive k from a random group element encrypted during the public-key phase. This in turns implies that the public key part of the scheme must be instantiated with larger security parameters which would result in slower execution times[1];

modularity we would like to have a scheme into which we can plug any secure component and it still remains secure. It would be hard to deploy a scheme in large-scale if it can be used only in conjunction with certain types of MACs and KDFs (and in particular, with KDFs and MACs that are not used at all by the designers of standard cryptographic algorithms).

In this note we show a new and different proof of security for the Kurosawa-Desmedt scheme. We show that it is indeed possible to use any secure key derivation function and message authentication code. This effectively improves the efficiency and applicability of their scheme.

2 The scheme

In this section we recall the Kurosawa-Desmedt scheme from [4]. We describe it using generic building blocks and at the end of the section we point out where the proof of security in [4] requires information theoretic security. The scheme makes use of:

- a group G of prime order q, with (random) generators g_1 and g_2.

  Security assumption (DDH): Hard to distinguish (g_1^r, g_2^r) *from* $(g_1^r, g_2^{r'})$*, where* r *is a random element of* $\mathbb{Z}_q$ *and* r' *is a random element of* $\mathbb{Z}_q \setminus \{r\}$*.*

- a message authentication code MAC, that for key k and message $e \in \{0,1\}^*$, produces a "tag" $t := MAC_k(e)$.

  Security assumption: For random k*, after (optionally) obtaining* $MAC_k(e^*)$ *for adversarially chosen* e^**, hard to compute* $MAC_k(e)$ *for adversarially chosen* e *(different from* e^**, of course).*

[1] For typical security parameters, this increase in computation times totally offsets the gain from performing one less exponentiation, thus making the Kurosawa-Desmedt scheme as efficient as the original Cramer-Shoup

- a symmetric key encryption scheme, with encryption algorithm E and decryption algorithm D, such that for key K and plaintext $m \in \{0,1\}^*$, $e := E_K(m)$ is the encryption of m under K, and for key K and ciphertext $e \in \{0,1\}^*$, $m := D_K(m)$ is the decryption of e under K.

  Security assumption (semantic security): hard to distinguish $E_K(m_0)$ from $E_K(m_1)$ for randomly chosen K and adversarially chosen m_0 and m_1 (where m_0 and m_1 are of equal length).

- a key derivation function KDF, such that for $v \in G$, $KDF(v) = (k, K)$, where k is a message authentication key, and K is a symmetric encryption key.

  Security assumption: hard to distinguish $KDF(v)$ from (k, K), where v, k and K are random.

- a hash function $H : G \times G \to \mathbb{Z}_q$.

  Security assumption (target collision resistance): given $u_1^* := g_1^r$ and $u_2^* := g_2^r$, for random $r \in \mathbb{Z}_q$, hard to find $(u_1, u_2) \in G \times G \setminus \{(u_1^*, u_2^*)\}$ such that $H(u_1, u_2) = H(u_1^*, u_2^*)$.

Note that the key space for the message authentication code is assumed to consist of all bit strings of a given length, so that by a random key k, we mean a random bit string of appropriate length. Similarly for the symmetric encryption keys.

Note also that KDF and H may have associated keys (which are publicly known).

Key Generation: The description of the group G is generated, along with random generators g_1 and g_2 for G. Any keys for KDF and H are also generated. Then:

$$x_1, x_2, y_1, y_2 \overset{\text{¢}}{\leftarrow} \mathbb{Z}_q,\ c \leftarrow g_1^{x_1} g_2^{x_2},\ d \leftarrow g_1^{y_1} g_2^{y_2}.$$

The public key consists of the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with the group elements c and d. The private key consists of the public key, along with x_1, x_2, y_1, y_2.

Encryption of $m \in \{0,1\}^*$:

$$\begin{aligned}
&r \overset{\text{¢}}{\leftarrow} \mathbb{Z}_q,\ u_1 \leftarrow g_1^r \in G,\ u_2 \leftarrow g_2^r \in G,\ \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q \\
&v \leftarrow c^r d^{r\alpha} \in G,\ (k, K) \leftarrow KDF(v),\ e \leftarrow E_K(m),\ t \leftarrow MAC_k(e) \\
&\text{output } C := (u_1, u_2, e, t)
\end{aligned}$$

Decryption of $C = (u_1, u_2, e, t)$:

$\alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q,\ v \leftarrow u_1^{x_1+y_1\alpha} u_2^{x_2+y_2\alpha} \in G,\ (k, K) \leftarrow KDF(v)$
if $t \neq MAC_k(e)$ then
  reject
else
  $m \leftarrow D_K(e)$
  output m

In addition to the above computational security assumption, the proof of security in [4] requires the following *information theoretic* assumptions:

- *information-theoretically secure KDF.* If $v \in G$ is random, then at least the first component k of the output of $KDF(v)$ should be (statistically close to) uniform.

- *information-theoretically secure MAC.* For all e and t, if k is chosen at random, then $\Pr[MAC_k(e) = t]$ is negligible.

Our proof of security, described in the next section, does *not* need these conditions.

In Appendix A we recall the Cramer-Shoup hybrid scheme from [7] and compare the two schemes. In particular we point out how for typical security parameters the gains posted by the Kurosawa-Desmedt scheme may be offset by the requirement that KDF and MAC be information theoretically secure.

3 Security proof

Game 0

We now define a game, called *Game 0*, which is an interactive computation between an *adversary* and a *simulator.* This game is simply the usual game used to define CCA security, in which the simulator provides the adversary's environment.

Initially, the simulator runs the key generation algorithm, obtaining the description of G, generators g_1 and g_2, keys for KDF and H (if any), along with the values $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$ and $c, d \in G$. The simulator gives the public key to the adversary.

During the execution of the game, the adversary makes a number of "decryption requests." Assume these requests are $C^{(i)}, \ldots, C^{(Q)}$, where

$$C^{(i)} = (u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)}).$$

For each such request, the simulator decrypts the given ciphertext, and gives the adversary the result. We denote by $\alpha^{(i)}$, $v^{(i)}$, $k^{(i)}$, and $K^{(i)}$ the corresponding intermediate quantities computed by the decryption algorithm on input $C^{(i)}$.

The adversary may also make a single "challenge request." For such a request, the adversary submits two messages m_0, m_1, which are bit strings of equal length, to the simulator; the simulator chooses $b \in \{0, 1\}$ at random, and encrypts m_b, obtaining the "target ciphertext" $C^* = (u_1^*, u_2^*, e^*, t^*)$. The simulator gives C^* to the adversary. We denote by r^*, α^*, v^*, k^*, and K^* the corresponding intermediate quantities computed by the encryption algorithm.

The only restriction on the adversary's requests is that after it makes a challenge request, subsequent decryption requests must not be the same as the target ciphertext.

At the end of the game, the adversary outputs $\hat{b} \in \{0, 1\}$.

Let X_0 be the event that $\hat{b} = b$. Security means that $|\Pr[X_0] - 1/2|$ should be negligible.

We prove this by considering other games, *Game 1*, *Game 2*, etc. These games will be quite similar to Game 0 in their overall structure, and will only differ from Game 0 in terms of how the simulator works. However, in each game, there will be well defined bits $\hat{b}$ and b, so that in Game i, we always define X_i to the event that $\hat{b} = b$ in that game. All of these games should be viewed as operating on the same underlying probability space.

Before moving on, we make a couple of additional assumptions about the internal structure of Game 0 that will be convenient down the road. First, we assume that g_2 is computed as:

$$w \xleftarrow{\$} \mathbb{Z}_q^*, \ g_2 \leftarrow g_1^w.$$

Note that the value of w is never explicitly used in Game 0, except to compute g_2. Second, we assume that the quantities r^*, u_1^*, u_2^*, α^*, v^*, k^*, and K^* are computed at the very start of the game (they do not depend on the values m_0, m_1 provided later by the adversary, so this can be done).

Game 1

Game 1 is the same as Game 0, except that if the adversary ever submits $C^{(i)}$ for decryption with

$$(u_1^{(i)}, u_2^{(i)}) \neq (u_1^*, u_2^*) \ \text{ and } \ \alpha^{(i)} = \alpha^*,$$

the simulator *rejects* the given ciphertext.

In Game 1, the simulator may reject ciphertexts that would not have been rejected in Game 0. Let us call Rejection Rule 0 the rule by which ciphertexts are rejected as in the ordinary decryption algorithm (i.e., the message authentication tags do not match). Let us call Rejection Rule 1 this new rejection rule, introduced in Game 1.

Let F_1 be the event that the simulator applies Rejection Rule 1 in Game 1 to a ciphertext to which Rejection Rule 0 does not apply. Because Game 0 and Game 1 proceed identically until the this event occurs, we have

$$|\Pr[X_0] - \Pr[X_1]| \leq \Pr[F_1]. \tag{1}$$

Moreover, we have

$$\Pr[F_1] \leq \epsilon_{\text{tcr}}, \tag{2}$$

where ϵ_{tcr} is the success probability that one can find a collision in H using resources similar to those of the given adversary. By assumption, ϵ_{tcr} is negligible.

Game 2

Game 2 is the same as Game 1, except that the simulator computes v^* as

$$v^* \leftarrow (u_1^*)^{x_1+y_1\alpha^*}(u_2^*)^{x_2+y_2\alpha^*}.$$

This change is purely conceptual, since v^* has the same value either way. In particular,

$$\Pr[X_1] = \Pr[X_2]. \tag{3}$$

Game 3

Now generate u_2^* by the rule

$$r' \stackrel{\$}{\leftarrow} \mathbb{Z}_q \setminus \{r^*\},\ u_2^* \leftarrow g_2^{r'}.$$

We have

$$|\Pr[X_2] - \Pr[X_3]| \leq \epsilon_{\text{ddh}}, \tag{4}$$

where ϵ_{ddh} is the advantage with which one can solve the DDH problem, using resources similar to those of the given adversary. By assumption, ϵ_{ddh} is negligible.

> **The details.** We can easily build a "hybrid" Game 2/3 that takes $\tau := (g_1, g_2, u_1^*, u_2^*)$ as input, so that if τ is a random DH-tuple, Game 2/3 acts just like Game 2, and if τ is a random non-DH-tuple, then Game 2/3 acts just like Game 3. The distinguishing algorithm runs Game 2/3 on input τ, and outputs 1 if $\hat{b} = b$, and outputs 0 otherwise. The distinguishing advantage of this algorithm is exactly equal to $|\Pr[X_2] - \Pr[X_3]|$.

Game 4

In this game, the simulator makes use of the value $w \in \mathbb{Z}_q$, where $g_2 = g_1^w$. The simulator did not need to make explicit use of this value in previous games. Indeed, we could not have used the DDH assumption if the simulator had to use w. However, we are now finished with the DDH assumption, and so the simulator is free to make use of w in this and subsequent games.

Game 4 is the same as Game 3, except that we introduce a new Rejection Rule 2: in responding to decryption requests, the simulator *rejects* any ciphertext $C^{(i)}$ such that

$$(u_1^{(i)})^w \neq u_2^{(i)},$$

which is equivalent to saying that

$$\log_{g_1} u_1^{(i)} \neq \log_{g_2} u_2^{(i)}.$$

Define F_4 to be the event that a ciphertext is rejected during Game 4 using Rejection Rule 2 to which Rejection Rules 0 and 1 are not applicable.

Clearly, we have

$$|\Pr[X_3] - \Pr[X_4]| \leq \Pr[F_4], \tag{5}$$

and we want to show that $\Pr[F_4]$ is negligible.

We postpone this until later. However, at this point we augment Game 4 just slightly: the simulator chooses $j \in \{1, \ldots, Q\}$, and we define F_4' to be the event that in Game 4, Rejection Rules 0 and 1 do not apply to $C^{(j)}$, but Rejection Rule 2 does apply to $C^{(j)}$. Clearly,

$$\Pr[F_4] \leq Q \Pr[F_4'], \tag{6}$$

and so it suffices to show that $\Pr[F_4']$ is negligible.

Game 5

To motivate Game 5, we begin with some observations about Game 4. Let $x := x_1 + wx_2$ and $y := y_1 + wy_2$. Then we have

$$c = g_1^x \text{ and } d = g_1^y.$$

Also, for $i = 1, \ldots, Q$, if

$$\log_{g_1} u_1^{(i)} = \log_{g_2} u_2^{(i)}$$

then

$$v^{(i)} = u_1^{x+y\alpha^{(i)}}.$$

Moreover, v^* is uniformly distributed over G, independently of x and y. Further, if

$$\alpha^{(j)} \neq \alpha^* \text{ and } \log_{g_1} u_1^{(j)} \neq \log_{g_2} u_2^{(j)},$$

then $v^{(j)}$ is uniformly distributed over G, independently of x, y, and v^*. These observations follow from simple linear algebra considerations, as in [1].

Based on these observations, in Game 5, we compute a number of quantities in a different, but equivalent, manner. Let $\bar{x}, \bar{y}$ be random elements of $\mathbb{Z}_q$, and let $\bar{v}_1$ and $\bar{v}_2$ be random elements of G. Let

$$(\bar{k}_1, \bar{K}_1) := KDF(\bar{v}_1) \text{ and } (\bar{k}_2, \bar{K}_2) := KDF(\bar{v}_2).$$

The key generation algorithm is modified as follows:

$$c \leftarrow g_1^{\bar{x}},\ d \leftarrow g_1^{\bar{y}}.$$

The values k^* and K^* are computed as:

$$(k^*, K^*) \leftarrow (\bar{k}_1, \bar{K}_1).$$

In processing decryption requests, for a given $C^{(i)}$ that is not subject to Rejections Rules 1 or 2, the value $v^{(i)}$ is computed as

$$v^{(i)} \leftarrow (u_1^{(i)})^{\bar{x}+\bar{y}\alpha^{(i)}}.$$

Finally, we define the event F_5' to be the event in Game 5 that $C^{(j)}$ is subject to Rejection Rule 2, $C^{(j)}$ is not subject to Rejection Rule 1, and

$$t^{(j)} = \begin{cases} MAC_{\bar{k}_1}(e^{(j)}) & \text{if } (u_1^{(j)}, u_2^{(j)}) = (u_1^*, u_2^*); \\ MAC_{\bar{k}_2}(e^{(j)}) & \text{otherwise.} \end{cases}$$

Note that the values $x_1, x_2, y_1, y_2, v^*, v^{(j)}$ are not used in Game 5.

We claim that

$$\Pr[X_4] = \Pr[X_5] \tag{7}$$

and

$$\Pr[F_4'] = \Pr[F_5']. \tag{8}$$

This follows from the observations above — we have simply replaced one set of random variables by another set with same joint distribution.

It is perhaps helpful at this point to state how Game 5 works, starting from scratch:

- The simulator generates the description of G, along with a random generator g_1, and any keys for KDF and H. It computes

  $$\begin{array}{l} w \xleftarrow{\text{¢}} \mathbb{Z}_q^*,\ g_2 \leftarrow g_1^w \\ \bar{x}, \bar{y} \xleftarrow{\text{¢}} \mathbb{Z}_q,\ c \leftarrow g_1^{\bar{x}},\ d \leftarrow g_1^{\bar{y}} \\ r^* \xleftarrow{\text{¢}} \mathbb{Z}_q,\ r' \xleftarrow{\text{¢}} \mathbb{Z}_q \setminus \{r^*\},\ u_1^* \leftarrow g_1^{r^*},\ u_2^* \leftarrow g_1^{wr'},\ \alpha^* \leftarrow H(u_1^*, u_2^*) \\ \bar{v}_1 \xleftarrow{\text{¢}} G,\ (\bar{k}_1, \bar{K}_1) \leftarrow KDF(\bar{v}_1) \\ \bar{v}_2 \xleftarrow{\text{¢}} G,\ (\bar{k}_2, \bar{K}_2) \leftarrow KDF(\bar{v}_2) \\ j \xleftarrow{\text{¢}} \{1, \ldots, Q\} \end{array}$$

  The simulator gives the description of G, the generators g_1 and g_2, keys for KDF and H (if any), along with c and d to the adversary.

- In processing a decryption request $C^{(i)} = (u_1^{(i)}, u_2^{(i)}, e^{(i)}, t^{(i)})$, the simulator first checks if $(u_1^{(i)})^w \neq u_2^{(i)}$; if so, the ciphertext is rejected. Otherwise, the simulator computes

  $$\alpha^{(i)} \leftarrow H(u_1^{(i)}, u_2^{(i)})$$

  and checks if $(u_1^{(i)}, u_2^{(i)}) \neq (u_1^*, u_2^*)$ and $\alpha^{(i)} = \alpha^*$; if so, the ciphertext is rejected. Otherwise, the simulator computes

  $$v^{(i)} \leftarrow u_1^{\bar{x}+\bar{y}\alpha^{(i)}},\ (k^{(i)}, K^{(i)}) \leftarrow KDF(v^{(i)}).$$

  It then tests if $t^{(i)} = MAC_{k^{(i)}}(e^{(i)})$; if not, the ciphertext is rejected. Otherwise, the simulator returns $D_{K^{(i)}}(e^{(i)})$ to the adversary.

- In processing the challenge request, the adversary gives m_0, m_1 to the simulator. The simulator computes

  $$b \xleftarrow{\text{¢}} \{0,1\},\ e^* \leftarrow E_{\bar{K}_1}(m_b),\ t^* \leftarrow MAC_{\bar{k}_1}(e^*),$$

  and gives $C^* := (u_1^*, u_2^* e^*, t^*)$ to the adversary.

Note that the values j and $\bar{v}_2$ (and the derived values $\bar{k}_2$ and $\bar{K}_2$) are not used in this game, other than to define the event F_5'.

Game 6

Game 6 is the same as Game 5, except that instead of applying *KDF* to derive the keys $\bar{k}_1, \bar{K}_1, \bar{k}_2, \bar{K}_2$, these keys are simply generated at random. Define the event F_6' in Game 6 in the same way as it was defined in Game 5.

It is easy to see that

$$|\Pr[X_5] - \Pr[X_6]| \leq 2\epsilon_{\text{kdf}} \tag{9}$$

and

$$|\Pr[F_5'] - \Pr[F_6']| \leq 2\epsilon_{\text{kdf}}, \tag{10}$$

where ϵ_{kdf} is the advantage of distinguishing the output of the *KDF* from a random key pair, using resources similar to those of the given adversary. The factor of 2 comes from applying a standard "hybrid" argument to the two *KDF* outputs to be distinguished in moving from Game 5 to Game 6. By assumption, ϵ_{kdf} is negligible.

We claim that

$$|\Pr[X_6] - 1/2| \leq \epsilon_{\text{enc}}, \tag{11}$$

where ϵ_{enc} is the probability of breaking the semantic security of the underlying symmetric key encryption scheme, using resources similar to those of the given adversary. This follows by construction — note that the key $\bar{K}_1$ in Game 6 is random, and is used for no other purpose than to encrypt the challenge plaintext. By assumption, ϵ_{enc} is negligible.

We also claim that

$$\Pr[F_6'] \leq 2\epsilon_{\text{mac}}, \tag{12}$$

where ϵ_{mac} is the probability of breaking the message authentication code, using resources similar to those of the given adversary. This also follows by construction — one has to make a simple "hybrid" argument to account for the fact that we are breaking one out of two message authentication schemes (one keyed with $\bar{k}_1$ and the other keyed with $\bar{k}_2$, whence the factor of 2). By assumption, ϵ_{mac} is negligible.

We are now in a position to complete the proof of security. We have

$$\begin{aligned}
\Pr[F_4] &\le Q \Pr[F_4'] && \text{[by (6)]} \\
&= Q \Pr[F_5'] && \text{[by (8)]} \\
&\le Q(\Pr[F_6'] + 2\epsilon_{\mathrm{kdf}}) && \text{[by (10)]} \\
&\le Q(2\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}) && \text{[by (12)]}
\end{aligned}$$

Thus, we have

$$\Pr[F_4] \le Q(2\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}). \tag{13}$$

Finally, combining (1), (2), (3), (4), (5), (7), (9), (11), and (13), we have:

$$|\Pr[X_0] - 1/2| \le \epsilon_{\mathrm{tcr}} + \epsilon_{\mathrm{ddh}} + 2\epsilon_{\mathrm{kdf}} + \epsilon_{\mathrm{enc}} + Q(2\epsilon_{\mathrm{mac}} + 2\epsilon_{\mathrm{kdf}}). \tag{14}$$

By assumption, the right-hand side of (14) is negligible, which finishes the proof.

4 Hash Proof Systems

In [2] Cramer and Shoup showed that their original scheme in [1] was a special instance of a generic paradigm based on *hash proof systems*. We briefly recall here the basic ideas and how they can be applied to the scheme described in the previous section.

Smooth projective hashing [2]: Let X be a set and $L \subset X$ a language. Loosely speaking, a hash function H_a that maps X to some set is projective if there exists a projection key that defines the action of H_a over the subset L of the domain X. That is, there exists a projection function $\alpha(\cdot)$ that maps keys k into their projections $s = \alpha(a)$. The projection key s is such that for every $x \in L$ it holds that the value of $H_a(x)$ is uniquely determined by s and x. In contrast, nothing is guaranteed for $x \notin L$, and it may not be possible to compute $H_a(x)$ from s and x. A smooth projective hash function has the additional property that for $x \notin L$, the projection key s actually says *nothing* about the value of $H_a(x)$. More specifically, given x and $s = \alpha(a)$, the value $H_a(x)$ is uniformly distributed (or statistically close) to a random element in the range of H_a.

An interesting feature of smooth projective hashing is that if L is an NP-language, then for every $x \in L$ it is possible to efficiently compute $H_a(x)$ using the projection key $s = \alpha(a)$ and a witness of the fact that $x \in L$. Alternatively, given a itself, it is possible to efficiently compute $H_a(x)$ even without knowing a witness.

Using the techniques from [2], Kurosawa and Desmedt in [4] generalize the above scheme can be generalized using smooth projective hashing as follows. The sets X, L and a projection key $s = \alpha(a)$ will be the public key. The key a will be the secret key.

To encrypt m, the sender chooses an element $x \in L$ together with a witness. He then computes $v = H_a(x)$ using the projection s and the witness. Then the keys $(k, K) = \mathit{KDF}(v)$ are derived as above. The rest of the encryption procedure remains the same, i.e., $e = E_K(m)$ and $t = MAC_k(e)$. The ciphertext is x, e, t.

The receiver on input (x, e, t) computes $v' = H_a(x)$ and $(k, K) = \mathit{KDF}(v')$. If $t = MAC_k(e)$ then it decrypts $m = D_K(e)$.

Security Analysis. As in the proof in [2] the basic computational assumption underlying the security of this scheme is that it is hard to distinguish between random elements in L and random elements outside of L.

The proof of security in [4] requires the projective hash function to be *strongly 2-universal*, which is a stronger condition than smoothness. Basically it is required that for $x \notin L$, even given $s = \alpha(a)$ *and the value* $H_a(x')$ *for* $x' \notin L$ *and* $x' \neq x$, the distribution of the value $H_a(x)$ is statistically close to the uniform distribution over the range of H_a. Their generalized scheme, however, still requires information-theoretically secure *KDF* and MAC functions.

Our proof, which lifts such requirements on the *KDF* and MAC functions, also generalizes assuming strong 2-universal projective hashing, that one can efficiently sample elements outside of L, and there is a trapdoor that allows for efficiently testing language membership.

References

[1] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98*.

[2] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. In *EuroCrypt'02*.

[3] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *STOC'91*, pages 542–552, 1991.

[4] K. Kurosawa and Y. Desmedt. A New Paradigm of Hybrid Encryption Scheme. To appear in *CRYPTO'04*. Available on-line at `http://kuro.cis.ibaraki.ac.jp/~kurosawa/04.html`.

[5] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC'90*, pages 427–437, 1990.

[6] C. Rackoff and D. Simon. Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology–Crypto '91*, pages 433–444, 1991.

[7] V. Shoup Using hash functions as a hedge against chosen ciphertext attack. In *EuroCrypt'00*.

A The original scheme

The Cramer-Shoup hybrid encryption scheme proposed in [1], and refined in [7], uses the same tools as the one described above. However key generation, encryption and decryption algorithms are different.

Key Generation: The description of the group G is generated, along with a random generator g_1 for G. Any keys for *KDF* and H are also generated. Then:

$$w, x, y, z \xleftarrow{\$} \mathbb{Z}_q,\ g_2 \leftarrow g_1^w,\ c \leftarrow g_1^x,\ d \leftarrow g_1^y,\ h \leftarrow g_1^z.$$

The public key consists of the description of G, the generators g_1 and g_2, keys for *KDF* and H (if any), along with the group elements c, d, h. The private key consists of the public key, along with w, x, y, z.

Encryption of $m \in \{0,1\}^*$:

$r \xleftarrow{\$} \mathbb{Z}_q,\ \kappa \leftarrow h^r,\ u_1 \leftarrow g_1^r \in G,\ u_2 \leftarrow g_2^r \in G,\ \alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q$
$v \leftarrow c^r d^{r\alpha} \in G,\ (k, K) \leftarrow KDF(\kappa),\ e \leftarrow E_K(m),\ t \leftarrow MAC_k(e)$
output $C := (u_1, u_2, v, e, t)$

Decryption of $C = (u_1, u_2, v, e, t)$:

$\alpha \leftarrow H(u_1, u_2) \in \mathbb{Z}_q,\ v' \leftarrow u_1^{x+y\alpha} \in G,\ \kappa' \leftarrow u_1^z,\ (k, K) \leftarrow KDF(\kappa')$
if $t \neq MAC_k(e)$ or $v' \neq v$ or $u_2 \neq u_1^w$ then
    reject
else
    $m \leftarrow D_K(e)$
    output m

Notice that compared to the Kurosawa-Desmedt scheme, the encryption algorithm in this scheme computes an extra exponentiation (the computation of κ) and a longer ciphertext (it includes the group element v). However, that does not translate into a direct gain in efficiency.

In the Cramer-Shoup scheme we can choose the prime q to be 160-bit long. This results in a random value κ which is computationally indistinguishable from a random group element. Then, under a suitable computational assumption on KDF, we can derive keys k, K of any required length using a pseudo-random number generator.

On the other hand, the key k in the Kurosawa-Desmedt scheme must be derived from v in an information-theoretic way. We can't apply a pseudo-random number generator, otherwise we lose the information-theoretic security. For common security parameters k is required to be at least 170-bits long. The only way we know how to do this is to map v into an 160-bit string using universal hashing and the Entropy Smoothing Theorem. But this requires v to come from a distribution with min-entropy at least, say, 320. Considering that from κ we also need to derive the key K (say, another 128 bits), then it seems that the group G must have order q of at least about 450 bits. This increase in the security parameter clearly offsets the gain obtained by dropping one exponentiation.

Using our proof, however, we can claim that the Kurosawa-Desmedt scheme can be used with a group G of order q where q is a 160-bit prime.

We also note that the scheme in [7] is optimized so that all exponentiations in the decryption algorithm are with respect to the same base — this allows for speedups using techniques for exponentiation with preprocessing. We believe that similar optimizations can be applied to the Kurosawa-Desmedt scheme. Also, the scheme in [7] can be proven secure in the random oracle model under the *computational* Diffie-Hellman assumption — we believe that the same can be proven for the Kurosawa-Desmedt scheme.