Indian Statistical Institute, Kolkata

M. Tech. (Computer Science) Dissertation Thesis

A Study on Cryptographic Key Exchange Protocols

A dissertation submitted in partial fulfillment of the requirements
for the award of Master of Technology
in
Computer Science
July, 2016

Author:
Subhadip Singha
Roll No: CS1410

Supervisor:
Dr. Rishiraj Bhattacharyya
CSRU Unit, ISI

CERTIFICATE

This is to certify that the dissertation entitled **"A Study on Cryptographic Key Exchange Protocols"** submitted by **Subhadip Singha** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dr. Rishiraj Bhattacharyya
Cryptology and Security Research Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr. Rishiraj Bhattacharyya for the continuous support for my M.Tech dissertation and related research, for his patience, motivation, and in depth knowledge. His guidance helped me in all the time of research and writing of this thesis. It was his support that led me through to complete the anticipated work. He is open to new ideas and through out the course of my dissertation he steered me in the right direction whenever he thought I needed it.

Besides my guide, I would like to thank all the professors who taught me various subjects during last two years and I genuinely thank all my classmates who also helped me in learning and motivated me to do good research.

Subhadip Singha
M.Tech. II Year
Discipline of Computer Science
ISI Kolkata

Abstract

Key-exchange protocol are one of the interesting fields of study in Cryptography. These are the mechanisms by which two or more parties that communicate over an adversarially-controlled network can generate a common secret key. In my desertion thesis, I tried to focus on two aspects of key-exchange protocols. First is the behaviour of these protocols when they are exposed to related randomness attacks (RRA) or when the adversary partially controls the the randomness pool to be used by the parties and how to secure those protocols in these scenarios to make them useful. The second aspect is to extend two party Non-interactive Key-exchange (NIKE) protocols to three party in standard model extending the underlying model with proper security bound.

Contents

Chapter 1

Introduction

1.1 Introduction

Key-exchange protocols are mechanisms by which two parties that communicate over an adversarially-controlled network can generate a common secret key. Key-exchange protocols are essential for enabling the use of shared-key cryptography to protect transmitted data over insecure networks. As such they are a central piece for building secure communications or secure channels. The most commonly used cryptographic protocols include SSL, IPSec, SSH, etc. Design and analysis of Key exchange protocols has been proved to be non trivial with lots of work done on this topics, such as [6], [4], [3], [5], [2].

1.2 Foundation

Cryptographic key exchange protocol is a method of securely exchanging cryptographic keys over a public channel. The first key exchange protocol was originally conceptualized by Ralph Merkle and named after Whitfield Diffie and Martin Hellman as DiffieHellman (DH) [1] Key exchange protocol. DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical channel, such as paper key lists transported by a trusted courier. The DiffieHellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

In 2002, Hellman suggested the algorithm be called DiffieHellmanMerkle key exchange in recognition of Ralph Merkle's contribution to the invention of public-key cryptography.

Overview of Diffie-Hellman Key Exchange Protocol

Diffie-Hellman Key Exchange establishes a shared secret between two parties that can be used for secret communication for exchanging data over a public network.

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. These two values are chosen in this way to ensure that the resulting shared secret can take on any value from 1 to $p-1$.

1. Alice chooses a secret integer a, then sends Bob $A = g^a \bmod p$
2. Bob chooses a secret integer b, then sends Alice $B = g^b \bmod p$
3. Alice calculates secret key as $s_1 = B^a \bmod p = g^{ab} \bmod p$
4. Alice calculates secret key as $s_2 = A^b \bmod p = g^{ba} \bmod p$

Both Alice and Bob have arrived at the same value $s = s_1 = s_2$, under mod p, Note that only a, b, and ($g^{ab} \bmod p = g^{ba} \bmod p$) are kept secret. All the other values $p, g, g^a \bmod p, andg^b \bmod p$ are sent in the clear. Once Alice and Bob compute the shared secret they can use it as an encryption key, known only to them, for sending messages across the same open communications channel.

1.3 Varieties of Key Exchange Protocol

Key exchange protocols can be of two types.

1. Interactive Key Exchange Protocol
2. Non-Interactive Key Exchange Protocol

Interactive Key Exchange Protocol

The fundamental approach to key exchange protocols is interactive.Interactive Key-exchange protocols are mechanisms by which two parties that communicate over an adversarially-controlled network can generate a common secret key by interacting with each others. Following are the few concepts which are central to the idea of interactive key exchange protocol. Here are the few basic notions related to interactive key exchange protocol.

Protocols : We consider a set of parties (probabilistic polynomial-time machines), which we usually denote by $P_1, ..., P_n$, interconnected by point-to-point links over which messages can be exchanged. Protocols are collections of interactive procedures, run concurrently by these parties, that specify a particular processing of incoming messages and the generation of outgoing messages. Protocols are initially triggered at a party by an external " call " and later by the arrival of messages. Upon each of these events, and according to the protocol specification, the protocol processes information and may generate and transmit a message and/or wait for the next message to arrive. We call these message-driven protocols.

Sessions : We call each copy of a protocol run at a party a session. Technically, a session is an interactive subroutine executed inside a party. Each session is identified by the party that runs it, the parties with whom the session communicates and by a session-identifier.

Key-exchange protocols : Interactive Key-exchange protocols are message-driven protocols where the communication takes place between pairs of parties and which return, upon completion, a secret key called a session key.

More specifically, the input to a interactive key exchange protocol within each party P_i is of the form $(P_i, P_j, s, role)$, where P_j is the identity of another party, s is a session id, and role can be either initiator or responder. A session within P_i and a session within P_j are called matching if their inputs are of the form $(P_i, P_j, s, initiator)$ and $(P_j, P_i, s, responder)$. The inputs are chosen by a higher layer protocol that calls the interactive key exchange protocol. We require the calling protocol to make sure that the session id's of no two sessions in which the party participates are identical.

Non-Interactive Key Exchange Protocol

Non-interactive key exchange (NIKE) is a cryptographic primitive which enables two parties, who know each others public keys, to agree on a symmetric shared key without requiring any interaction. The canonical example of a NIKE scheme can be found in the seminal paper by Diffie and Hellman [1].

For example, let G be a group of prime order p with generator g, and assume that Alice has public key $g^x \in G$ and private key $x \in Z_p$, while Bob has public key $g^y G$ and private key $y \in Z_p$. Then Alice and Bob can both compute the value $g^{xy} \in G$ without exchanging any messages. More properly, Alice and Bob should hash this key together with their identities in order to derive a symmetric key $H(Alice, Bob, g^{xy})$ where H is a cryptographic hash function.

1.4 Our Result

In this thesis, we have looked into the following

1. We formalize the security model for a key exchange protocol to be secure against Related Randomness Attacks (RRA).
2. We also studied the Three party NIKE protocol, secure in standard model.

Chapter 2

Interactive Key Exchange Protocol

In this section we will talk about security models of different types of key exchange protocols.

2.1 Introduction and Preliminaries

We follow the model of Canetti and Krawczyk [8], which has been considered as a standard definition of key exchange protocols.

The unauthenticated-link adversarial model (UM)

In order to talk about the security of a protocol we need to define the adversarial setting that determines the capabilities and possible actions of the attacker. We want these capabilities to be as generic as possible while not posing unrealistic requirements. We call this model the Unauthenticated Links Model (UM).

Basic attacker capabilities: We consider a probabilistic polynomial-time (ppt) attacker that has full control of the communications links: it can listen to all the transmitted information, decide what messages will reach their destination and when, change these messages at will or inject its own generated messages. The formalism represents this ability of the attacker by letting the attacker be the one in charge of passing messages from one party to another. The attacker also controls the scheduling of all protocol events including the initiation of protocols and message delivery. In addition to these basic adversarial capabilities, we let the attacker obtain secret information stored in the parties memories via explicit attacks. we classify attacks into three categories depending on the type of information accessed by the adversary.

2.2 Different Classes of Attacks

Session-State Reveal : The attacker provides the name of a party and a session identifier of a yet incomplete session at that party and receives the internal states of that session. The information, included in the local state of a session, is specified by each key exchange protocol. Therefore, our definition of security is parameterized by the type and amount of information revealed in this attack. Typically, the revealed information will include all the local state of the session and its subroutines, except for the local state of the subroutines that directly access the long-term secret information, e.g. the local signature/decryption key of a public-key cryptosystem, or the long-term shared key.

Session-Key Query : In this attack, the attacker provides a party's name and a session identifier of a completed session at that party and receives the value of the key generated by the named session. This attack provides the formal modeling for leakage of information on specific session keys that may result from events such as break-ins, cryptanalysis, careless disposal of keys, etc. It will also serve, indirectly, to ensure that the unavoidable leakage of information produced by the use of session keys in a security application (e.g., information leaked on a key by its use as an encryption key) will not help in deriving further information on this and other keys.

Party corruption : The attacker can decide at any point to corrupt a party, in which case the attacker learns all the internal memory of that party including long-term secrets (such as private keys or master shared keys used across different sessions) and session-specific information contained in the party's memory (such as internal state of incomplete sessions and session-keys corresponding to completed sessions). Since by learning its long term secrets the attacker can impersonate a party in all all its actions then a party is considered completely controlled by the attacker from the time of corruption and can, in particular, depart arbitrarily from the protocol specifications.

2.3 Security Model

Here we discuss a new security models of key exchange protocol in authenticated-link adversarial model (AKE). An AKE protocol consists of two probabilistic polynomial time algorithms: the Long-Lived Key generation algorithm SKG and a protocol execution algorithm P. Here we focus on the public key setting where the algorithm SKG returns a public key and a private key upon each invocation. An adversary can make the following oracle queries.

Register(U, pk_U) **:** This oracle query allows the adversary A to register a new user U with public key pk_U . Here we only require that neither the user identity U nor the public key pk_U exists in the system. In particular, we do not require the adversary to provide a proof of knowledge on the secret key with regard to pk_U .

NewInstance(U, i, N) **:** This oracle query allows A to initialize a new instance π_U^i within party U with a binary string N which serves as the random tape of π_U^i.

Send(U, i, M_{in}) **:** This oracle query invokes instance i of U with message M_{in}. The instance then runs the protocol and sends the response back to the adversary.

Reveal(U, i) **:** If oracle π_U^i has accepted and generated a session key ssk_U^i, then ssk_U^i is returned to the adversary.

Corrupt(U) **:** By making this oracle query, adversary A obtains the long-lived secret key sk_U of party U.

Test(U, V, sid) **:** A new session is created between party U and V and a session key is generated ssk. A coin b is flipped. if $b = 1$, the adversary is given the actual session key ssk. If $b = 0$, then a random session key is drawn from the session key space and returned to the adversary. This query is only asked once during the whole game. The success of an adversary is measured by its ability to distinguish a real session key from a random key in the session key space.

Definition of Session Key Security : A KE protocol P is called (Session Key) SK-secure if the following properties hold for any KE-adversary A in the UM.

1. Protocol P satisfies the property that if two uncorrupted parties complete matching sessions then they both output the same key.
2. the probability that any adversary A guesses correctly the bit b (i.e., outputs $b' = b$) is no more than 1/2 plus a negligible fraction in the security parameter.

If the above properties are satisfied for all KE-adversaries in the UM then we say that P is SK-secure.

2.4 Security Model under Reset randomness

In some practical situations however, the randomness may be controlled by an adversary and the seeds may no longer be fresh or truly random. For example, if an adversary has physical access to a hardware source or may be able to manipulate the randomness used in Key exchange protocols.

Adversarial reset of machines could make an AKE protocol reuse the same random coins in different sessions. In [12] these threats have been properly modelled.

Reset Model : We need to perform some modification to the Test query, Adversary A selects two parties U and V which had successfully completed few sessions with each other. These parties are asked to create a new session and use the randomness that they had already used in one of their completed sessions. So, NewInstance($U, i, N1$) and NewInstance($V, j, N2$) are called. Here both the parties choose binary string or the random tape N_i randomly from the previously completed sessions. Here, the adversary does not control the randomness directly but the randomness is reused. After completion of the session one of the parties is chosen. A coin b is flipped. if $b = 1$, the adversary is given the actual session key. If $b = 0$, then a random session key is drawn from the session key space and returned to the adversary. This query is only asked once during the whole game. The success of an adversary is measured by its ability to distinguish a real session key from a random key in the session key space.

2.5 Brief Review of existing work

In this section, we recall some of the important key-exchange protocols proposed in the literature.

Two-move Diffe-Hellman(2DH) : Common information: Primes $p, q, q/p-1$, and$g$ of order q in Z_p^*.

1. The initiator, P_i, on input (P_i, P_j, s), chooses $x \xleftarrow{\$} Z_q$ and sends $(P_i, s, \alpha = g^x)$ to P_j.

2. Upon receipt of (P_i, s, α) the responder, P_j chooses $y \xleftarrow{\$} Z_q$, sends $(P_j, s, \beta = g^y)$ to P_i, erases y, and outputs the session key $\gamma = \alpha^y$ under session-id s.

3. Upon receipt of (P_j, s, β), party P_i computes $\gamma' = \beta^x$, erases x, and outputs the session key γ' under session-id s.

Theorem : Assuming the Decisional Diffie-Hellman (DDH) assumption, protocol 2DH is SK-secure in the AM.

Signature based Diffe-Hellman(2DH) : Common information: Primes $p, q, q/p-1$, and$g$ of order q in Z_p^*. Each player has a private key for a signature algorithm sig, and all have the public verification keys of the other players.

1. The initiator, P_i, on input (P_i, P_j, s), chooses $x \xleftarrow{\$} Z_q$ and sends $(P_i, s, \alpha = g^x)$ to P_j.

2. Upon receipt of (P_i, s, α) the responder, P_j , chooses $y \xleftarrow{\$} Z_q$, sends $(P_j, s, \beta = g^y)$ to P_i, erases y together with the signature $sig_j(P_j, s, \beta, \alpha, P_i)$, and outputs the session key $\gamma = \alpha^y$ under session-id s.

3. Upon receipt of (P_j, s, β) and P_j's signature, party P_i verifies the signature and the correctness of the values included in the signature (such as players identities, session id etc.). If the verification succeeds then P_i sends to P_j the message $(P_i, s, sig_j(P_i, s, \alpha, \beta, P_j))$, $\gamma' = \beta^x$, erases x, and outputs the session key γ' under session-id s.

4. Upon receipt of (P_j, s, sig), P_j verifies P_i's signature sig and the values it includes. If the check succeeds it outputs the session key γ under session-id s.

The signature-based Diffe-Hellman paradigm has been used to design many popular AKE protocols, such as the ISO protocol [7], the SIGMA (SIGn-and-MAc) [10] and JFK (Just Fast Keying) [9]. These protocols are all proven secure in the SK model.

2.6 Proposed Model (Related Randomness)

Our Proposed model is same as before the only difference is in the adversarial capabilities. Here adversary can provide an index to choose a randomness from the randomness pool and it can also choose a function that will be used on that chosen randomness to get of a new randomness to be used to build the shared key in any session. If we try to figure out the connection between our proposed model and Reset model, we'll find that when the function is an identity function and the chosen index is an used one for the set of parties, we are in Reset model. Other than this situation the function can be any arbitrary one, hence here we capture capabilities of more powerful adversary. We need to figure out the pool of functions $\phi \in \Phi$ for which adversary can trivially break the Key exchange protocol. We'll consider key exchange protocol of the form of $A = (SKG, P)$, where SKG is key generation algorithm of users (public key and private key of a party) and P is a shared key generation algorithm.

The Register, New Instance, Send, Reveal and Corrupt queries will be same as before but we need to define Test query for our related randomness model.

Test$(U, V, sid, \phi_1, \phi_2, i_1, i_2)$ **:** By making this oracle query, Adversary A selects two parties U and V. These parties are asked to create a new session. So, NewInstance(U, $i, \phi_1(N_{i_1})$) and NewInstance($V, j, \phi_2(N_{i_2})$) are called. After completion of the session one of the parties is chosen. A coin b is flipped. if $b = 1$, the adversary is given the actual session key. If $b = 0$, then a random session key is drawn from the session key space and returned to the adversary. This query is only asked once during the whole game. The success of an adversary is measured by its ability to distinguish a real session key from a random key in the session key space.

We formalize the Related Randomness security of a key exchange protocol through the following subroutine used during the Test query. CoinTab is a table which contains the random strings used in the past sessions.

proc.Test$(ID_1, ID_2, \phi_1, \phi_2, i_1, i_2)$**:**
If CoinTab$[i_1] = \perp$ then
CoinTab$[i_1] = \xleftarrow{\$} Rnd$

If CoinTab$[i_2] = \perp$ then
CoinTab$[i_2] = \xleftarrow{\$} Rnd$

$r_{i_1} \leftarrow$ CoinTab$[i_1]$
$r_{i_2} \leftarrow$ CoinTab$[i_2]$

$c = P(\phi_1(r_{i_1}), \phi_2(r_{i_2}))$
$b \xleftarrow{\$} \{0, 1\}$
if $b = 1$ return c.
else $c \xleftarrow{\$} Keys_\lambda$
else return c.

We define the RRA advantage of an adversary A against a protocol P as

$$Adv_{P,A}^{rra-atk}(\lambda) = |Pr_{[P,A]}[b = b'] - 1/2|$$

2.6.1 Attack

Here we show that key exchange schemes which are secure in Reset-1 and Reset-2 model can be easily attacked in our proposed Related Randomness Model, showing the necessity of a new construction.

Attack against PKEDH-R2 : Consider a past session i of a user U with the user V session j. Let $ssk \leftarrow$ SessionReveal(U, i). Next, the adversary runs the test query

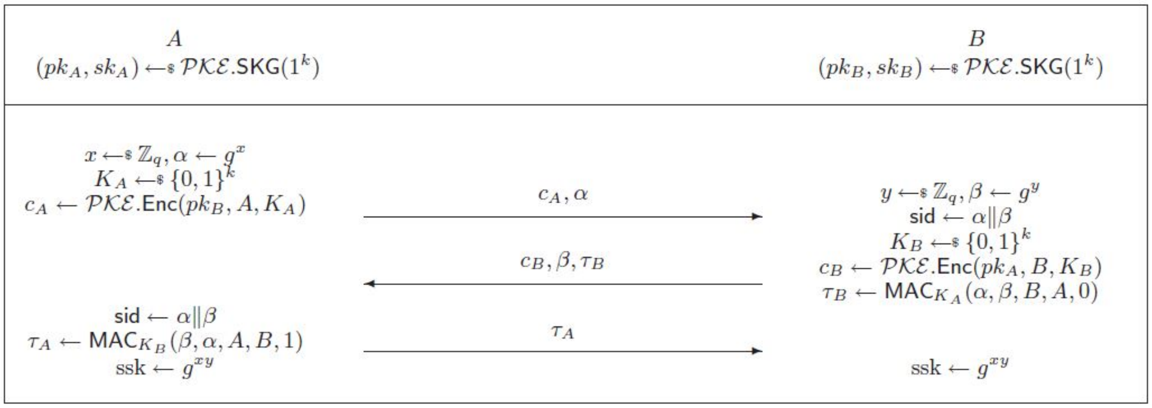

Figure 2.1: PKEDH-R2

Test$(U, V, \phi_1, \phi_2, i, j)$ where $\phi_1(x) \stackrel{def}{=} 2x$ and $\phi_2(x) \stackrel{def}{=} x/2$. Let K' be the response. The adversary outputs 1 if $K' = ssk$. Indeed, the session key generated during the test query is ssk.

2.6.2 Preliminaries

Here we define some necessary to tools which are required for the security proof of our protocol. We follow the definitions according to [14]

PRFReal Game

proc.Initialise(λ):
$K \xleftarrow{\$} Keys_\lambda$

proc.Function(x):
Retrun $F(K, x)$.

proc.Finalise(b):
Retrun b.

PRFRand Game

proc.Initialise(λ):
FunTab $\leftarrow \phi$

proc.Function(x):
If FunTab$[x] = \perp$ then
FunTab$[x] = \xleftarrow{\$} Rng_\lambda$

Return FunTab[x].

proc.Finalise(b):
Retrun b.

Pseudorandom Functions :

Let $F : Keys_\lambda \times Dom_\lambda \rightarrow Rng_\lambda$ be a family of functions. The advantage of a RKA-PRF adversary A against F is

$$Adv^{prf}_{F,A}(\lambda) = Pr[PRFReal^A_F(\lambda) \Rightarrow 1] - Pr[PRFRand^A_{\$}(\lambda) \Rightarrow 1]$$

We say F is a secure PRF family if the advantage of any polynomial-time adversary is negligible in the security parameter λ.

RKA-PRFReal Game

proc. Initialise(λ):
$K \xleftarrow{\$} Keys_\lambda$

proc. Function(ϕ, x):
Retrun $F(\phi(K), x)$.

proc. Finalise(b):
Retrun b.

RKA-PRFRand Game

proc. Initialise(λ):
$G \leftarrow FF(Keys_\lambda, Dom_\lambda, Rng_\lambda)$
$K \xleftarrow{\$} Keys_\lambda$

proc. Function(x):
Return $G(\phi(K), x)$.

proc. Finalise(b):
Retrun b.

Related Key Secure Pseudorandom Functions :

Let $F : Keys_\lambda \times Dom_\lambda \to Rng_\lambda$ be a family of functions. The advantage of a PRF adversary A against F is

$$Adv_{F,A}^{rka-prf}(\lambda) = Pr[RKA - PRFReal_F^A(\lambda) \Rightarrow 1] - Pr[RKA - PRFRand_\$^A(\lambda) \Rightarrow 1]$$

We say F is a secure Φ-RKA-PRF family if the advantage of any Φ-restricted, polynomial-time adversary is negligible in the security parameter λ.

2.6.3 Construction of RRA secure KE Protocol

Given a key exchange protocol KE = (SKG, P) that is secure in public key settings, and a Φ restricted pseudorandom function family $F = \{F_r : \{0,1\}^{\delta(k)} \to \{0,1\}^{\rho(k)} | K \in \{0,1\}^{\delta(k)}\}$, where $\rho(k)$ and $\delta(k)$ are polynomials of k. We construct a new protocol KE′ = (SKG′, P′) as follows:

1. SKG′(1^k): run SKG(1^k) to generate (pk, sk), select $K \xleftarrow{\$} \{0,1\}^{\delta(k)}$. Set $pk' = pk$ and $sk' = (sk, K)$.

2. P′: get a $\rho(k)$ bit random string r, then compute $r' \xleftarrow{\$} F_r(K)$ and run P with random coins r'.

Theorem 1 *Suppose A is a Φ-restricted adversary in the RRA-KE game against the scheme PRF-KE defined as per our construction. Suppose A makes q_{LR} proc.Test queries. Then there exists a Φ-restricted RKA-PRF adversary B and an IND-ATK-KE adversary C such that*

$$Adv_{PRF-KE,A}^{rra-atk}(\lambda) \leq q_{LR}.Adv_{PKE,C}^{ind-atk-ke}(\lambda) + 2/q_L Adv_{F,B}^{rka-prf}(\lambda)$$

Proof:

Let G_0 be the real RRA-ATK-KE security game played by an adversary A against the challenger correctly simulating the scheme PRF-KE and let G_2 be the game where outputs of the PRF F are replaced with values chosen uniformly at random.

We define an intermediate game G_1, where the challenger during the test query simulation replaces $F_{\phi_1(r_i)}(K_U)$ by a uniform random string r'_U. We claim that there is an adversary B against the Φ-RKA-PRF security of F such that:

$$|Pr[G_0^A = 1] - P[G_1^A = 1]| \leq 1/q_L Adv_{F,B}^{rka-prf}(\lambda)$$

where q_L is the number of *reveal* query of A.

The adversary B to the RKA-PRF security of F works as follows. Adversary B creates an instance of KE and runs A. Moreover, B chooses an i uniformly at random from $[q_L]$

When A submits the i^{th} reveal query for (ID', ID''), B queries $F_{\mathtt{id}}(K)$ and uses the returned string to execute a session between ID' and ID''. For other reveal queries, B chooses r on its own, and computes $F_r(K)$ to compute the required randomness. To simulate Corrupt query B outputs the corresponding secret key, sampled by B during the instance creation.

When A submits a test query $(.,.,\phi_1,.,i_1)$, B checks whether $i = i_1$. If $i \neq i_1$, B aborts. Otherwise, B uses its RKA oracle to get the required $r' = F_{\phi_1(r_i)}(K)$. Hence,conditioned on $i_1 = i$

$$|Pr[G_0^A = 1] - P[G_1^A = 1]| = |Pr_{r' \leftarrow_R \mathcal{R}}[B[r'] = 1] - Pr_{r' = F_{\phi_1(r_i)}(K)}[B[r'] = 1]| \leq Adv_{F,B}^{rka-prf}(\lambda).$$

Now, using $Pr[i_1 = i] = 1/q_L$, we get

$$|Pr[G_0^A = 1] - P[G_1^A = 1]| \leq 1/q_L Adv_{F,B}^{rka-prf}(\lambda)$$

By symmetry, we get

$$|Pr[G_1^A = 1] - P[G_2^A = 1]| \leq 1/q_L Adv_{F,B}^{rka-prf}(\lambda)$$

Hence, by triangle inequality,

$$|Pr[G_0^A = 1] - P[G_2^A = 1]| \leq 2/q_L Adv_{F,B}^{rka-prf}(\lambda)$$

Hence,

$$Adv_{PRF-KE,A}^{rra-atk}(\lambda) \leq q_{LR}.Adv_{PKE,C}^{ind-atk-ke}(\lambda) + 2/q_L Adv_{F,B}^{rka-prf}(\lambda)$$

$\square$

Chapter 3

Non-Interactive Key Exchange Protocol

Non-interactive key exchange (NIKE) is a cryptographic primitive which enables two parties, who know each others public keys, to agree on a symmetric shared key without requiring any interaction. The canonical example of a NIKE [13] scheme can be found in the seminal paper by Diffie and Hellman [1].

3.1 Basic Definitions

Non-Interactive Key Exchange (NIKE) scheme in the public key setting is collection of three algorithms: CommonSetup, KeyGen and SharedKey together with an identity space IDS and a shared key space SHK.

Common Setup : On input 1^k, outputs $params$, a set of system parameters.

KeyGen : On input $params$ and an identity $ID \in IDS$, outputs a public key/secret key pair (pk, sk). This algorithm is probabilistic and can be executed by any user. We assume without loss of generality, that $params$ is included in pk.

SharedKey : On input an identity $ID_1 \in IDS$ and a public key $pk1$ along with another identity $ID_2 \in IDS$ and a secret key sk_2, outputs either a shared key in SHK for the two identities, or a failure symbol $\perp$. This algorithm is assumed to always output $\perp$ if $ID_1 = ID_2$.

For correctness, we require that, for any pair of identities ID_1, ID_2, and corresponding key pairs (pk_1, sk_1) and (pk_2, sk_2), algorithm SharedKey satisfies the constraint:

$$\text{SharedKey}(ID_1, pk_1, ID_2, sk_2) = \text{SharedKey}(ID_2, pk_2, ID_1, sk_1)$$

3.2 Security Model

We work on the security model proposed by Cash, Kiltz and Shoup, or in short CKS model [11]. The power of an adversary is modeled through four function calls.

Register honest user ID : A supplies an identity $ID \in IDS$. On input *params* and ID, the challenger runs KeyGen to generate a public key/secret key pair (pk, sk) and records the tuple $(honest, ID, pk, sk)$. The challenger returns pk to A.

Register corrupt user ID : In this type of query, A supplies both an identity $ID \in IDS$ and a public key pk. Challenger records the tuple $(corrupt, ID, pk,)$. We stress that A may make multiple Register corrupt user ID queries for the same ID during the experiment. In that case, only the most recent $(corrupt, ID, pk, \perp)$ entry is kept.

Extract : Here A supplies an identity ID that was registered as an honest user. The challenger looks for a tuple $(honest, ID, pk, sk)$ containing ID and returns sk to A.

Reveal : Here A supplies a pair of registered identities ID_1, ID_2, subject only to the restriction that at least one of the two identities was registered as honest. The challenger runs SharedKey using the secret key of one of the honest identities and the public key of the other identity and returns the result to A. Note that here the adversary is allowed to make reveal queries between two users that were originally registered as honest users. Honest reveal the queries are those which involve two honest users and corrupt reveal queries involve an honest user and a corrupt user.

Test : Here A supplies two distinct identities ID_1, ID_2 that were both registered as honest. The challenger returns $\perp$ if $ID_1 = ID_2$. Otherwise, it uses the bit b to answer the queries. If $b = 0$, the challenger runs SharedKey using the public key for ID_1 and the secret key for ID_2 and returns the result to A. If $b = 1$, the challenger generates a random key, records it for later, and returns that key to the adversary. In this case, to keep things consistent, the challenger returns the same random key for the pair ID_1, ID_2 every time A queries for their paired key, in either order.

A's queries may be made adaptively and are arbitrary in number. To prevent trivial wins for the adversary, no query to the reveal oracle is allowed on any pair of identities selected for test queries (in either order), and no extract query is allowed on any of the identities involved in test queries. Also, we demand that no identity registered as corrupt can later be the subject of a register honest user ID query, and vice versa.

When the adversary finally outputs b', it wins the game if $b' = b$. For an adversary A, we define its advantage in this security game as:

$$Adv_A^{CKS}(k) = |Pr[b' = b] - 1/2|$$

Modified CKS Model

We extend the original CKS model for three party NIKE. We have three different models as CKS-light, CKS-heavy, m-CKS-heavy other than CKS itself. We'll give the adversarial capabilities for each of these models.

Model	Register Honest	Register Corrupt	Extract	Honest Reveal	Corrupt Reveal	Test
CKS-light	3	✓	✗	✗	✓	1
CKS	✓	✓	✗	✗	✓	✓
CKS-heavy	✓	✓	✓	✓	✓	1
m-CKS-heavy	✓	✓	✓	✓	✓	✓

Table 3.1: Adversarial capabilities in different models

Notation : ✓ means that an adversary is allowed to make an arbitrary number of queries and ✗ means that no query can be made, numbers represent the number of queries allowed to an adversary.

New Constraints : In the three party model the function calls which remain same are Register honest user ID, Register corrupt user ID and Extract. The nature of the Reveal query remains the same, except that adversary will provide three identities and among them at least two identities should be registered as honest. In case of the Test query the adversary has to provide three honest identities in place of two. The nature of the function call remains the same.

Theorem : The m-CKS-heavy, CKS-heavy, CKS and CKS-light security models are all polynomially equivalent. We'll provide the proof in the later section.

The Decisional Bilinear Diffe-Hellman Assumption (DBDH)

Our pairing based scheme will be parameterized by a Type 1 pairing parameter generator. This is a polynomial time algorithm that on input a security parameter 1^k, returns the description of two multiplicative cyclic groups G_1, and G_T of the same prime order p, generator g_1, for G_1, and a bi-linear non-degenerate and efficiently computable pairing $e : G_1 \times G_1 \to G_T$. Throughout, we write $PG = (G_1, G_T, g_1, p, e)$ for a set of groups and other parameters with the properties just described.

We consider the following version of the Decisional Bi-linear Diffie-Hellman problem for type 1 pairings: Given $(g_1, g_1^a, g_1^b, g_1^c, T) \in G_1^4 \times G_T$

we associate the following experiment to a Type 1 pairing parameter generator G1 and an adversary B.

Experiment$\text{Exp}_{B,G1}^{dbdh}$

$PG \xleftarrow{\$} G1(1^k)$
$a, b, c, d \xleftarrow{\$} Z_p$
$\beta \xleftarrow{\$} \{0, 1\}$
If $\beta = 1$ then $T \leftarrow e(g_1, g_1)^{abc}$ else $e(g_1, g_1)^z$
$\beta' \xleftarrow{\$} B(1^k, PG, g_1^a, g_1^b, g_1^c, T)$
If $\beta = \beta'$ 0 then return 0 else return 1

The advantage of B in the above experiment is defined as

$$Adv_{B,G1}^{dbdh}(k) = |Pr[Exp_{B,G1}^{dbdh}(k) = 1] - \frac{1}{2}|$$

3.3 Our Result

Construction of Three User Key Exchange Protocol in Standard Model

We construct a 3 user NIKE scheme, NIKE3USER, that is secure in the CKS-light security model under the DBDH assumption in the standard model. Our construction makes use of a tuple $PG = (G_1, G_T, g_1, p, e)$, output by a parameter generator $G1$, and a chameleon hash function $ChamH : \{0, 1\}^* \times RCham \to Z_p$. The component algorithms of the scheme NIKE3USER are defined as follows:

CommonSetup(1^k) **:**

$PG \xleftarrow{\$} G1\{1\}^k$,
where $PG = (G_1, G_T, g_1, p, e)$
$u_0, u_1, u_2, u_3 \xleftarrow{\$} G_1^*$,
$hk, ck \xleftarrow{\$} Cham.KeyGen(1^k)$

$params \leftarrow (PG, u_0, u_1, u_2, u_3, hk)$
Return $param$

KeyGen$(params, ID)$ **:**

$x \xleftarrow{\$} Z_p, r \xleftarrow{\$} R_{Cham}$
$Z \leftarrow g_1^x$
$t \leftarrow ChamH_{hk}(Z||ID; r)$
$Y \leftarrow u_0 u_1^t u_2^{t^2} u_3^{t^3}; X \leftarrow Y^x$
$pk \leftarrow (X, Z, r), sk \leftarrow x$
Return (pk, sk)

SharedKey$(ID_1, pk_1, ID_2, pk_2, ID_3, sk_3)$ **:**

If $ID_i = ID_j$ where $i \neq j$ return $\perp$
Parse pk_1 as (X_1, Z_1, r_1), pk_2 as (X_2, Z_2, r_2) and sk_3 as x_3
$t_1 \leftarrow ChamH_{hk}(Z_1||ID_1; r_1)$ and $t_2 \leftarrow ChamH_{hk}(Z_2||ID_2; r_2)$
If $e(X_1, g_1) \neq e(u_0 u_1^{t_1} u_2 t_1^2 u_3^{t^3}, Z_1)$ OR $e(X_2, g_1) \neq e(u_0 u_1^{t_2} u_2 t_2^2 u_3^{t^3}, Z_2)$
then $K_{1,2,3} \leftarrow \perp$
else $K_{1,2,3} \leftarrow \{e(Z_1, Z_2)\}^{x_3}$
Return $K_{1,2,3}$

The check in the SharedKey algorithm for valid public keys can be implemented by evaluating the bilinear map twice. It is clear that SharedKey defined in this way satisfies the requirement that entities ID_1, ID_2 and ID_3 are able to compute a common key. To see this, note that $\{e(Z_1, Z_2)\}^{x_3} = e(g_1, g_1)^{x_1 x_2 x_3}$. We will prove the above NIKE3USER scheme to be secure under the DBDH assumption.

Relationships between NIKE3USER Security Models

We show that the NIKE3USER security models discussed, are polynomially equivalent to each other.

Theorem 2 (CKS-light $\iff$ CKS) *A NIKE scheme $NIKE3USER$ is secure in the CKS model if and only if it is also secure in the CKS-light model. In more detail, for any adversary A against NIKE3USER in the CKS model, there is an adversary B that breaks NIKE3USER in the CKS-light model with*

Proof:

Clearly, security in the CKS model implies security in the CKS-light model as

$$Adv_A^{CKS}(k, 2, q_C, q_{CR}, 1) = Adv_B^{CKS-light}(k, q_C, q_{CR})$$

Hence we concentrate on the other side of the proof which is non trivial. We assume that there exists an adversary A against NIKE3USER in the CKS model with advantage

$$Adv_A^{CKS}(k, q_H, q_C, q_{CR}, q_T) = |Pr[b' = b] - 1/2|$$

We consider a sequence of games $G_0, G_1, , , , G_{qT}$ all defined over the same probability space. Starting from the actual adversarial game G_0 (attack game with respect to an adversary A against NIKE3USER in the CKS model), when $b = 1$ (that is, test queries will always be answered with random keys), we make slight modifications between successive games, in such a way that the adversary's view is still indistinguishable among the games. The last game, Game G_{qT}, will be exactly like Game G_0, except that this time A's challenger will use $b = 0$ to answer A's test queries. Note that this means that A can distinguish games Game G_0 and Game G_{qT} with advantage $Adv_A^{CKS}(k, q_H, q_C, q_{CR}, q_T) = |Pr[A(G_0) = 1] - Pr[A(G_{qT}) = 1]|$. We write $A(G_i)$ to denote adversary A playing in game G_i. For every $0 \leq i \leq qT$, we define a hybrid variable H^i where the first i elements are the actual shared keys associated to the corresponding users involved in the first i test queries, and the $qT - i$ following elements are random keys.

Game G_0, be the original game as described in the CKS security model when $b = 1$. Game G_i $(1 \leq i \leq qT)$. This game is identical to game Game G_{i-1}, except that whenever A makes its i^{th} Test query on a tuple of three identities, say ID_A, ID_B and ID_C, A's challenger will return to A the actual shared key, $K_{(ID_A, ID_B, ID_C)}$, between those identities. Note that Games G_i and G_{i+1} differ in only one single test query.

Now, we construct an adversary B against NIKE3USER in the sense of the CKS-light model. B plays the CKS-light security game with challenger C and acts as a challenger for A. C takes as input the security parameter 1^k, runs algorithm CommonSetup of the NIKE3USER scheme and gives B $params$. C then takes a random bit b and answers oracle queries for B until B outputs a bit b'

Let qT and qH be bounds on the number of test queries and register honest user ID queries, respectively, made to B by A in the course of its attack. Without loss of generality, we assume that the qT test queries are all distinct. B chooses a random $i \in \{0, ...qT - 1\}$ and three distinct indices I, J and K uniformly at random from $\{1, 2, 3, ..., qH\}$. Effectively B is guessing that the I-th, J-th and K-th identities to be honestly registered by A will be involved in the $(i + 1)$-st test query made by A. A makes a series of queries which B answers as follows:

Register corrupt user ID : If A makes a Register corrupt user ID query, supplying (ID, pk), then B makes the same register corrupt user ID query to C. C records the tuple $(corrupt, ID, pk, \perp)$.

Register honest user ID : Here A supplies a string ID to B. If this is the I-th or J-th or K-th such query, then B makes the same register honest user ID query to C, setting $ID_I = ID$ or $ID_J = ID$ or $ID_K = ID$ as appropriate. On input $params$ and ID, C runs KeyGen, generating a key pair (pk, sk), records $(honest, ID, pk, sk)$ and returns pk to B. If $ID \notin \{ID_I, ID_J, ID_K\}$ then B generates a key pair (pk, sk) by running algorithm KeyGen on input $params$ and ID, and makes a Register corrupt user ID query to C on inputs the string ID and the public key pk. B then gives pk to A.

Corrupt reveal : Whenever A supplies three identities ID, ID', ID'', where ID was registered by A as corrupt and $ID' and ID''$ were registered as honest, B will check if $ID' \in \{ID_I, ID_J, ID_K\}$ or $ID'' \in \{ID_I, ID_J, ID_K\}$. If so, B will make the same corrupt reveal query to C, obtaining $K_{(ID,ID',ID'')}$, and give the result to A else, B runs SharedKey on input $(ID, pk_{ID}, ID', sk_{ID'}, ID'', pk_{ID''})$. Note that in this case, B has $sk_{ID'}$ because it generated for itself the pair $(pk_{ID'}, sk_{ID'})$. B gives $K_{(ID,ID',ID'')}$ to A.

Test : B will answer A's first i Test queries with the actual shared keys associated to the corresponding users involved in those test queries, the $(i+1)^{st}$ test query with a value that can be either the actual shared key associated to the users involved in that $test$ query or a random value, and the other $qT - i - 1$ Test queries with random values. Next, we explain in more detail exactly how B handles A's Test queries.

When A makes its j^{th} $(j \leq i)$ Test query on a tuple of identities $\{ID, ID', ID''\}$, that were registered as honest users, B will check if $\{ID, ID', ID''\} = \{ID_J, ID_J, ID_K\}$. If so, B aborts the simulation. Otherwise, suppose $|\{ID, ID', ID''\} \cap \{ID_J, ID_J, ID_K\}| \leq 2$. B gives $K_{(ID,ID',ID'')}$ to A.

When A makes its $(i+1)^{st}$ Test query on a tuple of identities $\{ID, ID', ID''\}$, B checks if $\{ID, ID', ID''\} = \{ID_J, ID_J, ID_K\}$. If not, B aborts the simulation. If $\{ID, ID', ID''\} = \{ID_J, ID_J, ID_K\}$, B makes the same Test query to C receiving α. B gives α to A. For all other Test queries B will respond with a random value.

Whenever A terminates by outputting a bit b', then B outputs the same bit. Now, if α is the actual key $K_{(ID_A,ID_B,ID_C)}$ associated to (ID_A, ID_B, ID_C) (the identities involved in the $(i+1)^{st}$ Test query made by A), then A was playing game Game G_{i+1}. Otherwise, if α is a random value, A was playing game G_i.

Let G_0' and G_1' be the games played by B against NIKE3USER in the CKS-light model when $b = 0$ and $b = 1$, respectively. Let F denote the event that B is not forced to abort during its simulation. So, $Pr(F) \geq 1/\binom{qH}{3} \geq 6/qH^3$.

And we have

$$Pr[B(G_0') = 1] = Pr[F]\frac{1}{qT}\sum_{i=0}^{qT-1} Pr[A(G_{i+1}) = 1]$$

and

$$Pr[B(G_1') = 1] = Pr[F]\frac{1}{qT}\sum_{i=0}^{qT-1} Pr[A(G_i) = 1]$$

So,

$$\begin{aligned}
Adv_B^{CKS-light}(k, q_C', q_{CR}') &= |Pr[B(G_0') = 1] - Pr[B(G_1') = 1]| \\
&= \frac{Pr[F]}{qT}|\sum_{i=0}^{qT-1} Pr[A(G_i) = 1 - \sum_{i=0}^{qT-1} Pr[A(G_{i+1}) = 1| \\
&= \frac{Pr[F]}{qT}|Pr[A(G_0) = 1 - Pr[A(G_1) = 1| \\
&= \frac{Pr[F]}{qT} Adv_B^{CKS}(k, q_H, q_C, q_{CR}, q_T) \\
&\geq 2.Adv_B^{CKS}(k, q_H, q_C, q_{CR}, q_T)/q_H^3 q_T
\end{aligned} \tag{3.1}$$

This concludes the proof.

□

Theorem 3 (CKS-heavy $\iff$ CKS-light) *A NIKE scheme NIKE3USER is secure in the CKS-heavy model if and only if it is also secure in the CKS-light model. In more detail, for any adversary A against NIKE3USER in the CKS-heavy model, there is an adversary B that breaks NIKE3USER in the CKS-light model with*

$$Adv_B^{CKS-light}(k, q_C', q_{CR}') \geq 2.Adv_B^{CKS-heavy}(k, q_H, q_C, q_E, q_{HR}, q_{CR},)/q_H^3$$

Proof:

Security in the sense of the CKS-heavy model implies security in the sense of the

CKS-light model. Here we prove that if a NIKE scheme NIKE3USER is secure in the CKS-light model, it is also secure in the CKS-heavy model.

Suppose there is an adversary A against NIKE3USER in the CKS-heavy model with advantage $Adv_A^{CKS-heavy}(k, q_H, q_C, q_E, q_{HR}, q_{CR},)$, like the previous proof we show how to construct an algorithm B against NIKE3USER in the CKS-light model that uses A to break NIKE3USER with advantage

$$Adv_B^{CKS-light}(k, q_C', q_{CR}') \geq 2.Adv_B^{CKS-heavy}(k, q_H, q_C, q_E, q_{HR}, q_{CR},)/q_H^3$$

where k is the security parameter.

B plays the CKS-light security game with challenger C and acts as a challenger for A. C takes as input the security parameter 1^k, runs algorithm CommonSetup of the NIKE3USER scheme and gives B params. C then takes a random bit b and answers oracle queries for B until B outputs a bit b'.

Let q_H be a bound on the number of register honest user ID queries made to B by A in the course of its attack. B chooses three distinct indices I, J and K uniformly at random from $\{1, 2, ...q_H\}$. A makes a series of queries which B answers as follows:

Register corrupt user ID : If A makes a register corrupt user ID query supplying (ID, pk) as input, B makes the same register corrupt user ID query to C. C records the tuple $(corrupt, ID, pk, \perp)$.

Register honest user ID : Here A supplies a string ID to B. If this is the I^{th} or J^{th} or K^{th} such query, then B sets $ID_I = ID$ or $ID_J = ID$ or $ID_K = ID$ as appropriate. Then B makes the same register honest user ID query to C. On input $params$ and ID, C runs KeyGen, generating a key pair (pk, sk), records $(honest, ID, pk, sk)$ and returns pk to B. B gives pk to A. Otherwise, when this is not the I^{th} or J^{th} or K^{th} such query, B generates a key pair (pk, sk), by running algorithm KeyGen on input $params$ and ID, and makes a register corrupt user ID query to C on inputs the string ID and the public key pk. B then gives pk to A.

Extract : Whenever A makes an extract query on a user identity ID, that was registered by A as $honest$, B checks if $ID \in \{ID_I, ID_J, ID_K\}$. If so, B aborts the simulation. If $ID \notin \{ID_I, ID_J, ID_K\}$, B finds ID in the list $(honest, ID, pk, sk)$ and returns sk to A.

Honest reveal : Whenever A supplies three identities ID, ID', ID'', where ID, ID' and ID'' were registered by A as honest users, B will check if $\{ID, ID', ID''\} = \{ID_J, ID_J, ID_K\}$. If so, B aborts the simulation. (Note that in this case B does not have either of the secret keys needed to compute the paired key among the three identities.) Otherwise, B runs SharedKey on the appropriate inputs. (Note that in this case, B has at least one of the secret keys needed to execute SharedKey.)

Corrupt reveal : Now, if A supplies three identities ID, ID', ID'' where ID was registered by A as corrupt and ID', ID'' were registered as honest, B will check if $ID' \in \{ID_I, ID_J, ID_K\}$ or $ID'' \in \{ID_I, ID_J, ID_K\}$. If so, B will make a Corrupt reveal query to C obtaining the shared key between $ID, ID' and ID''$, $K_{(ID,ID',ID'')}$. B then returns the result to A. If $ID' \notin \{ID_I, ID_J, ID_K\}$ and $ID'' \notin \{ID_I, ID_J, ID_K\}$, then this means that B has $sk_{ID'}$ and $sk_{ID''}$. Then B runs SharedKey using $sk_{ID'}$ and $sk_{ID''}$ as an input and returns $K_{(ID,ID',ID'')}$ to A.

Test : Whenever A makes its Test query on a set of three of user identities $\{ID_A, ID_B$ $, ID_C\}$, B checks if $\{ID_A, ID_B, ID_C\} = \{ID_J, ID_J, ID_K\}$. If so, B makes a $Test$ query to C on $\{ID_A, ID_B, ID_C\}$ and gives the result to A. If not, B aborts simulation.

This completes our description of B's simulation. When A terminates by outputting a bit b' then B outputs the same bit. We now assess B's success probability. Let F denote the event that B is not forced to abort during its simulation. So,

$$Pr(F) \geq 1/\binom{qH}{3} \geq 6/qH^3$$

Hence we conclude that

$$Adv_B^{CKS-light}(k, q'_C, q'_{CR}) \geq 2.Adv_B^{CKS-heavy}(k, q_H, q_C, q_E, q_{HR}, q_{CR},)/q_H^3$$

$\square$

In the same way we can prove that CKS-heavy andm-CKS-heavy models are polynomially equivalent. Hence it follows from the above theorems that all the NIKE models are polynomially equivalent.In the next theorem we'll prove the security of NIKE3USER only for CKS-light model and the security follows for other models accordingly.

Theorem 4 *Assume ChamH is a family of chameleon hash functions. Then NIKE3USER is secure under the DBDH assumption relative to generator G_1. In particular, suppose A is an adversary against $NIKE3USER$ in the CKS-light security model. Then there exists a DBDH adversary B with:*

$$Adv^{dbdh}_{B,G1}(k) \geq Adv^{CKS-light}_{A,NIKE3USER}(k) - Adv^{coll}_{A_{CH},ChamH}(k)$$

Proof:

Game 0 : Let Game 0 be the original attack game as described in the CKS-light security model. By definition, we have that:

$$Adv^{CKS-light}_{A,NIKE3USER}(k) = |Pr[S_0] - 1/2|$$

Game 1 (Eliminate Hash Collision) : In this game, the challenger changes its answers to $registercorruptuserID$ queries as follows: let A, B and C be the identities of the two honest users, and let their public keys be $(X_A, Z_A, r_A), (X_B, Z_B, r_B), (X_C, Z_C, r_C)$ respectively. Let D be the identity of a user that is the subject of a register corrupt user ID query with $pk_D = (X_D, Z_D, r_D)$. If $t_D = ChamH_{hk}(Z_D||D; r_D) = ChamH_{hk}(Z_A||A; r_A)$ or $t_D = ChamH_{hk}(Z_D||D; r_D) = ChamH_{hk}(Z_B||B; r_B)$ or $t_D = ChamH_{hk}(Z_D||D; r_D) = ChamH_{hk}(Z_C||C; r_C)$, the challenger aborts, otherwise it continues as in the previous game.

Let $abort_{ChamH}$ be the event that a collision was found. Until $abort_{ChamH}$ happens, Game 0 and Game 1 are identical. By the difference lemma, we have :

$$|Pr[S_1] - Pr[S0]| \leq Pr[abort_{ChamH}]$$

and

$$Pr[abort_{ChamH}] \leq Adv^{coll}_{A_{CH},ChamH}(k)$$

Game 2 : In this game a DBDH adversary B on inputs $(g_1, g_1^a, g_1^b, g_1^c, T) \in G_1^4 \times G_T$, where $a, b, c, \in Z_p$, runs adversary A against NIKE3USER simulating the challenger's behaviour as in Game 1. B's job is to determine whether T equals $e(g_1, g_l)^{abc}$ or a random element from G_T, where g_1 is the generator of G_1. B runs Cham.KeyGen(1^k) to obtain a key pair for a chameleon hash function, (hk, ck) (here ck is the trapdoor information for the chameleon hash). It then selects $m_1, m_2, m_3 \xleftarrow{\$} \{0,1\}^*$ and $r_1, r_2, r_3 \xleftarrow{\$} R_{Cham}$, where $RCham$ is the chameleon hash function's randomness space. B computes $t_A = ChamH_{hk}(m_1; r_1), t_B = ChamH_{hk}(m_2; r_2)$ and $t_C = ChamH_{hk}(m_3; r_3)$.

Let $p(t) = p_0 + p_1 t + p_2 t^2 + p_3 t^3$ be a polynomial of degree 3 over Z_p such that

$p(t_A) = p(t_B) = p(t_C) = 0$. Let $q(t) = q_0 + q_1 t + q_2 t^2 + q_3 t^3$ be a random polynomial of degree 3 over Z_p. Then B sets $u_i = (g_1^c)^{p_i} g_1^{q_i}$. Since $q_i \xleftarrow{\$} Z_p$, we have $u_i \xleftarrow{\$} G_1$. Note that then $u_0 u_1^t u_2^{t^2} = (g_1^c)^{p(t)} g_1^{q(t)}$. In particular, $Y_A = g_1^{q(t_A)}, Y_B = g_1^{q(t_B)}$ and $Y_C = g_1^{q(t_C)}$, where $q(t_A), q(t_B)$ and $q(t_C)$ are known values. B then answers the following queries:

Register honest user ID : When B receives a register honest user ID query for identity A from adversary A, it uses the trapdoor information ck of the chameleon hash function to obtain $r_A \in R_{Cham}$ such that $ChamH_{hk}(g_1^a||A; r_A) = ChamH_{hk}(m_1; r_1) = t_A$. According to the definition of chameleon hash functions r_A is uniformly distributed over $R_C ham$ and independent from r_1. Similarly, when B receives a second register honest user ID query for identity B from A, it obtains $r_B \in R_{Cham}$ such that $ChamH_{hk}(g_1^b||B; r_B) = ChamH_{hk}(m_2; r_2) = t_B$. Then r_B is also uniformly distributed over R_{Cham}. Similiarly when B receives the third register honest user ID query for identity C from A, it obtains $r_C \in R_{Cham}$ such that $ChamH_{hk}(g_1^c||C; r_C) = ChamH_{hk}(m_3; r_3) = t_C$. Then r_C is also uniformly distributed over R_{Cham}. Now B sets:

$pk_A = ((g_1^a)^{q(t_A)}, g_1^a, r_A), pk_B = ((g_1^b)^{q(t_B)}, g_1^b, r_B)$ and $pk_C = ((g_1^c)^{q(t_C)}, g_1^c, r_C)$
These are correct public keys since $p(t_A) = p(t_B) = p(t_C) = 0$.

Register corrupt user ID : When B receives a public key pk and a string ID from A, and registers them. As in the original attack game, B aborts if ID equals one of the honest identities, A, B or C.

Corrupt reveal queries : Here we allow adversary to corrupt either A or B. Let D be the corrupt user. B first checks if $pk_D = (X_D, Z_D, r_D)$ is a valid public key using the pairing. If not, it rejects the query. This makes sure that pk_D is of the form (Y_D^d, g_1^d, r_D) for some $d \in Z_p$, where $Y_D = (g_1^c)^{p(t_D)} g_1^{q(t_D)}$ and $r_D \in R$. This means that $X_D = (g_1^{cd})^{p(t_D)} g_1^{q(t_D)}$. Thus, g_1^{cd} can be computed from $X_D, Z_D = g_1^d$ and r_D by:

$$g_1^{cd} = (X_D / Z_D^{q(t_D)})^{1/p(t_D) \mod p}$$

where we use the property that $p(t_D) \neq 0 \mod p$, which follows from the facts that p is a polynomial of degree 3 with roots t_A, t_B, t_C and that $t_D \neq t_A, t_B, t_C$ (because we have eliminated hash collisions already in Game 1). Now writing $pk_A = (X_A Z_A, r_A)$, $pk_C = (X_C Z_C, r_C)$ for the public key of the honest user A, C respectively, the shared key among A, C and D can be correctly computed as:

$$K_{A,C,D} = e(g_1^{cd}, Z_A)$$

Test query : Return T.

This completes our description of B's simulation. Note that distinguishing the real case from the random case for A in Game 2 is equivalent to solving the DBDH problem. To see this, note that for user A, we have $Z_A = g_1^a$ and $X_A = Z_A^{q(t_A)}$, for user B we have $Z_B = g_1^b$ and $X_B = Z_B^{q(t_B)}$ and for user C, we have $Z_C = g_2^c$ and $X_C = Z_C^{q(t_C)}$. Hence $K_{A,B,C} = e(g_1, g_2)^{abc}$.

Now, since B's simulation properly handles all of A's queries and sets up all values with the correct distributions, we have: $Pr[S_2] = Pr[S_1]$.

Game 3 : In this game B replaces the value T with a random element from G_T . Since T is now completely independent of the challenge bit, we have: $Pr[S_3] = \frac{1}{2}$. Game 2 and Game 3 are identical unless adversary A can distinguish $e(g_1, g_2)^{abc}$ from a random element. Therefore we have:

$$|Pr[S_3] - Pr[S2]| \leq Adv_{B,G_1}^{dbdh}(k).$$

By collecting the probabilities relating the different games, we have

$$Adv_{B,G_1}^{dbdh}(k) \geq Adv_{A,NIKE3USER}^{CKS-light}(k) - Adv_{A_{CH},ChamH}^{coll}(k)$$

This conclude our proof.

□

Chapter 4

Conclusion and Future Work

In this thesis, we tried the extend the formal study on Authenticated Key Exchange (AKE) protocols under related randomness for more general scenarios. Our model captures situations where the randomness of an AKE protocol goes bad and proposed a generic transformation of any secure key exchange protocol in public key setting to be secured in related randomness attack scenarios.

Secondly, We provided a different security model for three user NIKE protocols and explored the relationships among them. We provided a specific constructions for secure three user NIKE in the standard model.

As for future work, we hope to construct three user ID based secure NIKE scheme in standard model under DBDH assumption which can easily extended from our proposed model.

At the end, we need to mention that three user AKE/NIKE from twin Bilinear Diffe-Hellman assumption is another interesting problem which we have looked and can be formalized in the near future.

Bibliography

[1] Whitfield Diffie and Martin E. Hellman. “New Directions in Cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[2] Christoph G. Günther. “An Identity-Based Key-Exchange Protocol”. In: *Advances in Cryptology – EUROCRYPT’89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Vol. 434. Lecture Notes in Computer Science. Houthalen, Belgium: Springer, Heidelberg, Germany, 1990, pp. 29–37.

[3] Ray Bird et al. “Systematic Design of Two-Party Authentication Protocols”. In: *Advances in Cryptology – CRYPTO’91*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, 1992, pp. 44–61.

[4] Mihir Bellare and Phillip Rogaway. “Provably Secure Session Key Distribution: The Three Party Case”. In: *27th Annual ACM Symposium on Theory of Computing*. Las Vegas, Nevada, USA: ACM Press, 1995, pp. 57–66.

[5] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. “Key Agreement Protocols and Their Security Analysis”. In: *6th IMA International Conference on Cryptography and Coding*. Ed. by Michael Darnell. Vol. 1355. Lecture Notes in Computer Science. Cirencester, UK: Springer, Heidelberg, Germany, 1997, pp. 30–45.

[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols*. Cryptology ePrint Archive, Report 1998/009. `http://eprint.iacr.org/1998/009`. 1998.

[7] Ran Canetti and Hugo Krawczyk. *Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels*. Cryptology ePrint Archive, Report 2001/040. `http://eprint.iacr.org/2001/040`. 2001.

[8] Ran Canetti and Hugo Krawczyk. “Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels”. In: *Advances in Cryptology – EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. Lecture Notes in Computer Science. Innsbruck, Austria: Springer, Heidelberg, Germany, 2001, pp. 453–474.

[9] William Aiello et al. "Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols". In: *ACM CCS 02: 9th Conference on Computer and Communications Security*. Ed. by Vijayalakshmi Atluri. Washington D.C., USA: ACM Press, 2002, pp. 48–58.

[10] Ran Canetti and Hugo Krawczyk. *Security Analysis of IKE's Signature-based Key-Exchange Protocol*. Cryptology ePrint Archive, Report 2002/120. `http://eprint.iacr.org/2002/120`. 2002.

[11] David Cash, Eike Kiltz, and Victor Shoup. "The Twin Diffie-Hellman Problem and Applications". In: *Journal of Cryptology* 22.4 (Oct. 2009), pp. 470–504.

[12] Guomin Yang et al. *Authenticated Key Exchange under Bad Randomness*. Cryptology ePrint Archive, Report 2011/688. `http://eprint.iacr.org/2011/688`. 2011.

[13] Yu Chen, Qiong Huang, and Zongyang Zhang. "Sakai-Ohgishi-Kasahara Identity-Based Non-Interactive Key Exchange Revisited and More". In: *ACISP 14: 19th Australasian Conference on Information Security and Privacy*. Ed. by Willy Susilo and Yi Mu. Vol. 8544. Lecture Notes in Computer Science. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, 2014, pp. 274–289. DOI: `10.1007/978-3-319-08344-5_18`.

[14] Tsz Hon Yuen et al. "Related Randomness Attacks for Public Key Cryptosystems". In: *ASIACCS 15: 10th ACM Symposium on Information, Computer and Communications Security*. Ed. by Feng Bao et al. Singapore: ACM Press, 2015, pp. 215–223.