On the Random Oracle Model and the Game Hopping Technique

Murat Ak, Turgut Hanoymak

***Abstract*—In modern cryptography, provable security gained much attention because it provides theoretical evidence about the security of public key schemes. Provable security is achieved via a reduction technique which relates the security of a scheme to a mathematically hard problem. Typically, a reduction from a mathematically hard problem to the cryptographic scheme is given and it is argued that if there exists an adversary that can break the scheme, one may construct an algorithm solving the underlying hard problem using the adversary as a subroutine. There are several techniques used in such proofs two of which are random oracles and game hopping. In this paper, we discuss the random oracle model (ROM) which has been proposed as an alternative to the standard model. In ROM, basically, hash functions are modeled as random oracles, which is a hypothetical black box that gives truly random outputs to distinct inputs. We also discuss two major objections on the validity ROM and we point out that one of these objections is not indeed sound. Then we give the game hopping technique since sometimes security proofs in cryptography may be constructed as sequence of games. In particular, we give some concrete illustrations of public key encryption schemes and their security proofs that utilize random oracles and game hopping.**

***Index Terms*—Provable Security, Public Key Encryption, Random Oracle Model, Game Hopping**

I. Introduction

SECURITY of the cryptographic schemes in the public key setting typically depends on the mathmatical problems that are believed to be hard to solve such as integer factorization and discrete logarithms.

Anyone who proposes a new public key encryption scheme must first take potantial security proofs into consideration because otherwise the resulting scheme becomes most probably vulnurable. Currently, there are two models for proving security :

- The standard model is the default security model which assumes nothing but well established mathmatically hard problems. However, the problem with this model is that the schemes which can be proven secure turns out to be inefficient. There exist schemes that are both secure and efficient such as [4]
- The random oracle model which was proposed by Bellare and Rogaway [3] is an alternative to the standard counterpart. In this model, hash functions are modeled as random oracles which are not actually possible in the real world however, due to this relaxation, one may construct more efficient schemes. As a matter of fact, random oracle proofs are not considered to be rigorous mathematical proofs and indeed it is shown that there exist schemes which can be proven in the random oracle model but completely insecure in real world such as [8]. Probably, this is why the use of ROM is objected by a number of cryptographers such as Menezes and Koblitz [1]. But this does not make ROM completely useless because at least it guarantees that there are no inherent design flaws and this is certainly better than providing no proof at all.

We stress that, it is still preferable to use a scheme secure in the standard model even if it is slightly less efficient as opposed to the random oracle counterpart.

Random oracle is a hypothetical "magic" box that is assumed to have the following two properties:

- It produces truly random outputs
- For each input it gives the same output

The objections on ROM comes from that fact that random oracles do not exist in the real world, and the simulator needs to see the queries of the adversary which is not so realistic. Also it is assumed that the simulator can set the output of the random oracle as it wishes. This is called *programmability* yet it is not realistic, neither.

One of the most popular techniques used in security proofs is game hopping. In this method, we construct different versions of the security game played between the adversary and the challenger. Typically, the first game is the original security game. And the last game is a game in which the adversary gains no information, theoretically. Finally, the basic argument of game hopping methods is that since the views of the adversary in successive versions of the game are indistinguishable, the success probability in the original attack game is related to the success probability in the last game which acts like a one-time pad, thus must be negligible due to transitivity.

In this paper, we will present the random oracle methodology in detail and give three example proofs. Then, we will present two proofs that use game hopping techniques.

II. Preliminaries and Notation

A. Reduction Proofs

Most of the security proofs in the literature are in the form of a reduction. Typically, a mathematically hard problem M is reduced to breaking the scheme S that is wanted to be proven secure. Existence of such a reduction implies that the problem of breaking the scheme S is as hard as M. This implication stems from the following contraction argument: If there exist a

M. Ak is with the Department Computer Engineering, Bilkent University, Ankara, 06800, Turkey. e-mail: muratak@cs.bilkent.edu.tr

T. Hanoymak is with the Institute of Applied Mathematics, Middle East Technical University, Ankara, 06531, Turkey. He is also affiliated with Yüzüncü Yıl University, Van, Turkey. e-mail: hturgut@metu.edu.tr

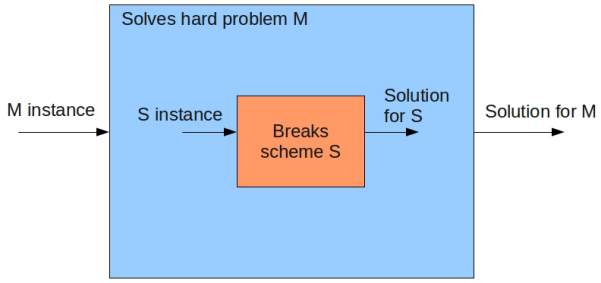

Fig. 1. The idea of reduction illustrated. Given an instance of M, one needs to form an instance of the scheme S and use the breaker to break S and this result is used to solve M instance.

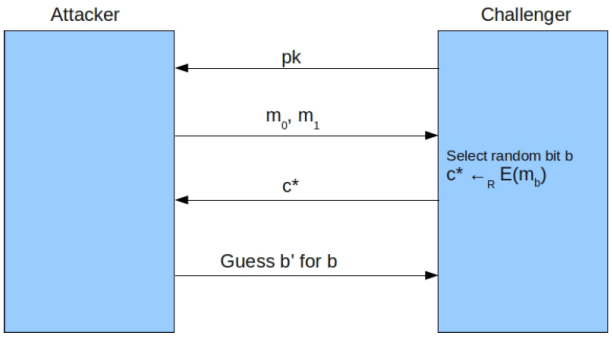

Fig. 2. The chosen plaintext attack (CPA) game is illustrated.

polynomial time algorithm A that breaks the scheme S, then due to this reduction, one may construct a polynomial time algorithm B which uses A as a subroutine to solve M which is assumed to be impossible. See Figure 1.

B. Security Models

When we talk about the security of a cryptographic scheme, we need to define the goal and the capabilities of the adversary. As the goal becomes more difficult or as the capabilities are more limited the security proof becomes easier. In the cryptography literature, several goals and capabilities are discussed so far. The most popular adversarial goal is to distinguish between the plaintext of a given ciphertext. When we prove that such an adversary cannot exist, this is called indistinguishability. Note that indistinguishability means that a ciphertext gives semantically no information about the plaintext. So, it is also called semantic security.

There are several possible capabilities of an attacker in the public key setting depending on the availability of the decryption oracle which is a hypothetical black box that is presented to the attacker so that it can make decryption queries of its own choice and gets the corresponding plaintexts. This captures the possible real life attacks that consist of attackers that has gained temporary access to the decryption oracle. In this respect, there are three types of decryption oracle access:

- CPA (Chosen plaintext attack): if there is no decryption oracle access at all, we call this a chosen plaintext attack.
- CCA1 (Non-adaptive chosen ciphertext attack, or lunchtime attack): adversary can access the decrpytion oracle until it sees the ciphertext it needs to break.
- CCA2 (Adaptive chosen ciphertext attack): adversary always has access to the decryption oracle but querying the ciphertext it needs to break is prohibited.

The most widely accepted level of security is CCA2 security. However, in CCA security proofs, there is a difficulty of providing decryption oracles. Because usually, the decryption key is not in the control of the simulator but it rather inherently depends on the instance of the hard problem. In this paper, we focus on CPA security, since it is enough to present the main idea of random oracle proofs and game hopping methods.

C. Semantic Security

The notion of semantic security captures the following: Even if the adversary chooses two messages m_0, m_1 and gets the ciphertext of one of them c^*, it cannot distinguish whether $c^* = E(m_0)$ or $c^* = E(m_1)$. This means that the ciphertext gives *no information* about the plaintext. Note that semantic security requires the encryption algorithm to be randomized. Otherwise, the adversary can always encrypt both the messages and check with the challenge ciphertext c^*.

In order to prove semantic security we consider a hypothetical game between an adversary and a challenger. There are three steps:

1) Adversary chooses two messages of its own choice and sends them to the challenger.
2) Challenger selects one of these messages randomly, encrypts it and sends back the ciphertext as the challenge.
3) In the final step, the adversary tries to guess the message that is used in the challenge.

These steps are also shown in Figure 2.

Then the success of the adversary is measured with its advantage in this game defined as $Pr[b' = b] - 1/2$.

D. Cryptographic Hardness Assumptions

In this section, we recall four popular hardness assumptions that the security proofs we will discuss later depend on. Namely, the RSA assumption, the discrete log assumption, and decisional and computational versions of the Diffie-Hellman assumption.

RSA assumption Let N be the product of two large primes p and q, and let e and d be two integers such that

$$ed = 1 \mod \phi(N)$$

Given N, e, y it is computationally hard to find an $x \in \mathbb{Z}_N^*$ such that $x^e = y \mod N$.

Discrete logarithm assumption Let $\mathbb{G}$ be a group of primer order q. Given a generator g and an element of $\mathbb{G}$, h, finding x such that $g^x = h$ is computationally hard.

Computational Diffie-Hellman assumption Let $\mathbb{G}$ be a group of primer order q. Given a generator g and two elements of $\mathbb{G}$, g^a, g^b, finding g^{ab} is computationally hard.

Decisional Diffie-Hellman assumption Let $\mathbb{G}$ be a group of primer order q. Given a generator g and three elements of $\mathbb{G}$, g^a, g^b, g^c, finding whether $ab = c \mod q$ is computationally hard. This assumption can also be represented in terms of

probabilities as follows: Let D be a polynomial time algorithm that is designed for deciding whether a three-tuple is a DDH tuple, and let

$$Pr[D(g^a, g^b, g^{ab}) = 1] - Pr[D(g^a, g^b, g^c) = 1]$$

where a, b, and c are selected randomly from $\mathbb{Z}_q$ is defined as the advantage of D in distinguishing a DDH tuple distribution from a random one. The DDH assumption assumes that this advantage is negligible.

III. Random Oracle Model and Security Proofs

As we mentioned previously, random oracle model is well accepted as a method for proving security of public key encryption schemes, because the schemes that can be proven in the standard model require more complex operations whereas, the use of hash functions is usually sufficient to obtain schemes that can be proven in the random oracle model. The core idea of the random oracle model is that we assume hash functions are truly random functions. In fact, it is tricky to use random oracle idea in a rigorous proof since hash functions are not truly random. In random oracle proofs, basically, the adversary claims to perform a successful attack even if the challenger simulates and provides a truly random function of its choice. This is indeed a very strong claim and it is not very surprising that an attack of this kind can be proven to be impossible. But, random oracle model still draws a good deal of attention because usually, efficient schemes can be proven secure only in this model. In the random oracle model proofs, hash functions are thought as random oracles and are provided for the attacker by the challenger (simulator) who has full control over this oracle. The main assumptions of the random oracle model are the following:

- When the attacker needs to use the hash function, it has to submit a query to the random oracle simulated by the challenger.
- The challenger can set the output of the random oracle query to any value of its choice which is called programmability.

In order to make this idea more clear, we now present different random oracle proofs from the literature.

A. CPA Secure RSA Encryption in the ROM

The first scheme we will discuss is an IND-CPA secure version of RSA based encryption scheme. The scheme is defined by the following three algorithms:

- **Setup**: Let N be an RSA number and e, d is an RSA key pair. Let $H : \mathbb{Z}_N^* \to \{0,1\}^\ell$ be a hash function.
- **Encrypt**(m): Given a message $m \in \{0,1\}^\ell$, a random $r \in \mathbb{Z}_N^*$ is chosen and m is encrypted as:

$$c = r^e \mod N, H(r) \oplus m$$

- **Decrypt**(c): Given a ciphertext $c = (c_1, c_2)$,

$$m = H(c_1^d) \oplus c_2$$

Theorem III.1 *If the RSA problem is hard, then the scheme above is IND-CPA secure in the random oracle model.*

Proof: The proof will proceed as follows: a reduction from breaking the scheme to solve the RSA problem is presented. Suppose there exists an adversarial algorithm A that can break the scheme. We will construct another algorithm B that solves the RSA problem using A as a subroutine. Algorithm B will simulate an attack game for the successful attacker A given. We construct B as follows:

Algorithm 1 $B(N, e, y)$

1: Choose $s \in_R \mathbb{Z}_N^*$. Consider $t = y^{1/e}$ which we do not actually know, and due to programmability, let $H(t) = s$.
2: Simulate IND-CPA game for A as follows:
3: Provide the public key (N, e) to A.
4: Maintain an initially empty table T for random oracle queries and set $t = \perp$.
5: Whenver A makes a random oracle query q:
6: Check whether $q^e = y$ if so, set $t = q$ return s.
7: Otherwise, randomly choose $H(q)$, save $(q, H(q))$ to table T and return $H(q)$ to A (provided that q is not queried before, in that case return the same value from the table).
8: A outputs two messages m_0, m_1.
9: Randomly select a bit b and give the challenge ciphertext $c = (c_1, c_2) = (y, s \oplus m_b)$
10: Whenever A makes a random oracle query q it proceeds same as Line 5.
11: After the game is over return t.

The core idea of the proof is as follows: If t is not queried, then $H(t)$ is considered to be truly random, hence, c_2 is like a one-time pad encryption. If it is queried, then t must be the answer we are looking for, i.e. the answer of the RSA problem. So we can argue that Algorithm B solves the RSA problem if A can break the scheme. This contradicts with the RSA assumption, so the scheme is IND-CPA secure. ■

B. Security of Hashed El Gamal under CDH Assumption

Now, we will look at a similar but slightly more complicated example random oracle proof. The reduction will be similar to that of the previous section but there will be a probabilistic argument. This time, the encryption scheme is as follows:

- **Setup**: Let g be a generator of some prime order group $\mathbb{G}$. Let x be the private key and $h = g^x$ be the public key. Let $H : \mathbb{G} \to \{0,1\}^\ell$ be a hash function.
- **Encrypt**(m): Given a message $m \in \{0,1\}^\ell$, a random $r \in \mathbb{G}$ is chosen and m is encrypted as:

$$c = (g^r,\ H(h^r) \oplus m)$$

- **Decrypt**(c): Given a ciphertext $c = (c_1, c_2)$,

$$m = H(c_1^x) \oplus c_2$$

Theorem III.2 *If the computational Diffie-Hellman (CDH) problem is hard, then the scheme above is IND-CPA secure in the random oracle model.*

Proof: In order to prove this theorem, a reduction from breaking the scheme to solve the CDH problem will be given.

Similar to the previous proof, we will assume that we have an adversarial algorithm A that can break the scheme above and construct another algorithm B that solves the CDH problem using A as a subroutine. Algorithm B will be similar to that of the previous proof except that we cannot check the queries whether they correspond directly to the answer of the CDH problem we are looking for. However, note that when we set the parameters g^x and g^r to the input of the CDH problem g^a and g^b respectively, if the attacker is successful in breaking the scheme this means that it must have queried the oracle with g^{ab} somewhere in its execution. So, one of the queries consists of the answer of the CDH problem. Because otherwise, the ciphertext part c_2 is like a one-time pad due to the truly randomness assumption of random oracles. Hence, at the end of the game simulation, algorihtm B selects one of the queries randomly and returns it as the output. The probability of this output to be correct is $(1/q)\epsilon$ where q is the number of queries performed, and ϵ is the success probability of the adversary. Since the number of queries is polynomially bounded, the probability is divided into polynomial factor which is still non-negligible. Hence, the scheme is IND-CPA secure. ■

Note that in the proof of the IND-CPA RSA scheme, the success probability is preserved since we can explicitly test whether a random oracle query is the one that consists of the answer of the RSA problem, but we cannot do the same check in the proof of IND-CPA El Gamal scheme, but fortunately returning one of the queries still works since it decreases the success probability only in polynomial factor.

C. Existentially Unforgeable RSA Signature Scheme

In this section we will present an example proof in the random oracle model for a signature scheme. First, let us consider the plain RSA signature scheme:

- **Setup**: Let N be an RSA number and (e, d) is an RSA key pair.
- **Sign**(m): Given a message $m \in \mathbb{Z}_N^*$, m is signed as: $s = m^d \mod N$.
- **Verify**(s): Given a signature (m, s), verification equation is: $m = s^e \mod N$.

Unfortunately, this is not considered as a secure signature scheme because one can forge a message-signature pair by simply choosing a signature s randomly and computing the corresponding message m as $s^e \mod N$

Indeed, the most widely accepted security definition for signature schemes is as follows:

Existential unforgeability: A signature scheme is existentially unforgeable under an adaptive chosen message attack if there exists no polynomial time algorithm that can output a valid message-signature pair given access to a signature oracle.

So, according to this definition, the scheme above is not existentially unforgeable. However, it is possible to construct a secure signature scheme that is a modified version of the basic RSA signature scheme using a hash function as follows:

- **Setup**: Let N be an RSA number and (e, d) is an RSA key pair. Let $H : \mathbb{Z}_N^* \to \{0,1\}^\ell$ be a hash function.
- **Sign**(m): Given a message $m \in \{0,1\}^\ell$, m is signed as: $$s = H(m)^d \mod N$$
- **Verify**(s): Given a signature (m, s), verification equation is: $$H(m) = s^e \mod N$$

Recall that for signature schemes, in a *no-message attack*, adversary needs to find a valid message-signature pair without having access to a signature oracle.

Theorem III.3 *If the RSA problem is hard then the modified RSA signature scheme above is existentially unforgeable under a no-message attack in the random oracle model.*

Proof: The proof proceeds similar to the proof of the RSA encryption scheme of Section III-A. We construct an algorithm B for solving the RSA problem assuming that we have an algorithm A that can forge a signature pair. Algorithm B is constructed as follows: Given an RSA problem instance (N, e, y), it gives A the public key (N, e). Whenever A makes random oracle queries, B responds with random values. Except that for a randomly selected query, it responds with y instead of a random value. So, when the simulation is over, and A outputs a forgery (m, s), B outputs s. Note that s is the correct answer of the RSA problem, i.e. $y = s^e \mod N$ with a probability $(1/q)\epsilon$ where q is the number of random oracle queries and ϵ is the success probability of algorithm A. Because A must have queried m to the random oracle since otherwise $H(m)$ remains truly random and s cannot be determined thus cannot be included in a forgery. So, since a polynomially many random oracle queries are allowed, and ϵ is assumed to be non-negligible, the success probability of algorithm B is also non-negligible which contradicts the RSA assumption. So the scheme is existentially unforgeable under a no-message attack in the random oracle model. ■

Now we will present the same scheme is indeed existentially unforgeable under an adaptive chosen message attack in the random oracle model.

Theorem III.4 *If the RSA problem is hard then the modified RSA signature scheme above is existentially unforgeable under an adaptive chosen message attack in the random oracle model.*

Proof: The proof proceeds same as the proof of the no-message case except that the adversary must be provided with a signature oracle. This signature oracle can be simulated by using the programmability feature of the random oracle. When the adversary A requests a signature on message m, B sets $H(m)$ to be $s^e \mod N$ for a randomly selected s and returns s to A. Note that here, B does not have to know d explicitly, and since it sets $H(m)$ it can respond to later random oracle queries consistently with the previous ones. Together with the previous construction, this addition is sufficient to complete the proof. ■

An Important Remark on the Validity of the Random Oracle Model: In the cryptography literature, there are two major objections about the validity of the random oracle

model originating from the usage of hash functions to simulate random oracles:

- Programmability property is not possible in the real world:
  Since in the real world it is impossible to set the output of an hash function to a specific value, this is indeed a sound objection. Because most of the time, a concrete hash function (e.g. SHA-1) is used and the result is immediately set as soon as the input is given. Some sort of programmability would be possible in a highly hypothetical environment where hash families could be used instead of a concrete hash function.
- In the real world, it should be impossible for any other entity to see the hash evaluations that the adversary makes:
  Unlike the previous one, this objection may not be correct. Note that by arguing such an impossibility, one imposes that the hard problem being reduced cannot be broken by the same breaker as the scheme security of which is being proved. To make this point clear, suppose that we are proving the security of a scheme S by reducing a mathematically hard problem P to S. In such a reduction, we need to provide a breaker B_P for P assuming that we have a breaker B_S for S. If we do not allow B_P to see the hash evaluations that B_S makes, this implies the restriction that B_P cannot be designed by the same entity who designed B_S. However, this is an unnecessary restriction because it must be sufficient to show that if S can be broken so can P, no matter who designs the breaker algorithms B_S and B_P. So, since B_S and B_P can possibly be designed by the same entity (at least we cannot prohibit this), the hash evaluations of B_S should be visible to B_P.

IV. Game Hopping Techniques

In this section, we will look at another security proof technique, which is called game hopping. The idea is to start with the original attack game, usually called G_0, and slightly modify into a number of games satisfying the following:

- Successive games are indistinguishable from the view of the adversary and this indistinguishability is shown by relating them to the usual underlying mathematically hard problems.
- Adversary gains no information in the last game, meaning that its success probability is $1/2$.

By showing these two properties, one proves that the view of the adversary in the original game and the last game are indistinguishable which means that the success probability of the adversary in the original game is negligible.

In order to illustrate this idea, we present two different examples which are CPA-secure in the standard model.

A. Security Analysis of El Gamal Encryption Scheme under DDH Assumption

First we briefly recall the El Gamal encryption scheme.

- **Setup**: Let g be a generator of some prime order group $\mathbb{G}$. Let x be the private key and $h = g^x$ be the public key.
- **Encrypt**(m): Given a message $m \in \mathbb{G}$, a random $r \in \mathbb{G}$ is chosen and m is encrypted as:
$$c = (g^r, h^r \cdot m)$$
- **Decrypt**(c): Given a ciphertext $c = (c_1, c_2)$,
$$m = c_2 / c_1^x$$

Now, we will briefly explain the game hopping proof of the CPA-security of the El Gamal encryption scheme in the standard model.

Theorem IV.1 *If the Decisional Diffie-Hellman (DDH) problem is hard, then the El Gamal encryption scheme is IND-CPA secure in the standard model.*

Proof: We will first consider the original CPA attack game Game 0 between an adversary A and a challenger C. Game 0 proceeds as follows:

Game 0 :

1) *First, C runs the Setup algorithm and provides the public key to A.*
2) Find Stage: *A chooses two messages m_0, m_1 and sends them to C.*
3) Challenge: *C selects a random bit b and encrypts m_b as $c^* = (g^r, h^r \cdot m_b)$.*
4) Guess: *A guesses b' for b.*

Next, with a small change, we define a new game Game 1 same as Game 0 except in the challenge ciphertext C sends $c^* = (g^r, g^z \cdot m_b)$ where z is randomly chosen from $\mathbb{Z}_q$. Now, we argue that the adversary cannot distinguish its view in games Game 0 and Game 1 depending on the DDH assumption. Because note that knowing g^r and $h = g^x$ the adversary needs to distinguish $h^r = g^{xr}$ and the random value g^z which corresponds exactly the DDH decision problem. What remains is that the success probability of the attacker in Game 1 is information theoretically $1/2$ since g^z is randomly selected from a uniform distribution over $\mathbb{G}$ which makes it like a one-time pad. This concludes the proof. ■

B. Security Analysis of Hashed El Gamal under DDH Assumpiton and Entropy Smoothing

In this section we will discuss the hashed version of the El Gamal encryption scheme which we already defined in Section III-B. However, this time, the security of the scheme will be proved in the standard model under the DDH and entropy smoothing assumptions. First we define entropy smoothing.

Entropy smoothing: Let $\mathcal{H} = \{H_k\}_{k \in K}$ be a family of hash functions where each $H_k : \mathbb{G} \to \{0,1\}^\ell$. $\mathcal{H}$ is called entropy smoothing if there exists no polynomial time adversary that can effectively distinguish between the pairs $(k, H_k(x))$ and (k, h), where $k \in_R K$, $x \in_R \mathbb{G}$ and $h \in_R \{0,1\}^\ell$.

The intution behind entropy smoothness is that the range of an arbitrary hash function from the hash family is uniformly distributed.

Now we present the semantic security of hashed El Gamal encryption scheme in the standard model using game hopping techniques.

Theorem IV.2 *Under the DDH assumption and entropy smoothing property of the hash familiy, hashed El Gamal encryption scheme is CPA-secure in the standard model.*

Proof: Let Game 0 be the original CPA attack game.

Game 0 :

1) *First, C runs the Setup algorithm and provides the public key to A.*
2) Find Stage: *A chooses two messages m_0, m_1 and sends them to C.*
3) Challenge: *C selects a random bit b and encrypts m_b as $c^* = (g^r, H(h^r) \oplus m_b)$.*
4) Guess: *A guesses b' for b.*

Let Game 1 be the same as Game 0 except a small change in the challenge, $c_2 = H(g^z) \oplus m_b$ instead of $H(h^r) \oplus m_b$ where z is randomly chosen from $\mathbb{Z}_q$. Note if a polynomial time adversary can distinguish its views in Game 0 and Game 1 with a non-negligible success probability, one can construct an efficient algorithm that can decide DDH problem which contradicts the hardness assumption of DDH.

Now, let Game 2 be the same as Game 1, except that in the challenge, $c_2 = r \oplus m_b$ instead of $H(g^z) \oplus m_b$ where r is a random element of $\{0,1\}^\ell$. Note that due to the entropy smoothing property of the hash family used in the scheme, no efficient adversary can distinguish between its views in Game 1 and Game 2.

Finally, the success probability of the adversary in Game 2 is information theoretically 1/2, i.e. it behaves like a one-time pad and this concludes the proof. ■

V. Conclusion

In this paper, we have briefly discussed several security proofs that utilize the random oracle model and the game hopping technique. We hope that the ideas presented here will help the readers having a better understanding about security proofs in modern cryptography. Here, we confined ourselves to rather simple constructions and proofs, however, random oracle and game hopping techniques are two of the main building blocks of most security proofs in the public key setting.

Acknowledgment

We would like to thank Ali Aydın Selçuk and Ersan Akyıldız for useful discussions.

References

[1] N. Koblitz and A. Menezes, *Another Look at Provable Security.* J. Cryptology 20(1): pages 3-37, 2007.
[2] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations among notions of security for public-key encryption schemes.* Advances in Cryptology, CRYPTO'98.
[3] M. Bellare and P. Rogaway, *Random Oracles are practical:a paradigm for designing efficient protocols.* In First ACM Conference on Computer and Communications Security, pages 62-73, 1993
[4] R. Cramer and V. Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack.* In Proceedings of Crypto 1998, LNCS 1462
[5] D. Pointcheval, *Contemporary Cryptology: Provable Security for Public Key Schemes*, Advanced Courses CRM Barcelona, pages 133-189, 2005.
[6] S. Goldwasser and S. Micali, *Probabilistic encryption and how to play mental poker keeping secret all partial information*, Proc. 14th Symposium on Theory of Computing, pages 365-377, 1982
[7] S. Golwasser and S. Micali, *Probabilistic encryption*, Journal of Computer and System Sciences, pages 270-299, 1984
[8] R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, In proceedings of the 30thAnnual ACM Symposium on the Theory of Computing, 1998