Hello everyone. Today I will be presenting my project, on the studies of security of deterministic encryption schemes. In the presentation, I will first introduce the idea of encryption schemes with additional properties which are not secure under the standard security model. In particular, we will look at DE and its applications. We will see that the scheme is secure under its security definition but not secure in practice in database applications. Finally, I will discuss the approaches I have taken in the research.

In modern cryptography, security is usually defined in terms of IND or SS. Specifically, for a scheme to be secure in this setting, it has to be probabilistic. For instance, textbook RSA and AES are not secure cryptosystems because they are deterministic. But being probabilistic makes it hard to process encrypted data efficiently.

On the other hand, there are schemes with additional properties that allows for efficient processing of encrypted data. One such example is OPE. Given messages $m_0 < m_1$, the encryption of m_0 is less than that of m_1. So the scheme reveals relative distance of the underlying plaintext. This scheme cannot satisfy IND security because of this special property. So we need to understand what is the maximal level of security these schemes can offer.

Now let's turn our attention to DE. I will show two constructions of it which have provable security. In any case, the idea of the constructions is to take a randomized encryption scheme and flip its coins deterministically.

In the first construction, hash function is used to do exactly that. In the second construction, trapdoor permutation is used in place of hash function. For those who is not familiar with trapdoor permutation, it is a family of functions that is easy to compute but hard to invert, just like one-way permutations. But if given some additional information, known as the 'trapdoor', the inversion can be easily computed. The construction of DE with TDP is very similar to that of hash function, just that the plaintext is transformed by the TDP before encryption.

DE is very useful in some applications. For example, it is possible that the source of randomness is poor in probabilistic encryption schemes and DE can be used to generate randomness with provable security. In fact, it is possible to turn a DE into a randomized encryption algorithm but that is beyond the scope of the presentation. DE can also be used to encrypt data in databases. As the entries in the database are deterministic, various dictionary data structures can be used to allow for efficient searching. For example, with red-black tree or 2,3,4-tree, searching can be done in log-time. If more advanced data structure such as Van Emde Boas tree is used, searching can be done in log-log-time.

Let us analyse the security of DE. Recall that the scheme is deterministic, it cannot meet the usual IND security. So we need to have an alternative security notion. In the original papers for DE, the authors suggest to give the adversary less power. Formally, we can define an IND adversary as a triple (I_c, I_m, I_g), where I_c is the function to generate a state, that will be used by I_m. I_m will take in that state and generates a pair of messages. Then the challenger encrypts one of the message and returns it to the adversary. The adversary uses I_g to guess the challenge bit b. The IND game for DE is summarized in the following slide.

I wish to point out a few key differences between the standard IND security and the definition

here. Firstly, I_c and I_m have no access to the public key. Secondly, I_g have no knowledge of the challenge messages. Finally, the message space must have high min-entropy, i.e. the plaintexts need to be very random.

Alternatively, the security can be defined as a IND-DCPA game. The adversary will query pairs of messages, with no repetition at all, and the challenger encrypts one of the messages for each pair. The adversary has to return his guess bit $b^{'}$ in polynomial time.

DE has been proven secure in both of the security definitions but in database applications, it is not secure at all. This is because DE leaks frequency so one can use frequency attack against it. In the figure, we try to encrypt letters A, B, C and D using DE. The frequency of the encryptions is marked by the blue line. But if we know that for this particular database, the distribution of the letters is of certain shape, say the red line, then we can just match the frequencies to guess the underlying plaintext.

It is worth to note that, although we have security guarantees, it may not be secure in reality. This is because the assumptions made in the security definition may not necessarily be true in practice. If we go back to the assumptions in the security definition of DE, we can see that the assumptions do not hold in the attack. In particular, I_g does know about the set of plaintexts so the attack is successful.

In summary, in the presentation, we have discussed the construction of DE and its usefulness. We have seen that although in theory, with some assumptions on the security model, DE can be prove secure, it may not be secure in real applications. In particular, it is not secure in database applications.

As a consequence, my research tries to address the following questions. One, how should security notion be defined for DE in application to databases? Two, is there an encryption scheme that is secure under that security definition?