Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Master’s Program in Computer Science

Master’s Thesis

On Key-dependent Encryption

submitted by

Matthias Berg

on May 21, 2007

Supervisor and Advisor

Prof. Dr. Michael Backes

Reviewers

Prof. Dr. Michael Backes
Dr. Matteo Maffei
Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Saarbrücken, May 21, 2007

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, May 21, 2007
Acknowledgement

I would like to thank Prof. Dr. Michael Backes for supervising my work and for issuing this interesting topic to me. In long discussions he made it accessible to me and he proofread drafts of this thesis. I also thank Dr. Matteo Maffei for agreeing to review my work. Thanks go to my friends and fellow students whose company has made my years of study so much fun. Finally, I would like to thank my family for the steady support throughout my life and Caroline for being “there”.

iv
Abstract

We tackle the question whether it is safe to encrypt keys and to allow key cycles. If an encryption scheme satisfies the criteria of key-dependent message (KDM) security, it will resist passive attacks in the presence of key cycles. This notion was introduced by Black, Rogaway and Shrimpton, mainly motivated by formal cryptography. So far there is no known encryption scheme which achieves KDM security under standard complexity assumptions. We prove that semantic security is insufficient in the presence of key cycles, even if we only allow cycles of an arbitrary minimum length. We also prove weakened forms of KDM security for two encryption schemes including ElGamal only given standard complexity assumptions such as DDH.
# Table of Contents

1 Introduction ............................................. 1  
2 Preliminaries ........................................... 3  
3 Key-dependent Messages ............................... 9  
   3.1 Defining Key-dependent Message Security ......... 10  
   3.2 An alternative Definition .......................... 12  
4 Key Cycles ............................................. 15  
5 Towards KDM Security .................................. 19  
   5.1 Partially constant Queries ....................... 19  
   5.2 Variations of DDH and the ElGamal Encryption Scheme ........ 21  
6 Conclusion ........................................... 29
Section 1

Introduction

In [6] a new security notion for encryption schemes has been introduced: Key-dependent message (KDM) security. It is strictly stronger than the notion of indistinguishability under chosen-plaintext attacks (IND-CPA) [13], where adversaries can ask for the encryption of self-chosen plaintexts. In the KDM setting this concept is generalized in such a way that adversaries are given the ability to obtain encryptions of messages which depend on the secret decryption keys. In other words it tackles the question whether it is safe to encrypt keys. In chosen-plaintext attacks adversaries cannot ask for encryptions of key-dependent messages, because they do not know the keys needed to construct such messages. In fact we can construct encryption schemes which are IND-CPA-secure but break down if adversaries can query encrypted keys.

The notion of KDM security is mainly motivated by the efforts made to relate the formal and the computational view of cryptography [1]. Using KDM-secure encryption schemes, the soundness of formal encryption can be proven even in the presence of so-called key cycles [2], which is a generalization of encrypted keys. A similar notion called circular security is defined in [8]. There it is used in the construction of a credential system in order to discourage users from sharing their credentials. A more practical application of KDM security is for systems which store encrypted data where the data includes the key.

The ability of adversaries to ask for the encryption of key-dependent messages is modeled by letting them query functions instead of messages. These functions are evaluated with the decryption keys as argument. The result, which can depend on the keys, is used as key-dependent message and its encryption is presented to the adversary. Thereby the adversary obtains the encryption without having direct access to the actual keys.

By this means an adversary can obtain a ciphertext where the contained message depends on the key used to compute the ciphertext. In the KDM setting we deal with multiple keys. Therefore it is also possible to encrypt a message depending on some key $k_1$ by using a different key $k_2$. Then in turn we can let a message dependent on $k_2$ be encrypted using $k_1$, thereby creating a key cycle. We will see that even though there occurs no message dependent on the key used for its encryption, such key cycles can break IND-CPA-secure encryption schemes. This has already been shown in [3] for stateful encryption schemes. We will prove that it also holds true for the stateless case.

In [6] it is shown that KDM security is achievable within the random oracle model [4]. It

\[1\] This holds for the symmetric setting where encryption and decryption keys are equal. In the asymmetric setting the key used to compute the ciphertext can be the corresponding encryption or public key.
remains an open problem whether there are KDM-secure encryption schemes given only standard complexity assumptions. Instead of solving this problem we will weaken the definition of KDM-security so that we can give fulfilling encryption schemes under standard assumptions.

In our first attempt we restrict the set of functions adversaries are allowed to query in such a way that the output of each function must be partially constant. Given an IND-CPA-secure encryption scheme we provide the construction of a scheme which is, in some sense, aware of the functions being queried and prove this scheme to fulfill this weakened form of KDM security.

Our second attempt proves another weakened form of KDM security for the ElGamal encryption scheme [11]. As before we restrict the set of functions allowed in queries. We require them to raise the generator of the group to some kind of linear combination of the secret exponents with the addition that also some quadratic summands are allowed. Note that it is assumed, that no efficient adversary can compute messages where an exponent occurs in squared form [14]. Therefore the functions are not simplistic, since they cannot be computed without knowing the secrets. We use a slightly non-standard definition of encryption schemes with a parameterized key generation. This allows us to define ElGamal so that all keys in a KDM setting work on the same group. This seems reasonable because otherwise the keys would be incompatible. The security proof relies on the assumed computational infeasibility of a problem called multi-DDH which we show to be equivalent to the decisional Diffie Hellman (DDH) problem [10]. The multi-DDH problem generalizes the DDH problem in such a way that multiple tuples are output with the further extension that adversaries can ask for multiple tuples that even contain a common value.

The thesis is structured as follows: In section 2 we give basic definitions such as encryption schemes and IND-CPA security. Section 3 introduces the concept of key-dependent messages and the notion of KDM security. The peril of key cycles is elaborated in section 4. Particularly we prove that even for longer key cycles there are stateless IND-CPA-secure encryption schemes that are not KDM-secure. Section 5 gives the construction of an encryption scheme which fulfills a weakened form of KDM security. It also defines challengers for problems related to DDH. These are used to show a weakened KDM security for the ElGamal encryption scheme. We conclude in section 6.
Section 2

Preliminaries

Cryptography can be summarized with the following words of the cryptographer Ron Rivest: “Cryptography is about communication in the presence of adversaries.” A typical scenario is that we want to prevent adversaries from learning the content of a sent message. This can be done by encrypting the message using a so-called encryption key: The message, called plaintext, is transformed into a so-called ciphertext which is unintelligible for the adversaries, but knowing a certain decryption key allows for reconstructing the message. This process is called decryption.

The requirement that a ciphertext is unintelligible for all possible adversaries is quite strong and hard to fulfill. For example it could happen that a lucky adversary just guesses the correct key, which allows him to decrypt the message. Therefore cryptographers typically weaken such requirements a bit and say that the ciphertext should be unintelligible except for a very small probability, e.g. the event that an adversary guesses the correct key should happen very rarely. We will now show how to express that something only happens with a very small probability.

The key, a bit string, is crucial for the security of encryption. Intuitively using a longer key should provide a higher security. Among other things this concept is captured by the so-called security parameter $k$, which is a natural number. In the following we assume that “everything” (e.g. machines, adversaries, encryption schemes, functions, etc.) is parameterized by $k$, but we will omit it for the sake of readability. In encryption typically the security parameter corresponds to the used key length. Having this parameter we can now express probabilities in terms of $k$. We can say that something happens with a very small probability, if its probability, which is a function of $k$, approaches zero very fast. Namely we require that it approaches zero faster then the reciprocal of any polynomial. We call such functions negligible.

**Definition 2.1 (Negligible Functions)**

A function $f : \mathbb{N} \to \mathbb{R}_0^+$ is negligible iff for all $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that for all $k \geq k_c$ we have $f(k) \leq \frac{1}{k^c}$.

So usually we want that the probability that an adversary guesses the correct key is negligible. If then we choose a sufficiently large $k$, i.e., a large key length, it will (almost) never happen that an adversary guesses the correct key.

But this requirement is still hard to fulfill. A powerful adversary could just try every possible key, until he finds the correct one, and decrypt the message. Therefore such adversaries are excluded by requiring them to be efficient: We only allow adversaries which run in probabilistic polynomial-time. In this setting finding a key is assumed to be difficult.
Definition 2.2 (Efficient Algorithm)

Let $k \in \mathbb{N}$ be the security parameter. An algorithm is called efficient iff it is computable in probabilistic polynomial-time in $k$.

Since we work with probabilistic algorithms we distinguish the following notations for variable assignment. By “$:=”$ we mean the deterministic assignment and by “$←"$ we mean the probabilistic assignment. Furthermore “$\mathcal{R}.$” denotes the uniform random choice from a set. Other notations are $|m|$ for the length of a string $m$ and $0^l$ for the string of $l$ zero-bits. We also assume an error symbol $\downarrow$ as an addition to the domains and ranges of all functions and algorithms.

Before we can discuss security of encryption in more detail, we need to formalize what an encryption scheme is. In the standard definition, encryption schemes consist of three algorithms. One is used to create keys and the other two use such keys to encrypt or decrypt messages, respectively. We will introduce a slightly non-standard definition including a fourth algorithm, which we can ignore for now. We will explain its use later in the text.¹

Definition 2.3 (Parameterized Encryption Scheme)

A parameterized encryption scheme consists of four efficient algorithms $(\text{param}, \text{gen}, E, D)$ with the following properties:

- The randomized parameter generation algorithm $\text{param}$ takes the security parameter in unary representation (for defining complexity in the length of the security parameter) as input and returns a parameter set $p$.

- The randomized key generation algorithm $\text{gen}$ takes the security parameter in unary representation and a parameter set $p$ as input and returns a pair $(pk, sk)$ of keys, where $pk$ is an encryption key and $sk$ is a matching decryption key.

- The encryption algorithm $E$ takes an encryption key $pk$ and a plaintext $m$ and returns a ciphertext $c$ or the error symbol $\downarrow$. This algorithm may be randomized.

- The deterministic decryption algorithm $D$ takes a decryption key $sk$ and a ciphertext $c$ and returns a plaintext $m$ or the error symbol $\downarrow$.

The message space $M_{pk}$ associated to an encryption key $pk$ is the set of plaintexts $m$ for which $E(pk, m)$ never returns $\downarrow$. We require for any parameter set $p$ that might be output by $\text{param}(0^k)$, any key-pair $(pk, sk)$ that might be output by $\text{gen}(0^k, p)$, and for any message $m \in M_{pk}$, if $c ← E(pk, m)$ then $D(sk, c) = m$.

Note that this definition is only about correctness, i.e., if a message is encrypted and then decrypted, the result is the original plaintext again. It does not say anything about confidentiality. In fact a scheme which ignores the keys and whose encryption and decryption algorithms act like the identity function on messages is perfectly valid.

An important detail is that the key generation algorithm returns a pair of keys. One key is used to encrypt messages whereas the other one takes part in the decryption. For many encryption schemes these keys are identical, i.e., the same key is used to encrypt and decrypt encryption messages whereas the other one takes part in the decryption. For many encryption schemes these keys are identical, i.e., the same key is used to encrypt and decrypt

¹We will explain the use of $\text{param}$ and parameter sets in section 5.2. Before that all presented results also hold if we just drop $\text{param}$ and parameter sets, i.e., if we work with the standard definition of encryption schemes.
messages. Such schemes are called symmetric encryption schemes. Schemes whose encryption and decryption keys differ are called asymmetric encryption schemes. If such a scheme fulfills certain security requirements, it is safe to authentically publish the encryption key, which allows everybody to use it to encrypt messages. But since the decryption key is kept secret, nobody except its owner is able to decrypt these messages. Therefore in the asymmetric setting the encryption key and the decryption key are also called public key and secret key, respectively. Clearly this is impossible for symmetric encryption schemes, since publishing the encryption key also gives the decryption key away.

We will define a notion of security which is known as semantic security or indistinguishability under chosen-plaintext attack (IND-CPA) [13]. The goal is to formalize that all efficient adversaries, if given a ciphertext, cannot deduce any significant properties of the contained plaintext (except for its length, which is impossible to hide in general [12]), even if the adversaries can ask for encryptions of arbitrary plaintexts. By significant property we mean something which does not hold for all plaintexts anyway, for example a statement like “the sum of digits is even”, and not something like “it is a string”.

Intuitively an adversary which is able to deduce such a property should be able to distinguish the encryption of a self-chosen plaintext from an encryption of a random message (at least with non-negligible probability). We use this intuition in the definition of IND-CPA-security, which is introduced as an interactive game. We write $A^C$ to denote the output of adversary $A$ interacting with challenger $C$. Here the adversary interacts with a so-called IND-CPA challenger which accepts queries for the encryption of plaintexts. The challenger has a bit $b$ and dependent on this bit it answers the queries correctly, i.e., it outputs an encryption of the queried plaintext, or it outputs a fake encryption, i.e., the output is the encryption of a random string. We call the challenger which outputs correct encryptions the real IND-CPA challenger and the other one the fake IND-CPA challenger. For readability we do not write 0 and 1 for the value of the bit $b$, we use $R$ and $F$ instead. The goal for the adversary is to distinguish whether it interacts with the real or the fake challenger, i.e., after a couple of queries the adversary outputs a bit $b^* \in \{R, F\}$ as the result of its computation. An encryption scheme is called IND-CPA-secure, if no efficient adversary can distinguish the real and the fake IND-CPA challenger significantly better than by pure guessing.

We mentioned earlier that for asymmetric encryption schemes it is possible to publish the encryption key while keeping the decryption key secret. We include this property in the definition by making the challenger output the encryption key in the beginning, if the used encryption scheme is asymmetric.

**Definition 2.4 (IND-CPA Security)**

Given a parameterized encryption scheme $E = (\text{param}, \text{gen}, E, D)$ the IND-CPA challenger for $E$ is defined as follows: It has a bit $b$ initialized as $b \leftarrow \{R, F\}$, a parameter set $p$ initialized as $p \leftarrow \text{param}(0^k)$, and a key-pair $(pk, sk)$ initialized as $(pk, sk) \leftarrow \text{gen}(0^k, p)$. At first it outputs $p$. If $E$ is asymmetric, it also outputs $pk$. Whenever it receives a query $(\text{enc}, m)$ with $m \in M_{pk}$ it does the following:

- If $b = R$, set $c \leftarrow E(pk, m)$ and output $c$.
- If $b = F$, select a random message $m_F \leftarrow \{r : r \in M_{pk} \land |r| = |m|\}$, set $c \leftarrow E(pk, m_F)$ and output $c$.

We write $\text{IND-CPA}_E(b)$ to denote the IND-CPA challenger for $E$ which initialized its bit to $b$. The IND-CPA advantage of an adversary $A$ for $E$ is defined as
\[
\text{Adv}_E^{\text{IND-CPA}}(A) := |\Pr(A^{\text{IND-CPA}_E(R)} = F) - \Pr(A^{\text{IND-CPA}_E(F)} = F)|.
\]

We say that \( E \) is IND-CPA-secure iff \( \text{Adv}_E^{\text{IND-CPA}}(A) \) is negligible for every efficient adversary \( A \).

It is easy to see that no deterministic encryption scheme can be IND-CPA-secure. An adversary could just ask for the encryption of the same message twice and test whether the resulting ciphertexts are equal. If this is the case, then he is interacting with the real challenger with very high probability.

In the asymmetric setting it is clear that IND-CPA security implies that no efficient adversary can deduce a secret key, even if given the matching public key. Otherwise such an adversary could tell the two challengers apart easily.

There are several possibilities to define IND-CPA security. For example one can allow the adversary to obtain the encryptions of some plaintexts and then the adversary chooses two messages. One of them is encrypted and the adversary has to tell which one it was. This is equivalent to our definition. Later in the text we will encounter security definitions where the challenger holds not one but multiple key-pairs. In order to ease the comparison of these definitions and IND-CPA security, we will define multi-IND-CPA security which extends the previous definition in such a way that the challenger holds multiple key-pairs. Nevertheless IND-CPA and multi-IND-CPA security are equivalent.

**Definition 2.5 (Multi-IND-CPA Security)**

Given a parameterized encryption scheme \( E = (\text{param}, \text{gen}, E, D) \) the multi-IND-CPA challenger for \( E \) is defined as follows: It has a bit \( b \) initialized as \( b \in \{R, F\} \), a parameter set \( p \) initialized as \( p \leftarrow \text{param}(0^k) \), and an infinite sequence of key-pairs \( ((pk_i, sk_i))_{i \in \mathbb{N}} \) where each pair, when first used, is initialized as \( (pk_i, sk_i) \leftarrow \text{gen}(0^k, p) \). At first it outputs \( p \). It has the following query types:

- **On input** \((\text{getpk}, i)\): If \( E \) is asymmetric, output \( pk_i \).
- **On input** \((\text{enc}, i, m)\) with \( m \in \mathcal{M}_{pk_i} \):
  - If \( b = R \), set \( c \leftarrow E(pk_i, m) \) and output \( c \).
  - If \( b = F \), select a random message \( m_F \leftarrow \{r : r \in \mathcal{M}_{pk_i} \land |r| = |m|\} \), set \( c \leftarrow E(pk_i, m_F) \) and output \( c \).

We write \( \text{multi-IND-CPA}_E(b) \) to denote the multi-IND-CPA challenger for \( E \) which initialized its bit to \( b \). The multi-IND-CPA advantage of an adversary \( A \) for \( E \) is defined as

\[
\text{Adv}_E^{\text{multi-IND-CPA}}(A) := |\Pr(A^{\text{multi-IND-CPA}_E(R)} = F) - \Pr(A^{\text{multi-IND-CPA}_E(F)} = F)|.
\]

We say that \( E \) is multi-IND-CPA-secure iff \( \text{Adv}_E^{\text{multi-IND-CPA}}(A) \) is negligible for every efficient adversary \( A \).

**Lemma 2.6 (IND-CPA ↔ Multi-IND-CPA)**

A parameterized encryption scheme \( E \) is IND-CPA-secure if and only if it is multi-IND-CPA-secure.

**Proof:** The direction \( \text{multi-IND-CPA} \rightarrow \text{IND-CPA} \) is trivial. An adversary which breaks the IND-CPA security of \( E \) can easily be used to break its multi-IND-CPA security by attacking a single key-pair.
To prove IND-CPA → multi-IND-CPA we use a so-called hybrid argument. The hybrid challenger \( H^\text{e}_R \) is defined like the multi-IND-CPA challenger for \( E \) from Definition 2.5 except that it acts like multi-IND-CPA \( E \) on queries with key numbers \( \leq n \) and like multi-IND-CPA \( E \) on queries with key numbers \( > n \).

Assume we are given an efficient adversary \( A \) for which \( \text{Adv}^{\text{multi-IND-CPA}}_E(A) \) is not negligible. Furthermore we assume that the queries of \( A \) employ at most \( \phi(k) \) key-pairs of the multi-IND-CPA challenger for \( E \). W.l.o.g. we assume the queries of \( A \) do not use key numbers greater than \( \phi(k) \). Therefore \( \phi(k) \) is polynomially bounded. We construct an adversary \( B \) which uses \( A \) as a blackbox in such a way that \( \text{Adv}^{\text{IND-CPA}}_E(B) \) is not negligible.

The input of \( B \) is a parameter set \( p \). If the encryption scheme is asymmetric \( B \) receives also an encryption key \( p_k \). At first \( B \) randomly chooses a value \( n \sim \mathbb{Z}_p \) and generates key-pairs \( (pk_i, sk_i) \leftarrow \text{gen}(0^k, p) \) for all \( i \in \{1, \ldots, \phi(k)\} \setminus \{n\} \). Then it runs \( A \) on input \( p \). Whenever \( B \) receives a query from \( A \) it does the following: If the query contains a key number \( < n \), it behaves like multi-IND-CPA \( E \). If the key number is \( > n \), it behaves like multi-IND-CPA \( E \). If the query is \( (\text{getpk}, n) \) and the encryption scheme is asymmetric, it sends \( pk \) to \( A \). If the query is \( (\text{enc}, n, m) \), it sends the query \( (\text{enc}, m) \) to the IND-CPA challenger and forwards the result to \( A \). If \( A \) outputs its final result \( b^* \), then \( B \) also outputs \( b^* \).

It is easy to see that from the view of \( A \), \( B \) behaves like the hybrid challenger \( H^\text{e}_R \), if the bit of the IND-CPA challenger is \( 0 \). If the bit is \( 1 \), then \( B \) behaves like \( H^\text{e}_R^{-1} \). Note that \( H^\text{e}_R \) behaves like multi-IND-CPA \( E \) and \( H^\text{e}_R^{-1} \) like multi-IND-CPA \( E \). Now we can calculate the IND-CPA advantage of \( B \) for \( E \):

\[
\text{Adv}^{\text{IND-CPA}}_E(B) = |\Pr(B^{\text{IND-CPA}}(R) = F) - \Pr(B^{\text{IND-CPA}}(F) = F)|
\]

\[
= \left| \frac{1}{\phi(k)} \cdot \sum_{n=1}^{\phi(k)} \Pr(A^{H^\text{e}_R}_n = F) - \frac{1}{\phi(k)} \cdot \sum_{n=1}^{\phi(k)} \Pr(A^{H^\text{e}_R^{-1}}_n = F) \right|
\]

\[
= \left| \frac{1}{\phi(k)} \cdot \sum_{n=1}^{\phi(k)} \left( \Pr(A^{H^\text{e}_R}_n = F) - \Pr(A^{H^\text{e}_R^{-1}}_n = F) \right) \right|
\]

\[
= \left| \frac{1}{\phi(k)} \cdot \left( \Pr(A^{\text{multi-IND-CPA}}(R) = F) - \Pr(A^{\text{multi-IND-CPA}}(F) = F) \right) \right|
\]

\[
= \frac{1}{\phi(k)} \cdot \text{Adv}^{\text{multi-IND-CPA}}_E(A)
\]

This is not negligible, if \( \text{Adv}^{\text{multi-IND-CPA}}_E(A) \) is not negligible.

Most cryptographic constructions rely on the existence of so-called one-way functions. Such functions have the property that they are easy to compute, namely in deterministic polynomial-time, but infeasible to invert. Given a one-way function \( f \) and a value \( y := f(x) \) where \( x \) is a random element from the domain of \( f \), no efficient algorithm can find a preimage of \( y \) with non-negligible probability.

**Definition 2.7 (One-Way Function)**

A function \( f : \{0,1\}^* \rightarrow \{0,1\}^* \) is called one-way iff it fulfills the following conditions:

- There exists a deterministic polynomial-time algorithm that on input \( x \) outputs \( f(x) \).
- For every efficient algorithm \( A \) the following holds: Let \( x \sim \mathbb{Z}_p \{0,1\}^k \) and \( x' \leftarrow A(f(x), 0^k) \), then the probability \( \Pr(f(x) = f(x')) \) is negligible.
If additionally $f$ is a permutation, we call it a one-way permutation.

So far it is an open problem whether one-way functions exist. As a matter of fact their existence even implies $P \neq \text{NP}$.\footnote{A non-deterministic machine can easily find a preimage by just guessing one. Therefore, if one-way functions exist, $\text{NP}$ cannot be contained in $\text{BPP}$, the class of decision problems which efficient algorithms can solve with an error probability $\leq \frac{1}{3}$. Since $P \subseteq \text{BPP}$, this implies $P \neq \text{NP}$.} Nevertheless most constructions in modern cryptography build upon the assumption that they exist. In the following we will often assume that we are given an IND-CPA-secure encryption scheme. Given one-way functions, it is possible to construct such schemes [12].
Section 3

Key-dependent Messages

In the last section we introduced indistinguishability under chosen-plaintext attack, where efficient adversaries can ask for encryptions of self-chosen plaintexts without being able to distinguish them from random encryptions. Interestingly this does not imply for all plaintexts that their encryptions look as random encryptions. It just means that this holds for the plaintexts an efficient adversary can come up with. No efficient adversary is able to produce an “insecure” plaintext. A natural candidate for such a plaintext is, of course, the decryption key. While interacting with the IND-CPA challenger, no efficient adversary will ask for the encryption of this key with non-negligible probability, since it does not know it. In fact, given an IND-CPA-secure symmetric encryption scheme $E = (\text{param}, \text{gen}, E, D)$, we can construct an IND-CPA-secure symmetric encryption scheme $E' = (\text{param}, \text{gen}, E', D')$ which breaks down if the adversary somehow comes up with an encryption query for the (decryption) key:

- $E'(sk, m) := \begin{cases} \Omega & \text{if } sk = m \\ E(sk, m) & \text{otherwise} \end{cases}$

Here $\Omega$ is a special ciphertext which is never returned by $E$.

- $D'(sk, c) := \begin{cases} sk & \text{if } c = \Omega \\ D(sk, c) & \text{otherwise} \end{cases}$

An adversary can distinguish the encryption of the key from a random encryption easily by comparing the ciphertext to $\Omega$. The encryption schemes used are symmetric. For the asymmetric case a similar construction is possible, but it is a bit more complicated, since the encryption algorithm cannot check whether the message equals the decryption key. It is only given the encryption key. Later we will see an example for the asymmetric case.

We will now show that $E'$ is still IND-CPA-secure by showing that the IND-CPA advantage is negligible for all efficient adversaries $A$. Let $\omega$ denote the event that at least one of $A$’s queries asks for the encryption of the key. It is easy to see that $\Pr(\omega)$ must be negligible. Otherwise we could use $A$ as a blackbox to obtain the key and use it to break the IND-CPA security of $E$. This works as follows: We let $A$ interact with the IND-CPA challenger of $E$ and pick one of its queries at random. Since at least one of $A$’s queries contains the key with non-negligible probability and it only queries a polynomial number of times ($A$ is efficient), the query we pick will contain the key with non-negligible probability. Having the key it is easy to distinguish the real and the fake challenger. This contradicts the IND-CPA security of $E$. Therefore $\Pr(\omega)$ must be negligible. Now we can calculate the IND-CPA advantage of $A$ for $E'$:
3.1. Defining Key-dependent Message Security

$$\text{Adv}_{\mathcal{E}'}^{\text{IND-CPA}}(A) = |\Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(F) = F}) - \Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(R) = F})|$$

$$= |\Pr(\omega) \cdot (\Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(R) = F|\omega}) - \Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(F) = F|\omega})) + \Pr(\omega) \cdot (\Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(R) = F|\omega}) - \Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(F) = F|\omega}))|$$

$$\leq \Pr(\omega) + |\Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(R) = F|\omega}) - \Pr(A^{\text{IND-CPA}_{\mathcal{E}'}(F) = F|\omega})|$$

The negligibility of the latter term follows from the IND-CPA security of $\mathcal{E}$. The third equality holds, because under the condition $\omega$ the schemes $\mathcal{E}$ and $\mathcal{E}'$ behave equivalently.

The conclusion from this is that one should think twice before encrypting keys, even if the used encryption scheme fulfills standard security notions. Most protocol designers avoid this trouble by excluding the possibility of encrypted keys. But sometimes encrypted keys occur. In [8] it is used to establish an all-or-nothing property in a credential system to prevent users from sharing their credentials. Here the encryption scheme is required to be circular secure, meaning that it is safe to encrypt keys with this scheme. Another example where encrypted keys can occur are systems that store their data in encrypted form (e.g. as a backup) where the stored data include the key.

If one does not want to be bothered by the low-level details about how an encryption scheme works, such as probabilities and complexity issues, it is convenient to look at encryption from a high-level formal perspective. Here encryption does not operate on strings, but on terms. Encryption is just a symbol, and its behavior is modeled by a set of rules that determine how to manipulate it. In [1] it is shown how to relate the high-level formal view and the low-level computational view. It shows that if two expressions are equivalent in the formal world, their corresponding strings in the computational world are indistinguishable. Nevertheless this comes with a limitation. The expressions have to be acyclic, meaning they must not use a key to encrypt messages which contain this key. In [2] this correspondence is shown even for cyclic expressions, if the used encryption scheme satisfies the notion of key-dependent message (KDM) security [6]. This notion only concerns passive adversaries. In [3] the correspondence is shown even for active attacks and dynamic revelations of keys using extended security notions.

3.1 Defining Key-dependent Message Security

We will now introduce a definition of KDM security along the lines of [6, 3]. This notion is a generalization of IND-CPA security in two ways. First of all the challenger does not only work with a single key-pair, but with a vector of key-pairs. This is done since a single key-pair seems to entail a loss of generality. Secondly, since not only decryption keys can be unsafe to encrypt, but also messages that depend on them, the adversary sends functions instead of plaintexts in its encryption queries. The challenger evaluates these functions with the decryption key vector as argument. The result is encrypted with one of the encryption keys and sent back to the adversary. The ability to send functions gives the adversary indirect access to the keys. This way it can ask for the encryption of a key or a key-dependent message, even if the key is unknown to it. This is not possible in the IND-CPA setting.

Since we only consider efficient adversaries, it makes sense to restrict the set of functions an adversary may query accordingly. We require the functions to be computable in polynomial
3. Key-dependent Messages

time in \( k \). Furthermore they must have a polynomial size representation, since the adversary needs to output them somehow. Note that we use functions in encryption queries and not probabilistic algorithms. This is not a loss of generality, because an adversary can provide the functions with sufficient “hard-coded” randomness in advance. The adversary also decides which encryption key is used to encrypt the result, so we can require that the functions evaluate to a plaintext which is in the message space of this key. We mentioned earlier that it is impossible for encryption to hide the length of the plaintext. We circumvent this problem by requiring that the output of the functions is of fixed size for a fixed security parameter. This ensures that the adversary learns nothing from seeing the size of the messages. We call functions which satisfy these conditions permitted functions.

**Definition 3.1 (Permitted Functions)**

Given a parameterized encryption scheme \( \mathcal{E} \) and an encryption key \( \mathbf{pk} \) of \( \mathcal{E} \), a function \( f \) is called a permitted function with respect to \( \mathbf{pk} \) iff it fulfills the following conditions:

- \( f \) is computable in polynomial-time and has a polynomial size representation.
- \( f \) maps infinite sequences of decryption keys \( \mathbf{sk} \) of \( \mathcal{E} \) to plaintexts in \( \mathcal{M}_{\mathbf{pk}} \).
- For a fixed security parameter the output of \( f \) is of fixed size. We denote this size by \( |f| \).

We write \( \mathcal{F}_{\mathbf{pk}} \) to denote the set of all permitted functions with respect to \( \mathbf{pk} \).

Even though there is a difference between a function and its representation, in the following we will use them interchangeably and leave the classification to the reader. We define key-dependent message security as follows:

**Definition 3.2 (Key-dependent Message (KDM) Security)**

Given a parameterized encryption scheme \( \mathcal{E} = (\text{param}, \text{gen}, \mathcal{E}, \mathcal{D}) \) the key-dependent message challenger or KDM challenger for \( \mathcal{E} \) is defined as follows: It has a bit \( b \) which is initialized as \( b \leftarrow \{R, F\} \), a parameter set \( p \) initialized as \( p \leftarrow \text{param}(0^k) \), and an infinite sequence of key-pairs \( ((\mathbf{pk}_i, \mathbf{sk}_i))_{i \in \mathbb{N}} \) where each pair, when first used, is initialized as \( (\mathbf{pk}_i, \mathbf{sk}_i) \leftarrow \text{gen}(0^k, p) \). At first it outputs \( p \). It has the following query types:

- **On input \( \text{getpk}, i \):** If \( \mathcal{E} \) is asymmetric, output \( \mathbf{pk}_i \).
- **On input \( \text{enc}, i, f \):** Let \( m_R := f(\mathbf{sk}) \) and \( m_F \leftarrow \{ m : m \in \mathcal{M}_{\mathbf{pk}_i} \land |m| = |m_R| \} \), encrypt \( c \leftarrow \mathcal{E}(\mathbf{pk}_i, m_b) \) and output \( c \).

We write \( \text{KDM}_\mathcal{E}(b) \) to denote the KDM challenger for \( \mathcal{E} \) which initialized its bit to \( b \). The KDM advantage of an adversary \( A \) for \( \mathcal{E} \) is defined as

\[
\operatorname{Adv}_{\mathcal{E}}^{\text{KDM}}(A) := \Pr(A^{\text{KDM}_\mathcal{E}(R)} = F) - \Pr(A^{\text{KDM}_\mathcal{E}(F)} = F).
\]

We say that \( \mathcal{E} \) is KDM-secure iff \( \operatorname{Adv}_{\mathcal{E}}^{\text{KDM}}(A) \) is negligible for every efficient adversary \( A \) whose encryption queries \( \text{enc}, i, f \) fulfill the condition \( f \in \mathcal{F}_{\mathbf{pk}_i} \).
3.2 An alternative Definition

In [6] KDM security is defined slightly differently. There the fake challenger does not answer with an encryption of a randomly chosen message, but with the encryption of a constant string. Namely queries of the form \((\text{enc}, i, f)\) are answered with \(c \leftarrow E(pk_i, 0^{|s(k)|})\). We will call this definition const-KDM. It assumes that \(0^{|s(k)|}\) is a valid plaintext, but not all encryption schemes have this property. To circumvent this one could use a different constant message or use a publicly known function which selects the message to be encrypted. In the following we will show that KDM security and const-KDM security are equivalent (assuming \(0^{|s(k)|}\) is a valid plaintext).

**Lemma 3.3 (KDM ↔ const-KDM)**

A parameterized encryption scheme \(E\) is KDM secure if and only if it is const-KDM secure.

**Proof:** 1) We first show const-KDM \(\rightarrow\) KDM.

Given an efficient adversary \(A\) against the KDM challenger for \(E\) we construct an adversary \(B\) against the const-KDM challenger for \(E\) which uses \(A\) as a blackbox:

At first \(B\) chooses a bit \(b' \gets \{R, F\}\) and runs \(A\) on its own input \(p\). Queries of the form \((\text{getpk}, i)\) from \(A\) are forwarded to the const-KDM challenger and the result is given to \(A\). On each query \((\text{enc}, i, f)\) from \(A\) the following actions are performed:

- If \(b' = R\), forward \((\text{enc}, i, f)\) to the const-KDM challenger.
- If \(b' = F\), send \((\text{enc}, i, r)\) to the const-KDM challenger, where \(r\) is a function which selects a message uniformly at random out of \(\{m : m \in \mathcal{M}_{pk}, |m| = |f|\}\). Note that even though \(r\) is deterministic \(B\) can provide it with sufficient hard-coded randomness beforehand.
- The resulting ciphertext \(c\) from the const-KDM challenger is forwarded to \(A\).

When \(A\) finally outputs its result \(b^*\), \(B\) outputs \(R\) if \(b^* = b'\) and \(F\) if \(b^* \neq b'\). This setup is outlined in figure 3.1.

We will now calculate the const-KDM advantage of \(B\). If the bit \(b\) of the const-KDM challenger is \(F\), the ciphertext \(c\) does not leak any information about \(b'\) to \(A\). This gives us the following probability: \(\Pr(B^{\text{const-KDM}}_E(F) = F) = \frac{1}{2}\). If \(b = R\), \(B\) acts like \(\text{KDM}_E(b')\) in the view of \(A\). So we can calculate

\[
\Pr(B^{\text{const-KDM}}_E(R) = F) = \frac{1}{2} \cdot \Pr(B^{\text{const-KDM}}_E(R) = F | b' = R) + \frac{1}{2} \cdot \Pr(B^{\text{const-KDM}}_E(R) = F | b' = F) = \frac{1}{2} \cdot \Pr(A^{\text{KDM}}_E(R) = F) + \frac{1}{2} \cdot \Pr(A^{\text{KDM}}_E(F) = R) = \frac{1}{2} \cdot (\Pr(A^{\text{KDM}}_E(R) = F) - \Pr(A^{\text{KDM}}_E(F) = F)) + \frac{1}{2}.
\]

Therefore the const-KDM advantage of \(B\) is

\[
\text{Adv}^{\text{const-KDM}}_B(E) = |\Pr(B^{\text{const-KDM}}_E(R) = F) - \Pr(B^{\text{const-KDM}}_E(F) = F)| = \frac{1}{2} \cdot (\Pr(A^{\text{KDM}}_E(R) = F) - \Pr(A^{\text{KDM}}_E(F) = F)) + \frac{1}{2} - \frac{1}{2} = \frac{1}{2} \cdot \text{Adv}^{\text{KDM}}_A(E).
\]


2) We now show $\text{KDM} \rightarrow \text{const-KDM}$.

Analogously we construct an adversary $B$ against the KDM challenger for $\mathcal{E}$ which uses a given efficient adversary $A$ against the const-KDM challenger for $\mathcal{E}$ as a blackbox:

At first $B$ chooses a bit $b' \overset{\$}{\leftarrow} \{R, F\}$ and runs $A$ on its own input $p$. Queries of the form $(\text{getpk}, i)$ from $A$ are forwarded to the KDM challenger and the result is given to $A$. On each query $(\text{enc}, i, f)$ from $A$ the following actions are performed:

- If $b' = R$, forward $(\text{enc}, i, f)$ to the KDM challenger.
- If $b' = F$, send $(\text{enc}, i, c)$ to the KDM challenger, where $c$ is the constant function which returns $0_{|f|}$ for every argument.
- The resulting ciphertext $c$ from the KDM challenger is forwarded to $A$.

When $A$ finally outputs its result $b^*$, $B$ outputs $R$ if $b^* = b'$ and $F$ if $b^* \neq b'$. This setup is outlined in figure 3.2.

We will now calculate the KDM advantage of $B$. If the bit $b$ of the KDM challenger is $F$, the ciphertext $c$ does not leak any information about $b'$ to $A$. This gives us the following probability: $\Pr(B^{\text{KDM}_E(F)} = F) = \frac{1}{2}$. If $b = R$, $B$ acts like $\text{const-KDM}_E(b')$ in the view of $A$. So we can calculate

$$\Pr(B^{\text{KDM}_E(R)} = F) = \frac{1}{2} \cdot \Pr(A^{\text{const-KDM}_E(R)} = F | b' = R) + \frac{1}{2} \cdot \Pr(A^{\text{const-KDM}_E(R)} = F | b' = F)$$

$$= \frac{1}{2} \cdot \left( \Pr(A^{\text{const-KDM}_E(R)} = F) - \Pr(A^{\text{const-KDM}_E(F)} = F) \right) + \frac{1}{2}.$$ 

Therefore the KDM advantage of $B$ is

$$\text{Adv}_E^{\text{KDM}}(B) = \left| \Pr(B^{\text{KDM}_E(R)} = F) - \Pr(B^{\text{KDM}_E(F)} = F) \right|$$

$$= \left| \frac{1}{2} \cdot \left( \Pr(A^{\text{const-KDM}_E(R)} = F) - \Pr(A^{\text{const-KDM}_E(F)} = F) \right) + \frac{1}{2} - \frac{1}{2} \right|$$

$$= \frac{1}{2} \cdot \text{Adv}_E^{\text{const-KDM}}(A).$$

□
3.2. An alternative Definition

\[
\begin{align*}
\text{const-KDM challenger} & \quad \mathcal{R} \quad \{R, F\} \\
\text{B against const-KDM challenger} & \quad \mathcal{R} \quad \{R, F\} \\
\text{A against KDM challenger} & \quad \mathcal{R} \quad \{R, F\}
\end{align*}
\]

Figure 3.1: const-KDM-security implies KDM-security

\[
\begin{align*}
\text{KDM challenger} & \quad \mathcal{R} \quad \{R, F\} \\
\text{B against KDM challenger} & \quad \mathcal{R} \quad \{R, F\} \\
\text{A against const-KDM challenger} & \quad \mathcal{R} \quad \{R, F\}
\end{align*}
\]

Figure 3.2: KDM-security implies const-KDM-security
One difference between IND-CPA security and KDM security is that the KDM challenger holds multiple keys. In particular it is not only possible to ask that a key be encrypted with itself, but also to request that one key should be encrypted using another key. At first thought encrypting one key with another might not entail a problem similar to the problem above, since the keys are chosen independently and the adversary could have chosen the to-be-encrypted key by itself. Anyway this does not hold, if the adversary lets the first key be encrypted with the second key and later on asks to encrypt a message depending on the second key using the first key. This way the message in the second query may contain some information about the first key, which can cause non-KDM-secure encryptions schemes to break down. The problem here is, that even if the adversary does not ask for encryptions of messages which depend on the used key, the messages can still contain unsafe information about this key using such a key cycle as outlined above. To formalize what a key cycle is, we define the so-called key-dependency graph which contains an edge \((i, j)\), if the adversary asks for the encryption of a message dependent on the \(i\)-th key using the \(j\)-th key.

**Definition 4.1 (Key-dependency Graph)**

For a permitted function \(f\) and a key number \(i \in \mathbb{N}\) we write \(i \in f\) iff the result of \(f\) depends on the \(i\)-th decryption key, i.e., there exist two sequences of decryption keys \(\vec{sk}\) and \(\vec{sk}'\) with \(sk_i \neq sk'_i\) and \(sk_j = sk'_j\) for all \(j \neq i\) such that \(f(\vec{sk}) \neq f(\vec{sk}')\). Given a history \(h\) of calls to the KDM challenger, we define the key-dependency graph \(G_h\): It contains the edge \((i, j)\) iff \(h\) contains a call \((\text{enc}, j, f)\) with \(i \in f\).

A key cycle is just a cycle in the key-dependency graph. To address the issue that also key cycles of length \(\geq 2\) can be unsafe, we define a notion called \(l\)-KDM Security [3] which equals KDM security with the restriction that adversaries must not produce key cycles of length less than \(l\).

**Definition 4.2 (\(l\)-KDM Security)**

Given a parameterized encryption scheme, \(l\)-KDM security is defined like KDM security except that adversaries are restricted to produce no key cycles of length less than \(l\), i.e., for every possible history \(h\) of calls to the KDM challenger, the key-dependency graph \(G_h\) must not contain cycles of length less than \(l\).

In [3] it is shown that for all \(l \in \mathbb{N}\) there exist stateful encryption schemes that are IND-CPA-secure but not \(l\)-KDM-secure. The proof uses a scheme that is stateful in the manner...
that its encryption algorithm behaves differently depending on whether an encryption key is used for the first time or not. It has been an open problem if this also holds for stateless encryption schemes, i.e., schemes according to our definition. We will prove that there are indeed stateless encryption schemes that separate IND-CPA security from \( l \)-KDM security for arbitrary \( l \). The proof uses the insight that the decryption algorithm basically implements a permitted function which an adversary can use in queries. At first we will prove a lemma that states that if a scheme is not \( l \)-KDM-secure, then it is also insecure for even longer key cycles.

**Lemma 4.3 \((\neg l\text{-KDM} \rightarrow \neg 2l\text{-KDM})\)**

*Given a parameterized encryption scheme \( E \) whose message spaces \( M_{\text{pk}} \) are identical for all encryption keys \( \text{pk} \) generated with the same security parameter and parameter set, the following holds for all \( l \in \mathbb{N} \): If \( E \) is not \( l \)-KDM-secure, then it is also not \( 2l \)-KDM-secure.*

**Proof:** Given an adversary \( A \) against the KDM challenger which never produces key cycles of length less than \( l \), we construct an adversary \( B \) which uses \( A \) as a blackbox and never produces key cycles of length less than \( 2l \). W.l.o.g. we assume there is a polynomially bounded key number \( \mu \) such that for all \( i \geq \mu \) and for all possible histories \( h \) of calls by \( A \) to the KDM challenger \( i \) is never part of the key-dependency graph \( G_h \). Intuitively this means that \( A \) does not touch the secret keys from \( \text{sk}_h \) on.

Given input \( p \) adversary \( B \) runs \( A \) on input \( p \). When \( B \) receives a query of the form \((\text{getpk}, i)\) from \( A \), it forwards it to the KDM challenger and presents the answer to \( A \) (if there is one). If the query is of the form \((\text{enc}, i, f)\) it sends \((\text{enc}, i + \mu, f)\) to the KDM challenger and receives the answer \( c \). It then sends \((\text{enc}, i, D_{i+\mu}^c)\) to the KDM challenger, \( D_{i+\mu}^c \) being the function which decrypts \( c \) using the decryption key \( \text{sk}_{i+\mu} \). The result is forwarded to \( A \). When \( A \) outputs its final bit \( b \), \( B \) also outputs \( b \). It is easy to see that \( B \) gives \( A \) a perfect simulation of the behaviour of the KDM challenger. Therefore we have \( \text{Adv}^\text{KDM}_E(B) = \text{Adv}^\text{KDM}_E(A) \).

If a query \((\text{enc}, i, f)\) of \( A \) would produce an edge \((j, i)\), then the corresponding queries \((\text{enc}, i + \mu, f)\) and \((\text{enc}, i, D_{i+\mu}^c)\) produce the edges \((j, i + \mu)\) and \((i + \mu, i)\). This happens for all edges which \( A \) would produce. Therefore every cycle in the graph produced by \( A \) corresponds to a cycle of double length in the graph produced by \( B \). This is illustrated in figure 4.1 for \( \mu = 10 \). Since \( A \) does not touch the decryption keys from \( \text{sk}_h \) on, \( B \) does not produce any additional cycles. Therefore \( E \) is not \( 2l \)-KDM-secure, if it is not \( l \)-KDM-secure.

The lemma only talks about schemes whose message spaces are identical. This is because we must be able to encrypt the result of a function with a key which the adversary does not intend to use. This restriction can be relaxed to message spaces that can be transformed into each other using polynomial-time computable bijections. The same holds for the below statements. Using this lemma it is easy to see, that KDM security and \( l \)-KDM security are actually equivalent. This is expressed in the following corollary.

![Figure 4.1: Example of a key-dependency graph and its corresponding graph with key cycles of double length for \( \mu = 10 \).](image)
4. Key Cycles

Corollary 4.4 (KDM $\leftrightarrow$ $l$-KDM)

Given a parameterized encryption scheme $E$ whose message spaces $M_{pk}$ are identical for all encryption keys $pk$ generated with the same security parameter and parameter set, the following holds for all $l \in \mathbb{N}$: $E$ is KDM-secure if and only if it is $l$-KDM-secure.

Proof: The direction KDM $\rightarrow$ $l$-KDM follows trivially from their definitions. 
$\neg$ KDM $\rightarrow$ $\neg l$-KDM follows from Lemma 4.3 by applying it $\left\lceil \log(l) \right\rceil$ times. Note that KDM security is equivalent to 1-KDM security by definition. □

Our goal is to give an example which separates IND-CPA security and $l$-KDM security. Using Corollary 4.4 we only need to separate IND-CPA security and KDM security. We already did this above for symmetric encryption. In the proof of the following theorem we provide a separating example which works for the symmetric and the asymmetric case.

Theorem 4.5 (IND-CPA $\not\rightarrow$ $l$-KDM)

If IND-CPA-secure parameterized encryption schemes exist whose message spaces $M_{pk}$ are identical for all encryption keys $pk$ generated with the same security parameter and parameter set, and if one-way permutations on these message spaces exist, then for all $l \in \mathbb{N}$ there exist IND-CPA-secure parameterized encryption schemes which are not $l$-KDM-secure.

Proof: Let $E = (\text{param, gen, E, D})$ be such an IND-CPA-secure parameterized encryption scheme. We will use $E$ to construct the following parameterized encryption scheme $E' = (\text{param, gen', E', D'})$ which is still IND-CPA-secure, but not KDM-secure (and therefore not $l$-KDM-secure using Corollary 4.4):

- The key generation algorithm $\text{gen'}$ for security parameter $k$ and parameter set $p$ computes $(pk, sk) \leftarrow \text{gen}(0^k, p)$ and randomly chooses a message $x \leftarrow M_{pk}$. Let $\sigma$ be a one-way permutation on $M_{pk}$. It then returns $(pk', sk')$, where $pk' := (pk, \sigma, \sigma(x))$ and $sk' := (sk, x)$.

- The encryption algorithm $E'$ for an encryption key $pk' = (pk, \sigma, y)$ and a message $m$ is defined as follows:
  
  
  $E'(pk', m) := \begin{cases} 
  \Omega & \text{if } \sigma(m) = y \\
  E(pk, m) & \text{otherwise} 
  \end{cases}$

  Here $\Omega$ is a special ciphertext which is never returned by $E$.

- The decryption algorithm $D'$ for a decryption key $sk' = (sk, x)$ and a ciphertext $c$ is defined as follows:

  $D'(sk', c) := \begin{cases} 
  x & \text{if } c = \Omega \\
  D(sk, c) & \text{otherwise} 
  \end{cases}$

  The scheme $E'$ is not KDM-secure. An adversary can just ask for the encryption of $x$, since the function he sends has access to it. I.e. he sends something like $(\text{enc}, 1, \pi)$, where $\pi$ is the projection on the $x$-part of the first key. By comparing the result to $\Omega$ he can gain an advantage close to 1.

  To see that $E'$ is IND-CPA-secure, we calculate the IND-CPA advantage of a given adversary $A$ for $E'$. Let $X$ denote the event that $A$ asks for the encryption of an $x$-part of a decryption key. This only happens with a negligible probability, because it means that $A$ inverted a one-way permutation.

17
\[
\text{Adv}_{E'}^{\text{IND-CPA}}(A) = |\Pr(A^{\text{IND-CPA}}(R) = F) - \Pr(A^{\text{IND-CPA}}(F) = F)| \\
= |\Pr(X) (\Pr(A^{\text{IND-CPA}}(R) = F|X) - \Pr(A^{\text{IND-CPA}}(F) = F|X)) \\
+ \Pr(\overline{X}) (\Pr(A^{\text{IND-CPA}}(R) = F|\overline{X}) - \Pr(A^{\text{IND-CPA}}(F) = F|\overline{X}))| \\
= |\Pr(X) (\Pr(A^{\text{IND-CPA}}(R) = F|X) - \Pr(A^{\text{IND-CPA}}(F) = F|X)) \\
+ \Pr(\overline{X}) (\Pr(A^{\text{IND-CPA}}(R) = F|\overline{X}) - \Pr(A^{\text{IND-CPA}}(F) = F|\overline{X}))| \\
\leq \Pr(X) + |\Pr(A^{\text{IND-CPA}}(R) = F|X) - \Pr(A^{\text{IND-CPA}}(F) = F|X)|
\]

The negligibility of the latter term follows from the IND-CPA security of \( E \). The third equality holds, because under the condition \( X \) the schemes \( E \) and \( E' \) behave equivalently. □
Section 5

Towards KDM Security

The last section was about separating IND-CPA and KDM security. We explicitly constructed an encryption scheme that is not KDM-secure. Now we will try the opposite and aim at a provably KDM-secure encryption scheme. In [6] it has been shown that KDM security is achievable within the random oracle model [4], but it is an open problem whether KDM security can be achieved under standard complexity assumptions without random oracles. So far we have not solved this problem, but if we weaken the definition of KDM security, we can give constructions which fulfill it. We will accomplish such weakenings by restricting the set of permitted functions which adversaries are allowed to use in queries.

5.1 Partially constant Queries

A radical approach is to allow only constant functions in queries. If we weaken KDM security this way, it is equivalent to IND-CPA security. Querying constant functions and querying messages is basically the same. So this weakened form of KDM security can be achieved trivially, if an IND-CPA-secure encryption scheme is given. In the following we will investigate a more interesting approach. We do not require the functions to be constant, but they must be partially constant. We assume that our message spaces consist of pairs (given a reasonable encoding of pairs as bit strings) and only allow permitted functions which are constant in the first component of its output. In this setting we can construct a weakly KDM-secure encryption scheme.

The key idea is the following: If the encryption algorithm knows the function which was used to compute a message, it does not need to encrypt the message at all. It suffices to encrypt the function’s representation. The decryption algorithm can simply recompute the message by evaluating the function. This holds with some limitation. The decryption algorithm can only evaluate functions which depend on a single key, namely its decryption key. The other keys are unknown to the algorithm. In a symmetric setting we can circumvent this problem, since the encryption algorithm uses the same key as the decryption algorithm. It can check whether this key suffices to evaluate the function. If the check fails, it just rejects and thereby excludes such functions from the set of permitted functions, since they do not map into the message space. If the check succeeds, it just encrypts the function and not the message itself. Note that an adversary against the KDM challenger of such a scheme basically plays an IND-CPA game. It queries functions and obtains their encryptions instead of the encryptions of their results.
5.1. Partially constant Queries

Unfortunately the encryption algorithm does not know the function which was used to compute a message. Therefore we just let the encryption algorithm act as it was given the function. As mentioned above our message spaces consist of pairs. The encryption algorithm just uses the first component of these pairs as the queried function and the second part as the actual message. To exclude malformed pairs it checks whether the first component represents a function (using a reasonable encoding) and whether this function, if given the key, evaluates to the second component. This forces adversaries to query functions which produce pairs \((h, m)\) fulfilling \(h(\vec{sk}) = m\). Even though the actually queried function remains unknown to the encryption algorithm, its result looks like a queried function with its corresponding message. A technical subtlety is that the encryption algorithm still needs to run in polynomial-time. Therefore we impose a polynomial upper bound on the time needed to evaluate the function \(h\).

Definition 5.1 (Query-aware Encryption Scheme)
Let \(\mathcal{E} = (\text{param, gen, E, D})\) be an IND-CPA-secure symmetric parameterized encryption scheme and let \(t\) be some polynomial. The parameterized encryption scheme \(\mathcal{E}_Q = (\text{param, gen, E}_Q, D_Q)\) is defined as follows:

- Encryption \(E_Q(\vec{sk}, (h, m))\) for a key \(\vec{sk}\) and a message \((h, m)\) is computed as follows: If \(h\) represents a function and \(h(\vec{sk})\) evaluates to \(m\) with runtime \(\leq t(k)\), encrypt \(c \leftarrow E(\vec{sk}, h)\) and output \(c\). Otherwise output \(\downarrow\).

- Decryption \(D_Q(\vec{sk}, c)\) for a key \(\vec{sk}\) and a ciphertext \(c\) computes \(h := D(\vec{sk}, c)\) and outputs \((h, h(\vec{sk}))\).

While encrypting a pair \((h, m)\), just \(h\) is needed in the actual encryption operation. If we can extract \(h\) from a query \((\text{enc}, i, f)\) without knowing \(\vec{sk}\), then we can simulate the KDM challenger for \(\mathcal{E}_Q\) while attacking the multi-IND-CPA challenger for \(\mathcal{E}\). Therefore we restrict the set of permitted functions to partially constant functions. They compute pairs \((h, m)\) where \(h\) is independent of \(\vec{sk}\), i.e., \(h\) can be computed even if \(\vec{sk}\) is unknown.

Theorem 5.2 (KDM Security for partially constant Queries)
If IND-CPA-secure symmetric parameterized encryption schemes exist, then there exists a symmetric parameterized encryption scheme for which every efficient adversary \(A\) has a negligible KDM advantage, provided that the following holds for all encryption queries \((\text{enc}, i, f)\):

- \(f \in \mathcal{F}_{\vec{sk}_i}\), where \(\vec{sk}_i\) is the \(i\)-th key of the KDM challenger.

- There exists a message \(m\) such that for all sequences of keys \(\vec{sk}\) there exists a message \(m\) such that \(f(\vec{sk}) = (h, m)\).

Proof: Let \(\mathcal{E}\) be an IND-CPA-secure symmetric parameterized encryption scheme. We will show that the symmetric parameterized encryption scheme \(\mathcal{E}_Q\) as constructed in Definition 5.1 has the required properties. Assume we are given an efficient adversary \(A\) with a non-negligible KDM advantage for \(\mathcal{E}_Q\) which fulfills the requirements on queries of Theorem 5.2. We construct an adversary \(B\) against the multi-IND-CPA challenger for \(\mathcal{E}\) by using \(A\) as a blackbox.

Given input \(p\) adversary \(B\) simply runs \(A\) on input \(p\). Whenever \(A\) queries \((\text{enc}, i, f)\), \(B\) applies \(f\) to some key sequence \(\vec{sk}\) obtaining \((h, m)\). It then queries the multi-IND-CPA challenger for \(\mathcal{E}\) on \((\text{enc}, i, h)\) and forwards the result to \(A\). The final output \(b^*\) of \(A\) is also the final output of \(B\).
5. Towards KDM Security

Since the function $f$ is constant in its first component, $B$ gives $A$ a perfect simulation of the KDM challenger for $E_{Q}$ that holds a bit equal to the bit of the multi-IND-CPA challenger for $E$.

$$
\text{Adv}^{\text{multi-IND-CPA}}_{E}(B) = \left| \Pr(B^{\text{multi-IND-CPA}_{E}(R)} = F) - \Pr(B^{\text{multi-IND-CPA}_{E}(F)} = F) \right|
$$

$$
= \left| \Pr(A^{KDM_{E,Q}(R)} = F) - \Pr(A^{KDM_{E,Q}(F)} = F) \right|
$$

$$
= \text{Adv}^{\text{KDM}}_{E,Q}(A)
$$

The restriction that all queries must be partially constant can be expressed differently. One approach is to only allow efficient adversaries $A$ that query functions for which they “know” the first part of their output, i.e., for which there exists an efficient algorithm that, given the current state of $A$, can extract the first component $h$ of the message resulting from $A$’s query. The intuition is that $A$ should be aware of the $h$, which is going to be encrypted. This idea is related to plaintext-awareness [5], a notion of security for asymmetric encryption schemes, where an adversary can only produce ciphertexts for which it knows the contained plaintext.

5.2 Variations of DDH and the ElGamal Encryption Scheme

In section 2 we mentioned that most constructions in modern cryptography rely on the existence of one-way functions. So far nobody has succeeded in proving a function to be one-way, but there are several candidates which are conjectured one-way. Exponentiation in cyclic groups is such a candidate. In the following let $p$ and $q$ be primes such that $q$ divides $p - 1$ and $|p|$ is polynomially in $|q|$, i.e., there is a polynomial $k_p$ such that $|p| = k_p(|q|)$. We will use the cyclic group $G_q$ which is a subgroup of order $q$ of $\mathbb{Z}_p^*$. Given a generator $g$ of $G_q$ and an exponent $x \in \{1, \ldots, q\}$, computing $g^x \text{ mod } p$ is easy (polynomial-time in $|q|$), but the inverse operation is conjectured to be difficult. Given a generator $g$ and a group element $y$, there is no known efficient algorithm which computes the so-called discrete logarithm. This is the exponent $x \in \{1, \ldots, q\}$ for which $g^x = y$ (modulo $p$).

Related to discrete logarithms is the so-called Diffie-Hellman problem: Given a generator $g$ and two elements $g^x$ and $g^y$ for some $x, y \in \{1, \ldots, q\}$, finding the value of $g^{xy}$ is conjectured to be difficult [10]. In the group $G_q$ it is even assumed that no efficient adversary can distinguish the values $g^{xy}$ and $g^r$ significantly better than by pure guessing for some random exponent $r$. The task to decide whether one is given $g^{xy}$ or $g^r$ is called the decisional Diffie-Hellman (DDH) problem. The following challenger presents, dependent on a bit $b$, a tuple containing $g^{xy}$ or $g^r$. The goal for adversaries interacting with this challenger is to determine the value of the bit $b$.

**Definition 5.3 (DDH Challenger)**

The DDH challenger (for subgroups $G_q$ of $\mathbb{Z}_p^*$) for a given security parameter $k$ is defined as follows: It randomly chooses a $k$-bit prime $q$, a $k_p(k)$-bit prime $p$ such that $|q|p - 1$, an element $g \in \mathbb{Z}_p^*$ of order $q$, a bit $b \in \{R, F\}$, and values $x, y, z \in \{1, \ldots, q\}$. Depending on the value of $b$, it outputs

- $(q, p, g, g^x, g^y, g^{xy})$ if $b = R$
- $(q, p, g, g^x, g^y, g^z)$ if $b = F$.

A typical choice is to set $k_p(k) := k + 1$ and to look for a prime $q$ such that $p := 2q + 1$ is a prime as well.
5.2. Variations of DDH and the ElGamal Encryption Scheme

We write DDH(b) to denote the DDH challenger which initialized its bit to b. The DDH advantage of an adversary A is defined as:

\[
\text{Adv}^{\text{DDH}}(A) := | \Pr(A^{\text{DDH}(R)} = F) - \Pr(A^{\text{DDH}(F)} = F) |.
\]

Assumption 5.4 (Hardness of the DDH Problem in \(G_q\))

It is conjectured that for every efficient adversary A the DDH advantage \(\text{Adv}^{\text{DDH}}(A)\) as constructed in Definition 5.3 is negligible.

This is a standard assumption in cryptography and there are encryption schemes which base their security on it [11, 9]. The asymmetric ElGamal encryption scheme [11] works as follows: Key generation chooses primes \(q\) and \(p\), a generator \(g\) and an exponent \(x\) just like the DDH challenger. As public key it outputs \(pk := (q, p, g, g^x)\) and as secret key \(sk := (q, p, g, x)\).

To encrypt a message \(m \in G_q\) using the public key \((q, p, g, \alpha)\), the encryption algorithm chooses a random \(y \in \{1, \ldots, q\}\) and outputs the ciphertext \((g^y, m \cdot \alpha^y)\), i.e., \(m\) is multiplied by \(g^y\). The IND-CPA security of this scheme follows from Assumption 5.4. An adversary obtains \((q, p, g, g^x)\) from \(pk\) and \(g^y\) from the ciphertext, but cannot decide whether \(m\) was multiplied by \(g^{y'}\) (resulting in the encryption of \(m\)) or by \(g^r\) which results in the encryption of a random message.\(^2\) The decryption algorithm, having access to the secret exponent \(x\), can easily compute \(g^{xy}\) and invert the encryption.

Proving security guarantees for ElGamal in a KDM-like setting seems difficult. Challengers hold multiple key-pairs and each key-pair is selected independently from the others. For ElGamal this means that each key-pair contains different values for \(q\), \(p\), and \(g\), i.e., they work on different groups. This makes it difficult to give proofs that involve multiple keys. Since the keys are incompatible, it is not clear how to simulate encryption queries which depend on several keys. This is why we introduced parameterized encryption schemes in section 2. The parameter generation algorithm allows us to compute a common set of variables which is used to create the key-pairs. Thereby we can choose a common group for all keys.\(^3\) Intuitively this does not weaken the scheme, since its security does not depend on the choice of the group (which is given to the adversary anyway), but solely on the secret exponent \(x\). Furthermore our security definitions prevent that parameter sets contain security critical information. This is because we let the challengers output their parameter sets in the beginning. If a scheme is secure, than obviously the parameter sets contain no information that helps to break the scheme. Therefore all security critical parts of keys must be chosen by the key generation algorithm. It is important that adversaries can obtain the parameter sets, otherwise they could not use the sets to generate their own keys. This would limit the power of adversaries substantially. The following definition introduces a parameterized version of the ElGamal encryption scheme, where the parameter generation algorithm is used to establish common values for \(q\), \(p\), and \(g\).

Definition 5.5 (Parameterized ElGamal Encryption Scheme)

The parameterized ElGamal encryption scheme \(E_{\text{ElGamal}} = (\text{param}, \text{gen}, E, D)\) is defined as follows for security parameter \(k\):

- Parameter generation \text{param} randomly chooses a \(k\)-bit prime \(q\), a \(k_q(k)\)-bit prime \(p\) such that \(q|p - 1\), an element \(g \in \mathbb{Z}_p^*\) of order \(q\), and outputs \((q, p, g)\).

\(^2\)The actual proof additionally includes a hybrid argument, since we need to handle multiple encryption queries while having only a single DDH tuple. Definition 5.6 introduces a challenger which provides multiple DDH tuples.

\(^3\)Compare this to section 4, where we needed schemes with identical message spaces.
5. Towards KDM Security

- **Key generation** \( \text{gen} \) for parameter set \( (q, p, g) \) chooses a random value \( x \overset{\$}{\leftarrow} \{1, \ldots, q\} \) and computes \( \alpha := g^x \pmod{p} \). It sets \( \text{pk} := (q, p, g, \alpha) \) and \( \text{sk} := (q, p, g, x) \) and outputs the key-pair \( (\text{pk}, \text{sk}) \).

- **Encryption** \( \text{E}(\text{pk}, m) \) for a public key \( \text{pk} = (q, p, g, \alpha) \) and a message \( m \in G_q := \langle g \rangle \) is computed as follows: Choose \( y \overset{\$}{\leftarrow} \{1, \ldots, q\} \), compute \( \beta := g^y, c := m \cdot \alpha^y \pmod{p} \), and output \( c := (\beta, c') \).

- **Decryption** \( \text{D}(\text{sk}, (\beta, c')) \) for a secret key \( \text{sk} = (q, p, g, x) \) and a ciphertext \( (\beta, c') \) works by computing \( m := c' \cdot (\beta^x)^{-1} \pmod{p} \).

The KDM challenger or the multi-IND-CPA challenger for this scheme hold multiple key-pairs. In order to simplify relating their security guarantees and Assumption 5.4, we introduce another challenger, multi-DDH, which in contrast to the DDH challenger outputs multiple DDH tuples. Furthermore it can be asked to output multiple tuples with a common value for \( g^x \). This will facilitate the simulation of multiple encryption queries on the same key.

**Definition 5.6 (Multi-DDH Challenger)**

The multi-DDH challenger (for subgroups \( G_q \) of \( \mathbb{Z}_p^* \)) for a given security parameter \( k \) is defined as follows: It randomly chooses a \( k \)-bit prime \( q \), a \( k_p(k) \)-bit prime \( p \) such that \( q|p - 1 \), an element \( g \in \mathbb{Z}_p^* \) of order \( q \), and a bit \( b \overset{\$}{\leftarrow} \{R, F\} \). Furthermore it has an infinite sequence \( \bar{x} = (x_i)_{i \in \mathbb{N}} \) where each \( x_i \), when first used, is initialised as \( x_i \overset{\$}{\leftarrow} \{1, \ldots, q\} \). Whenever it receives an input \( i \), it randomly chooses the values \( y, z \overset{\$}{\leftarrow} \{1, \ldots, q\} \) and depending on the value of \( b \), it outputs

- \((q, p, g, g^{x_i}, g^y, g^{x_i} \cdot y)\) if \( b = R \)
- \((q, p, g, g^{x_i}, g^y, g^z)\) if \( b = F \).

We write \( \text{multi-DDH}(b) \) to denote the multi-DDH challenger which initialized its bit to \( b \).

The multi-DDH advantage of an adversary \( A \) is defined as

\[
\text{Adv}^{\text{multi-DDH}}(A) := |\Pr(A^{\text{multi-DDH}}(R) = F) - \Pr(A^{\text{multi-DDH}}(F) = F)|.
\]

The tasks to gain a non-negligible advantage against the DDH or the multi-DDH challenger are equally hard. To prove this we define a further challenger which will help us to transform a fake DDH tuple into multiple tuples that look like fake ones. The challenger outputs a series of values \( \bar{\beta}, \bar{\gamma} \) and the challenge is to decide whether or not there is a constant exponent \( z \), such that \( \beta_i^z = \gamma_i \) for all \( i \).

**Definition 5.7 (Multi-DLog Challenger)**

The multi-DLog challenger (for subgroups \( G_q \) of \( \mathbb{Z}_p^* \)) for a given security parameter \( k \) is defined as follows: It randomly chooses a \( k \)-bit prime \( q \), a \( k_p(k) \)-bit prime \( p \) such that \( q|p - 1 \), an element \( g \in \mathbb{Z}_p^* \) of order \( q \), a bit \( b \overset{\$}{\leftarrow} \{R, F\} \), and a value \( z \overset{\$}{\leftarrow} \{1, \ldots, q\} \). Whenever it receives an input, it randomly chooses the values \( y, z' \overset{\$}{\leftarrow} \{1, \ldots, q\} \), sets \( \beta := g^y, \gamma := g^{z'} \), and depending on the value of \( b \) it outputs

- \((q, p, g, \beta, \beta^z)\) if \( b = R \)
- \((q, p, g, \beta, \gamma)\) if \( b = F \). 

23
5.2. Variations of DDH and the ElGamal Encryption Scheme

We write multi-DLog(b) to denote the multi-DLog challenger which initialized its bit to b. The multi-DLog advantage of an adversary A is defined as

\[ \text{Adv}^{\text{multi-DLog}}(A) := |\Pr(A^{\text{multi-DLog}(R)} = F) - \Pr(A^{\text{multi-DLog}(F)} = F)|. \]

A similar challenge is found in [7]. There a hybrid argument is indicated which shows that winning this challenge is a hard problem, given Assumption 5.4. Intuitively this means that raising a series of independent values to a constant secret power is as good as picking values at random.

Lemma 5.8 (DDH \rightarrow Multi-DLog)

If Assumption 5.4 (Hardness of DDH problem) is correct, then it holds, that for every efficient adversary A the multi-DLog advantage \( \text{Adv}^{\text{multi-DLog}}(A) \) is negligible.

Proof: We prove this using a hybrid argument. The hybrid challenger \( H^n \) is defined like the multi-DLog challenger in Definition 5.7 except that it acts like multi-DLog(R) for its first \( n \) outputs and like multi-DLog(F) for its later outputs.

Assume that there exists an efficient adversary A and a polynomial \( \delta \) such that for infinitely many values of the security parameter \( k \) we have \( \text{Adv}^{\text{multi-DLog}}(A) > \frac{1}{\delta(k)} \). Assume A makes at most \( \phi(k) \) queries (\( \phi(k) \) is polynomially bounded).

We construct an adversary B against the DDH challenger which uses A as a blackbox. The input of B is \( (q, p, g, \alpha, \beta, \gamma) \) where \( \alpha = g^z \), \( \beta = g^y \) for some \( z, y \in \{1, \ldots, q\} \) and \( \gamma \) is either \( g^{yz} \) or \( g^r \) for some \( r \in \{1, \ldots, q\} \). At first B randomly chooses a value \( n \leftarrow \{0, \ldots, \phi(k) - 1\} \). When B receives the \( n' \)-th input from A it randomly chooses values \( y', z' \leftarrow \{1, \ldots, q\} \) and outputs

- \( (q, p, g, g^{y'}, \alpha^{y'}) \) if \( n' \leq n \)
- \( (q, p, g, \beta, \gamma) \) if \( n' = n + 1 \)
- \( (q, p, g, g^{y'}, g^{z'}) \) if \( n' \geq n + 2 \).

The final output of A is also the final output of B.

We will now determine the DDH advantage of B. It is easy to see that for \( n' \leq n \) the output of B is indistinguishable from the output generated by multi-DLog(R) with secret exponent \( z \). Similarly for \( n' \geq n + 2 \) the output of B is indistinguishable from the output generated by multi-DLog(F).

If B interacts with DDH(R), output number \( n + 1 \) is indistinguishable from an output of multi-DLog(R), since \( \gamma = g^{yz} \). Therefore B simulates the behaviour of the hybrid challenger \( H^{n+1} \). Similarly if B interacts with DDH(F), output number \( n + 1 \) is indistinguishable from an output of multi-DLog(F), making it simulate the behaviour of \( H^n \). This allows us to determine the DDH advantage of B:
is $R$ outputs \((q, p, g, \alpha)\). 

Proof: Assume we are given an efficient adversary \(B\) and the multi-DDH and the multi-DDH problem are equally difficult. The hardness of the multi-DDH problem. Using this theorem, it is easy to conclude that the DDH and the multi-DDH problem are equally difficult. 

Towards KDM Security

\[
\text{Adv}^\text{DDH}(B) = \left| \Pr(B^{\text{DDH}(R)} = F) - \Pr(B^{\text{DDH}(F)} = F) \right|
\]

\[
= \frac{1}{\phi(k)} \sum_{n=0}^{\phi(k)-1} \Pr(A^{H_{n+1}} = F) - \frac{1}{\phi(k)} \sum_{n=0}^{\phi(k)-1} \Pr(A^{H_n} = F)
\]

\[
= \frac{1}{\phi(k)} \left| \sum_{n=0}^{\phi(k)-1} \left( \Pr(A^{H_{n+1}} = F) - \Pr(A^{H_n} = F) \right) \right|
\]

\[
= \frac{1}{\phi(k)} \left| \Pr(A^{H_1} = F) - \Pr(A^{H_0} = F) \right|
\]

\[
= \frac{1}{\phi(k)} \left| \Pr(A^{\text{multi-DLog}(R)} = F) - \Pr(A^{\text{multi-DLog}(F)} = F) \right|
\]

\[
= \frac{1}{\phi(k)} \text{Adv}^{\text{multi-DLog}}(A)
\]

\[
> \frac{1}{\phi(k) \delta(k)}
\]

This result is used to prove the following theorem. It states that Assumption 5.4 implies the hardness of the multi-DDH problem. Using this theorem, it is easy to conclude that the DDH and the multi-DDH problem are equally difficult.

Theorem 5.9 (Hardness of the Multi-DDH Problem in \(G_q\))

If Assumption 5.4 (Hardness of DDH problem) is correct, then it holds that for every efficient adversary \(A\) the multi-DDH advantage \(\text{Adv}^{\text{multi-DDH}}(A)\) is negligible.

Proof: Assume we are given an efficient adversary \(A\) against the multi-DDH challenger and a polynomial \(\delta\) such that for infinitely many values of the security parameter \(k\) we have \(\text{Adv}^{\text{multi-DDH}}(A) > \frac{1}{\delta(k)}\). We construct an adversary \(B\) against the DDH challenger which uses \(A\) as a blackbox:

\(B\) gets a challenge \((q, p, g, \alpha, \beta, \gamma)\) from the DDH challenger, where \(\alpha = g^x\), \(\beta = g^y\), and \(\gamma = g^z\) for some \(x, y, z \in \{1, \ldots, q\}\). Furthermore it has an infinite sequence \((\tilde{x}_i)_{i \in \mathbb{N}}\) where each \(\tilde{x}_i\), when first used, is initialised as \(\tilde{x}_i \in \{1, \ldots, q\}\).

Whenever \(B\) receives a query \(i\) from \(A\) it randomly chooses the value \(\tilde{y} \in \{1, \ldots, q\}\) and outputs \((g, p, g, \alpha \tilde{x}, \beta \tilde{y}, \gamma \tilde{z}, \tilde{y})\). The result \(b^*\) of \(A\) is also the result of \(B\).

In the following we determine the DDH advantage of \(B\). If the bit \(b\) of the DDH challenger is \(R\), \(B\) acts like multi-DDH\((R)\) in the view of \(A\): \(\Pr(B^{\text{DDH}(R)} = F) = \Pr(A^{\text{multi-DDH}(R)} = F)\)

For the case \(b = F\) we show that \(\Pr(B^{\text{DDH}(F)} = F) - \Pr(A^{\text{multi-DDH}(F)} = F)\) is negligible by using Lemma 5.8. This is done by constructing an adversary \(B'\) against the multi-DLog challenger which uses \(A\) as a blackbox. In the view of \(A\) adversary \(B'\) will act like \(B^{\text{DDH}(F)}\) if the bit \(b'\) of the multi-DLog challenger is \(R\) and it will act like multi-DDH\((F)\) if \(b' = F\). It is defined as follows:

It has an infinite sequence \((x_i')_{i \in \mathbb{N}}\) where each \(x_i'\), when first used, is initialised as \(x_1' \in \{1, \ldots, q\}\). Furthermore it randomly chooses the values \(x', y' \in \{1, \ldots, q\}\). Whenver \(B'\) receives a query \(i\) from \(A\), it requests a new tuple \((q, p, g, \beta', \gamma')\) from the multi-DLog challenger and sends \((q, p, g, g^{x'}x, \beta' y', \gamma' z, y')\) to \(A\). The result \(b^*\) of \(A\) is also the result of \(B'\).

If the bit \(b'\) of the multi-DLog challenger is \(F\) then \(\beta'\) and \(\gamma'\) are chosen independently at random in each round. Therefore \(B'\) acts like multi-DDH\((F)\) in the view of \(A\), which implies \(\Pr(B^{\text{multi-DLog}(F)} = F) = \Pr(A^{\text{multi-DDH}(F)} = F)\). If \(b' = R\), the tuples sent to \(A\) are of the form \((q, p, g, g^{x'}x, g^{y'}y', g^{z'}z, y')\), where \(x', x', y',\) and \(z'\) are fixed (\(z'\) is the secret exponent of the multi-DLog challenger) and \(y'\) is chosen at random for every round by the multi-DLog challenger. When interacting with \(B^{\text{DDH}(F)}\) the tuples sent to \(A\) are of the
form \((q, p, g^{\bar{x}_p}, g^{\bar{y}}, g^{\bar{y}_z})\), where \(\bar{x}_p, x, y, z\) are fixed and \(\bar{y}\) is chosen at random for every round. Therefore \(A\) cannot distinguish whether it interacts with \(B_{\text{multi-DLog}(R)}\) or with \(B_{\text{DDH}(F)}\). So it holds that \(\Pr(B_{\text{multi-DLog}(R)} = F) = \Pr(B_{\text{DDH}(F)} = F)\).

So it holds that \(|\Pr(B_{\text{DDH}(F)} = F) - \Pr(A_{\text{multi-DDH}}(F) = F)| = |\Pr(B_{\text{multi-DLog}(R)} = F) - \Pr(B_{\text{DDH}(F)} = F)|\), which is negligible by Lemma 5.8.

Now we can calculate the DDH advantage of \(B\):

\[
\text{Adv}^{\text{DDH}}(B) = |\Pr(B_{\text{DDH}(R)} = F) - \Pr(B_{\text{DDH}(F)} = F)|
\]

\[
= |\Pr(A_{\text{multi-DDH}(R)} = F) - \Pr(B_{\text{DDH}(F)} = F)|
\]

\[
> |\Pr(A_{\text{multi-DDH}(R)} = F) - \Pr(A_{\text{multi-DDH}(F)} = F)| - \frac{1}{2^{\delta(k)}}
\]

\[
= \text{Adv}^{\text{multi-DDH}}(A) - \frac{1}{2^{\delta(k)}}
\]

\[
> \frac{1}{2^{\delta(k)}}
\]

\[\square\]

Corollary 5.10 (DDH \(\leftrightarrow\) Multi-DDH)

For every efficient adversary \(A\) the DDH advantage \(\text{Adv}^{\text{DDH}}(A)\) is negligible if and only if for every efficient adversary \(B\) the multi-DDH advantage \(\text{Adv}^{\text{multi-DDH}}(B)\) is negligible.

**Proof:** The implication DDH \(\rightarrow\) Multi-DDH follows from Theorem 5.9. Proving that \(\neg\text{DDH}\)

implies \(\neg\text{Multi-DDH}\) is trivial. An adversary that wins against the DDH challenger will win against the multi-DDH challenger by presenting him a single tuple. \[\square\]

Using this result, it is easy to prove that the parameterized ElGamal encryption scheme is IND-CPA-secure, given Assumption 5.4. An adversary \(A\) against the multi-IND-CPA challenger for this scheme can be used in a blackbox construction to attack the multi-DDH challenger. The \(i\)-th public key is given to \(A\) by querying the multi-DDH challenger on \(i\) and outputting the first four components \((q, p, g^{\bar{x}_i}, g^{\bar{y}_i})\) of the result. An encryption query \((\text{enc}, i, m)\) from \(A\) is handled by obtaining a tuple \((q, p, g^{\bar{x}_i}, \beta, \gamma)\) from the multi-DDH challenger and returning the ciphertext \(c := (\bar{\beta}, m \cdot \gamma)\) to \(A\). If we are dealing with multi-DDH\((R)\) then \(\gamma = \beta^{\bar{x}_i}\) and \(c\) is a valid encryption of \(m\). If we play against the fake challenger then \(\gamma\) is random and therefore \(c\) is the encryption of a random message.

We will now prove a weakened form of KDM security for the parameterized ElGamal encryption scheme. An adversary can obtain the values \((g^{\bar{x}_i})_{i \in P}\) for some polynomially bounded \(P\), where the \(x_i\)'s are the secret exponents of the challenger. Using them the adversary can compute values of the form \(g^{\bar{b}+c}\), for some \(\bar{b} \in \{1, \ldots, q\}, c \in \{1, \ldots, q\}\). So it can compute values, where the exponent is a linear combination of the secret exponents, but it is assumed that no efficient adversary can compute a value where an exponent occurs in squared form [14]. We will prove that an adversary does not gain an advantage by asking for encryptions of messages where the respective secret key occurs in squared form. Furthermore it is safe to encrypt Diffie-Hellman-like values of the form \(g^{\bar{x}_i \cdot \bar{x}_j}\) using the \(i\)-th public key. So we can simulate the encryption of messages which an adversary cannot compute by himself (Otherwise the result would not be very interesting). We allow encryption queries \((\text{enc}, i, f)\) where \(f\) is of the form \(f(\text{sk}) = g^{\bar{e}_i \cdot (\bar{a} \bar{x}) + \bar{b} \bar{x} + c}\).

Theorem 5.11 (Weakened KDM Security of ElGamal)

Provided that Assumption 5.4 (Hardness of DDH problem) is correct, the following holds: Every efficient adversary \(A\) against the KDM challenger for the parameterized ElGamal encryption scheme has a negligible KDM-advantage, if for all encryption queries \((\text{enc}, i, f)\) it holds
that \( f \) is of the form \( f(\mathbf{s}k) = g^{x_i \cdot (\bar{a} \cdot \bar{x}) + \bar{b} \cdot \bar{x} + c} \), for some \( c \in \{1, \ldots, q\} \) and \( \bar{a}, \bar{b} \in \{1, \ldots, q\}^\mathcal{P} \), for some polynomially bounded \( \mathcal{P} \).

**Proof:** Given an adversary \( A \) against the KDM challenger which only uses encryption queries as in Theorem 5.11, we construct an adversary \( B \) against the multi-DDH challenger which uses \( A \) as a blackbox.

- If \( B \) receives a query (getpk, \( i \)) from \( A \), it sends \( i \) to the multi-DDH challenger and receives \( (q, p, g, \alpha, \beta, \gamma) \) from it. The public key \( (q, p, g, \alpha) \) is sent to \( A \).

- If \( B \) receives a query (enc, \( i, g^{x_i \cdot (\bar{a} \cdot \bar{x}) + \bar{b} \cdot \bar{x} + c} \)) from \( A \), it sends \( j \) to the multi-DDH challenger for all \( j \leq \mathcal{P} \) to obtain \( (q, p, g, \alpha_j, \beta_j, \gamma_j) \). It then sends the following ciphertext to \( A \):
  \[
  (\beta_i \cdot \prod_j \alpha_j^{-a_j} \cdot \gamma_i \cdot g^c \cdot \prod_j \alpha_j^{b_j})
  \]

- If \( B \) receives the final result \( b^* \) from \( A \) it also outputs \( b^* \).

We will now show that \( B \) acts like the KDM challenger in the view of \( B \). If the bit \( b \) of the multi-DDH challenger is \( F \), then \( \beta_i \) and \( \gamma_i \) are chosen uniformly at random. So the resulting encryption is, as required, the encryption of a random string. If \( b = R \), the ciphertext looks like a correctly generated encryption of \( g^{x_i \cdot (\bar{a} \cdot \bar{x}) + \bar{b} \cdot \bar{x} + c} \) to \( A \): The first component of the ciphertext is random and it holds:

\[
\begin{align*}
D(\mathbf{s}k_i, (\beta_i \cdot \prod_j \alpha_j^{-a_j} \cdot \gamma_i \cdot g^c \cdot \prod_j \alpha_j^{b_j})) &= (\beta_i \cdot \prod_j \alpha_j^{-a_j})^{-x_i} \cdot \gamma_i \cdot g^c \cdot \prod_j \alpha_j^{b_j} \\
&= (g^{y_i} \cdot \prod_j g^{-a_j \cdot x_j})^{-x_i} \cdot g^{x_i \cdot y_i} \cdot g^c \cdot \prod_j g^{b_j \cdot x_j} \\
&= g^{-x_i \cdot y_i + x_i \cdot (\sum_j a_j \cdot x_j) + x_i \cdot y_i + c + \sum_j b_j \cdot x_j} \\
&= g^{x_i \cdot (\bar{a} \cdot \bar{x}) + \bar{b} \cdot \bar{x} + c}
\end{align*}
\]

Therefore it holds that \( \text{Adv}^{\text{multi-DDH}}(B) = \text{Adv}^{\text{KDM ElGamal}}(A) \).
Section 6

Conclusion

We have examined key-dependent message security, a security notion which is stronger than semantic security. We have shown that IND-CPA-secure encryption schemes are not necessarily KDM-secure, even if adversaries are restricted to produce no key cycles of length less than an arbitrary $l$. A similar result was shown in [3], but only for stateful encryption schemes.

It is still an open problem whether KDM security can be achieved under standard complexity assumptions without random oracles. Solving this problem goes beyond the scope of this thesis. Instead we presented encryption schemes which fulfill weakened forms of KDM security: At first we required the adversaries to only query partially constant functions. The resulting weakened KDM security was proven for an encryption scheme that is, in some sense, aware of the function being queried. In our second attempt we presented a series of challenges, in particular multi-DDH, which is proven equivalent to DDH. Using multi-DDH we proved a weakened form of KDM security for the parameterized ElGamal encryption scheme, where adversaries can only query functions of a certain format.
Bibliography


Bibliography


