ABSTRACT

Title of dissertation: Enhancing Privacy
in Cryptographic Protocols

Ji Sun Shin, Doctor of Philosophy, 2009

Dissertation directed by: Professor Virgil D. Gligor
Department of Electrical and Computer Eng.
Professor A. Udaya Shankar
Department of Computer Science

For the past three decades, a wide variety of cryptographic protocols have been proposed to solve secure communication problems *even* in the presence of adversaries. The range of this work varies from developing basic security primitives providing confidentiality and authenticity to solving more complex, application-specific problems. However, when these protocols are deployed in practice, a significant challenge is to ensure not just security but also privacy throughout these protocols' lifetime. As computer-based devices are more widely used and the Internet is more globally accessible, new types of applications and new types of privacy threats are being introduced. In addition, user privacy (or equivalently, key privacy) is more likely to be jeopardized in large-scale distributed applications because the absence of a central authority complicates control over these applications.

In this dissertation, we consider three relevant cryptographic protocols facing user privacy threats when deployed in practice. First, we consider *matchmaking protocols* among strangers to enhance their privacy by introducing the "durability" and “perfect forward privacy” properties. Second, we illustrate the fragility of formal definitions with respect to password privacy in the context of *password-based authenticated key exchange (PAKE)*. In particular, we show that PAKE protocols provably meeting the existing formal definitions do not achieve the expected level of password privacy when deployed in the real world. We propose a new definition for PAKE that is tightly connected to what is actually desired in practice and suggest guidelines for realizing this definition. Finally, we answer to a specific privacy question, namely whether privacy properties of *symmetric-key encryption schemes* obtained by non-tight reduction proofs are retained in the real world. In particular, we use the privacy notion of “multi-key hiding” property and show its non-tight relation with the IND$-CPA property of symmetric-key schemes. We use the experimental result by Gligor *et al.* to show how a real attack breaks the “multi-key hiding” property of IND$-CPA symmetric-key encryption schemes with high probability in practice. Finally, we identify schemes that satisfy the “multi-key hiding” and enhance key privacy in the real world.

Enhancing Privacy in Cryptographic Protocols

by

Ji Sun Shin

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2009

Dedication

To my parents — my father, Nyun Shik and my mother, Soonja.

Acknowledgments

I want to thank all the people who supported me during my life as a Ph.D. student. First of all, I want to thank the committee members. I thank my advisor, Virgil D. Gligor who patiently guided me and taught me what it is to be a true researcher. I also want to thank A. Udaya Shankar for being my department advisor during my program and for continuing on to be the committee chair. I thank Lawrence C. Washington for introducing cryptography to me and for being the Dean's representative. I thank Donald Perlis for guiding and trusting me when I was a teaching assistant for his courses. I thank Gang Qu for accepting to be a committee member even though he is on sabbatical and was in China when I asked him.

The work in this thesis was supported in part by US Army Research Laboratory and the UK Ministry of Defence under Agreement Number W911NF-06-3-0001 and by the US Army Research Office under Contract W911NF- 07-1-0287 at the University of Maryland.[1] I am grateful for their generosity.

I want to thank all of my friends and colleagues throughout my time as a Ph.D. student. First, I want to thank Taekyoung Kwon who gave me an opportunity to work with him on such an interesting problem. It was one of the most enjoyable research experiences in my life. I thank Maryland research mates Chiu Yuen Koo,

[1]The views and conclusions contained in this thesis are those of the author and collaborators and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, US Army Research Office, the U.S. Government, the UK Ministry of Defense, or the UK Government.

Omer Horvitz, Ruggero Morselli, Adam Bender, Soo Bum Lee, Jaehwan Lee and Gelareh Taban for supporting me and sharing with me so many important moments. I also thank Carnegie Mellon University CyLab mates Bryan Parno, Hayan Lee and Hyun Jin Kim for being friends to me, a short-time visitor from Maryland. I thank all my Korean graduate friends, especially Hyun Young Song.

I thank my parents and my sister for all their support and endless love.

Table of Contents

List of Figures vii

List of Abbreviations ix

1 Introduction 1
1.1 A New Privacy-Enhanced Matchmaking Protocol 2
1.1.1 Our Contributions 5
1.2 Enhancing Password Privacy of Password-Based Authenticated Key Exchange in the Real World 7
1.2.1 Our Contributions 10
1.3 Retaining Non-tightly Reduced Privacy Properties of Secure Encryption Schemes in the Real-World 12
1.3.1 Our Contributions 13
1.4 Thesis Organization 15

2 Preliminaries 16
2.1 Notations 16
2.2 The Adversarial Model 16
2.3 Basic Primitives 18
2.4 Password-based Authenticated Key Exchange (PAKE). 23
2.4.1 Previous Works 23
2.4.2 Efficiency of PAKE 24

3 A New Privacy-Enhanced Matchmaking Protocol 26
3.1 Outline of the Chapter 26
3.2 Related Work 26
3.3 Preliminaries and Assumptions 29
3.4 Security of Privacy-Enhanced Matchmaking 30
3.4.1 Security Properties that Counter On-line Adversaries 33
3.4.2 Security Properties that Counter Off-line Adversaries 36
3.5 Protocol Design 39
3.5.1 Relevant PAKE Security Properties 41
3.5.2 Generalizing Passwords as Low-entropy Secrets and Adding Perfect Blindness 45
3.5.3 Final Step of Building a Privacy-enhanced Matchmaking Protocol 47
3.6 PAKE security implies forward security 52

4 Enhancing Password Privacy of Password-Based Authenticated Key Exchange in the Real World 58
4.1 Outline of the Chapter 58
4.2 Background of Our Attacks 58
4.3 Delay-Based Attacks 59

4.3.1 Counting Attack Queries: Login Requests or Failed Logins? . 60
4.3.2 Timeout-Delay Attack 62
4.3.3 Synchronization-Delay Attack 67
4.3.4 Multi-Domain Attacks 71
4.4 Security Definitions for PAKE Protocols 71
4.4.1 The Formal Definition 72
4.4.2 A Concrete Security Definition 75
4.5 Analysis of the Delay-Based Attacks and Recommendations 77
4.5.1 Timeout-based Attacks 77
4.5.2 Representation of Synchronization-based Attacks 80
4.5.3 Recommendations for Existing PAKE Protocols 81
4.5.3.1 Enhancement 1 81
4.5.3.2 Enhancement 2 83
4.5.3.3 Fully Concurrent PAKE in TLS 85
4.6 Conclusion 85
4.7 Simple Experiments with Delay-Based Attacks 86

5 Retaining Non-tightly Reduced Privacy Properties of Secure Encryption Schemes in the Real-World 90
5.1 Outline of the Chapter 90
5.2 Security Properties of Encryption Schemes 90
5.2.1 Non-tight Reductions 92
5.2.2 Security Properties Obtained by Tight and Non-Tight Reductions 95
5.3 Attacks against Secure Encryption Schemes 100
5.3.1 Overview of Attack Strategies 101
5.3.2 What Enables Network Attacks in Practice? 102
5.3.3 Vulnerability Examples: Nonce-based IND$-CPA Schemes . . 105
5.3.3.1 Key-Collision Attacks against EKR$_p$-KPA/-PPA Security 106
5.3.3.2 Breaking the EKR$_p$-CPA property implies breaking the KH$_p$-CPA property 113
5.3.3.3 Examples of Secure Schemes that withstand Key-Collision Attacks 114
5.4 Related Work 115
5.5 Proofs of Claims 117
5.5.1 Proof of Claim 5.1 117
5.5.2 Proof of Claim 5.2 122

Bibliography 126

List of Figures

1.1 List of Each Chapter Topic and Considered Privacy Properties. . . . 2

3.1 Overview of Our Security Properties and Attacks Countered 32

3.2 Transformations to obtain Privacy-enhanced Matchmaking π'' from PAKE π 40

3.3 Compiler to be applied to PAKE protocol π to yield privacy-enhanced matchmaking protocol π''. 49

4.1 Login failures : Incorrect authentication vs. Time-out. 62

4.2 Timeout-delay attack against the JG protocol: a single protocol instance. 63

4.3 Experimental results of timeout-delay attacks against two provably secure PAKE protocols. (X-axis shows the different duration of timeout Δ in milliseconds, while y-axis shows the number of server authentication messages that allow password-candidate verification by an adversary.) 66

4.4 Experimental results of synchronization-delay attacks against the KOY protocol. (X-axis shows the delay in synchronization in milliseconds, while y-axis shows the number of server authentication messages that allow password-candidate verification by an adversary.) 68

4.5 Sets of instances representing login failures and online attacks. ($r = |\mathcal{R}|$, $r_h = |\mathcal{R}_h|$, $q_o = |\mathcal{Q}_o|$, $q_o^c = |\mathcal{Q}_o^c|$, and $q_o^u = |\mathcal{Q}_o^u|$) 78

4.6 Enhancing PAKE protocol π with implicit authentication into a protocol π'' secure against timeout-delay attacks. 82

4.7 Switching roles of client and server of protocol π to have a protocol π' secure against timeout-delay attacks. 83

4.8 The full handshake for PAKE-TLS ciphersuites in the full concurrency model. 84

4.9 Expanded experimental results of concurrent attacks on provably secure PAKEs. 87

4.10 Experiment of JG protocol. 88

4.11 Experiment of PAK-Z+ protocol. 88

5.1 Scheme CTR1. The nonce space is Nonce=$\{0,1\}^{l/2}$ [Rog04a]. 109

5.2 Scheme CTR2. The nonce space is Nonce=$\{0,1\}^{l}$ [Rog04a]. Unlike CTR1, CTR2 encrypts the nonce before using it as a counter. This scheme is similar to the XOR Scheme with Random Counters [Gli98], which outputs (S, C) at encryption. 109

5.3 Scheme CBC2. The nonce space is Nonce=$\{0,1\}^{l}$ [Rog04a]. 112

List of Abbreviations

PPT	Probabilistic Polynomial-Time
pk	public key
sk	secret (private) key
Π	pi
Σ	sigma
PAKE	Password-based Authenticated Key Exchange
MAC	Message Authentication Code
SSL	Secure Sockets Layer
TLS	Transport Layer Security
IPsec	Internet Protocol Security
KPA	Known Plaintext Attack
PPA	Predictable Plaintext Attack
CPA	Chosen Plaintext Attack
CCA	Chosen Ciphertext Attack
IND	Indistinguishability
IND$	Random-String Indistinguishability
KR	Key Recovery
KH	Key Hiding
EKR	Existential Key Recovery
CTR	Counter
CBC	Cipher Block Chaining

Chapter 1

Introduction

Secure communication problems pose challenges when two (or more) parties participate to complete predefined tasks in a certain desired secure way, *even* in the presence of adversaries. For the past two decades, to solve secure communication problems, cryptography has provided work by (1) establishing a concrete framework to formally define the adversarial model and security model, (2) designing and developing protocol constructions for the real world, and (3) guaranteeing these constructions satisfy the security model via rigorously driven proofs. The solutions provided by cryptography range from basic primitives to complex application-specific problems; fundamental primitives include encryption schemes [GM84, NY90, RS91, BDPR98, BDJR97, KY00, BHSV98, IL89, DDN00], message authentication codes [BKR00] and digital signature schemes [GMR88, NY89, Rom90]. These primitives form the base for more complex, application-specific problems such as password-based authenticated key exchange [BPR00, CHK$^+$05, GL03, GL01, KOY01, MPS00, NV04] and on-line matchmaking [BG85, Mea86, ZN01].

Nevertheless, for those cryptographic protocols that are already developed and currently deployed, it is still a *challenging* task to ensure not only security but also privacy throughout the protocols' lifetime. As computer-enhanced systems become ubiquitous and the Internet becomes more universally accessible, new

Chapter 3. A New Privacy-Enhanced Matchmaking Protocol	Chapter 4. Enhancing Password Privacy of PAKE in the Real World	Chapter 5. Retaining Non-tightly Reduced Privacy Properties of Secure Encryption Schemes in the Real-World
User Privacy and Wish Privacy	Password Privacy	Symmetric-Key Privacy

Figure 1.1: List of Each Chapter Topic and Considered Privacy Properties.
advertising

types of applications and new types of privacy threats can be introduced. Moreover, new Internet-enabled, computer-embedded systems may have a decentralized or distributed architecture. The absence of a single central authority complicates control over system managements. The more systems are distributed, the more likely user privacy (or equivalently, password privacy or secret key privacy) will be jeopardized.

In this dissertation, we consider three problems and in each we take a relevant cryptographic protocol facing privacy threats, identify them and provide necessary security enhancements. We describe our contributions in the next sections (also, see Figure 1.1).

1.1 A New Privacy-Enhanced Matchmaking Protocol

The notion of on-line matchmaking was introduced by Baldwin and Gramlich in 1985 [BG85] anticipating a now-common type of internet service: a job-referral service matching a company that wants to hire an employee having certain characteristics or "wishes" (e.g., skills, level of experience, salary level, credentials), without advertising the job publicly, with an applicant who wants to find a new job without revealing her/his plan to leave the current job. Currently, many other internet services require different forms of on-line matchmaking capabilities; e.g., internet dating services. Further, applications of mobile ad-hoc networks, such as dynamic discovery of peer nodes with identical characteristics (e.g., provenance, configuration, capabilities, level of trust) may benefit from private matchmaking capabilities.

Baldwin and Gramlich provided a solution for on-line matchmaking intended to support (1) anonymity of users (i.e., protecting company and job seekers' identities), (2) authentication of matches, and (3) joint notification to users only in the event of a positive match (i.e., a job seeker's identity is authenticated to the company and vice-versa only when the job-seeker's wishes match the company job requirements). Their solution required a *trusted* matchmaker who learned the identities of the protocol users and their wishes was relied upon not to reveal them. Analysis of the Baldwin-Gramlich protocol shows that their solution can be broken via a simple message replacement attack (viz., Zhang and Needham's attack [ZN01]) revealing users' identities and their desired characteristics to an adversary.

In 1986, Meadows introduced a correct protocol for on-line matchmaking that is independent of an on-line trusted matchmaker [Mea86] beyond the initialization step. Although Meadows' solution provides privacy for the users' credentials, it does not aim at providing anonymity of protocol users, and hence cannot be used for *private* matchmaking.

More recently, Zhang and Needham [ZN01] developed a simple protocol for on-line matchmaking providing some degree of user anonymity and privacy of matching wishes. Their protocol removes any direct interaction between users and requires an untrusted on-line matchmaker that acts as a public bulletin-board which posts encrypted user wishes for retrieval by all interested users. Specifically, a user U hashes his/her wish w using a public hash function to generate an encryption key K. Using this key, the user encrypts w and, separately, identity information and a session key for future communication, and submits the two ciphertexts to the public bulletin board for posting. Any user can download any pair of posted ciphertexts and verify whether his/her wishes match the posted ones and, if so, can obtain the session key for communicating with the matching partner. Although this simple protocol satisfies the first two requirements of Baldwin and Gramlich, it does not support joint notification of matches, since this property would require the (re)introduction of trusted third-parties [ZN01].

The Zhang-Needham protocol faces two significant privacy challenges. First, an adversary can launch an *off-line dictionary attack* to discover the identity of a user who posted wishes on the public bulletin board. Because the wish space must be relatively small to allow straightforward user specification of wishes and clear-cut matches, an adversary can choose any set of possible wishes, hash them to produce an encryption key, and then decrypt the pairs of posted ciphertexts [ZN01]. Thus, exhaustive use of all possible wishes is guaranteed to uncover a matching wish and the identities of the users posting it. Second, even if we make the (unrealistic) assumption that user wishes have large entropy, it is sufficient for an adversary to break the privacy of a posted wish to enable the compromise of *all* previous protocol executions containing that wish. In other words, this protocol does not provide *forward privacy* of users' identities and their wishes.

1.1.1 Our Contributions

We define the goals of *privacy-enhanced matchmaking* protocols by augmenting the original two requirements (i.e., anonymity of protocol users and authentication of wish matches[1]) with new security goals, which appear to be fundamental to private matchmaking. Our overall set of security goals comprises:

- authenticity of users and wish matches;
- privacy of users' identities and of their wishes; in particular:
  - anonymity of users and privacy of wish matches;
  - privacy resistance to off-line dictionary attacks; and
  - forward privacy of users' identities and their wishes.

These security goals are fundamental to privacy-enhanced matchmaking (and other similar) protocols. Authenticity is the basis for trust between users that matched their wishes as it prevents impersonation of legitimate users. Lack of wish privacy can lead to breaches of user privacy since wishes are typically specific to classes of users (e.g., known specific skill sets and other user characteristics, such as desired security clearances, can be linked with certain users and organizations). Resistance to off-line dictionary attacks is also fundamental because, in practice,

[1]As pointed out by Zhang and Needham [ZN01], support of joint notification of users only in the event of a positive match requires an on-line trusted authority, which we also want to avoid.

wish entropy is fairly low: the space of user wishes is rather limited and fairly predictable thereby enabling potent off-line attacks. Finally, forward privacy is also important due to the *durability of privacy concerns*: a breach of privacy in a current protocol run should not cause the break of privacy of older runs (i.e., by analogy to the basic notion of perfect forward security of key exchange protocols).

We also present a privacy-enhanced matchmaking protocol that provably counters any adversary that attempts to violate the privacy goals stated above. The protocol is based on a very simple construction that is efficiently implemented using a password-based authenticated key exchange (PAKE) protocol [BPR00, CHK$^+$05, GL03, GL01, KOY01, MPS00, NV04]. In addition, and as a side result of independent theoretical interest, we show that for any user authentication problem in which secrets are chosen from low entropy sets, two notions related to our last two goals, namely security against off-line dictionary attacks and forward security in the corruption model, are equivalent.[2]

A preliminary version of this work appeared previously [SG08].

[2]Since it is already known that forward security in the corruption model is stronger than security against off-line dictionary attacks in the non-corruption model, we only need to show that security definition of password-based authenticated key exchange in the non-corruption model implies forward security in the corruption model (viz., Section 3.6).

1.2 Enhancing Password Privacy of Password-Based Authenticated Key Exchange in the Real World

The introduction of formal definitions of security marked a turning point in cryptographic-protocol analysis, and has proved to be extremely beneficial in practice. Formal definitions are useful in their own right: they force precise specification of desired goals; enable comparisons between protocols meeting different notions of security; and offer guidance as to what protocols are appropriate to achieve a desired level of security when used as a building block of a larger system. Formal definitions have also made possible rigorous mathematical proofs of protocol security, and provide distributed system and network designers with increased confidence in real-world protocols that can be proven secure in this manner.

What is sometimes not sufficiently appreciated, however, is that translating from formal definitions of protocol security to real-world privacy guarantees can be an extremely delicate exercise. Mismatches between formal definitions of protocol security and real-world implementations often lead to unanticipated attacks and potential *privacy* vulnerabilities. We illustrate the fragility of formal definitions with respect to password privacy in the context of *password-based authenticated key exchange* (PAKE).

Password-Based Key Exchange. Authentication is impossible without sharing *some* information in advance. Perhaps the minimal such information that still provides a useful level of authentication is a short, easy-to-memorize password. Protocols for password-based authenticated key exchange (PAKE) allow two entities who have shared a low-entropy password to ensure that they are communicating with each other (that is, to perform mutual authentication), as well as to establish a high-entropy (cryptographic) session key that can be used to encrypt and authenticate their subsequent communication. Though password-based systems have their drawbacks — their security is inherently limited and this is only exacerbated by users' poor choice of passwords — their convenience (e.g., no special devices need to be carried by users) and ease of deployment (e.g., no public-key infrastructure to support use of public key primitives needed) seem to ensure their widespread use for the foreseeable future. Indeed, chances of large-scale deployment of PAKE protocols are greatly enhanced by their recent IEEE standardization [IEE05] and proposed use for Web applications within the SSL/TLS suite [ABC$^+$06].

Definitions for PAKE protocols are somewhat atypical in that they must explicitly take into account the fact that an adversary can "break" *any* protocol with "high" probability by either making a lucky guess of the correct password or by performing an on-line dictionary attack in which it repeatedly attempts to impersonate the client. (This is in contrast to typical cryptographic definitions which require that an adversary's probability of breaking some scheme be "negligible".) Informally, existing definitions take the following form. Let N denote the size of the space from which passwords are chosen, and assume for simplicity that passwords are chosen uniformly at random. Then a PAKE protocol is said to be "secure" if for any Q and any (polynomial-time) adversary making at most Q login attempts, the probability that the adversary succeeds in falsely authenticating as the client is at most (negligibly better than) Q/N. In particular, this implies the desirable property that *off-line* dictionary attacks – a major concern for PAKE security – succeed with only negligible probability, and the best attack an adversary can launch is an on-line dictionary attack.

The Problem with Existing Definitions. Our research shows that the existing, widely-accepted formal definitions of security for PAKE protocols are inadequate in that they do not match, nor do they provide any way to achieve, the level of security desired in practice. Specifically, these definitions bound the security of a protocol (formally, the probability of an adversary's "breaking" the scheme) as a function of the number of on-line attacks that occur, whereas in practice one would prefer an *absolute* bound on the security of the protocol independent of the number of on-line attempts. (Note that this implies some mechanism for limiting the number of on-line attacks that an adversary can carry out.) The natural way to translate from one to the other is to *lock* a user's account once a pre-specified number Q^* of failed log-in attempts occur; indeed, this was suggested as the "obvious" approach in several of the aforementioned works. In other words, to achieve security ϵ one would set $Q^* = \epsilon \cdot N$ and refuse any login requests once Q^* failed attempts have been made on any given user's account. We argue that *this does not work* for *any* PAKE protocol implemented in large-scale networks, such as the Internet. Moreover, we show that certain protocols fare worse than others in this regard.

Put differently, the fault is that the existing formal definitions are *descriptive* rather than *prescriptive*; i.e., they tell us after the fact what security we can expect when faced with an adversary who carries out a certain number of on-line attacks, but do not provide any way of bounding the number of on-line attacks so as to obtain a certain level of security.

1.2.1 Our Contributions

Overview of Our Attacks. In Chapter 4, we justify the claims made in the above and show two general classes of attacks that enable an adversary to exceed Q^* on-line attacks even if user accounts are locked after Q^* failed attempts. The first attack, which we call a *timeout-delay attack*, applies to any PAKE protocol where the server authenticates first. For any such scheme, we show that an adversary can carry out password guesses without causing an explicit authentication failure by simply aborting the protocol before sending its final message. Even if such "timed-out" sessions are eventually recorded by the server as failed log-in attempts, the (necessarily) long delay introduced before the failure is recorded allows an adversary to test many more than Q^* passwords. Thus, from a practical perspective, PAKE protocols in which the client authenticates first should be preferred (even though there is no difference vis-a-vis the formal definitions).

Our second attack, is a well-known *synchronization-delay attack* found in other areas of password protection. This attack applies even to PAKE protocols where the client authenticates first. Here, the attack relies on the fact that any real-world PAKE implementation will be *distributed* across multiple servers. The reason for this is that all server registries that store account/password information must be replicated for reasons of availability and responsiveness (viz., the replication of the

Kerberos Key Distribution Center databases, as a typical well-known example). Because of this, there will be a noticeable delay from the time the Q^*th failed login attempt occurs and the time this information is propagated to all server replicas. Once again, an adversary can exploit this delay to exceed the pre-specified bound Q^* on the number of on-line attacks tolerated.

Although, in practice, synchronization-delay attacks have had limited impact in the past,[3] we nevertheless present these attacks for three reasons. First, formal definitions are supposed to capture adversary bounds precisely, rather than approximately, and independently of an adversary's attack strategy; these definitions would be inadequate if they applied only to some attack strategies, but not others. Second, if these attacks unaccounted, formal PAKE definitions may become inconsistent with practical use of these protocols. Third, synchronization-delay attacks could be amplified in large-scale PAKE deployment to the point of non-compliance with published password standards and guidelines (discussed in Chapter 4).

The effects of these two attacks can be amplified by launching *multi-domain attacks*, in which a user has accounts with several distinct domains and he uses

[3] In practice, synchronization-delay attacks have been ignored for three reasons: (1) synchronization delays are unlikely to be the weakest link in password authentication (e.g., password guesses against multiple accounts may be more damaging); (2) their attack effectiveness, measured as the difference between between bound Q^* and password guesses allowed by synchronization delays, is relatively small when compared with other attacks (viz., time-delay attacks in Section 4.3.2), and hence more difficult to exploit on a large scale in practice; and (3) simple, obvious synchronization solutions do not work well in large-scale Internet deployment (explained in Section 4.3.3 below). Hence, the cost-effectiveness case for eliminating such delays may not be compelling.

the same (or related) password in all accounts. Unless synchronized cross-domain enforcement of failed-login bounds can be negotiated – a virtually impossible proposition for both administrative and technical reasons (discussed in Section 4.3.4) the effective number of queries the adversary can use reaches nQ, for n domains, far exceeding any reasonable security bound Q^* even for relatively small n (e.g., half a dozen domains).

Summary of Contributions. In Chapter 4, we specifically describe the attacks outlined above, and provide analytical and experimental evidence backing up our claims. Faced with a significant gap between what existing definitions guarantee and what is actually desired, we propose a new definition of security. The goal of the new definition is to specify a precise bound on the probability of success of the adversary, rather than to simply analyze the behavior of the protocol as a function of the number of queries the adversary makes. We analyze our attacks in light of the new definition, and provide privacy enhancements for realizing the new definition. Finally, we show how to apply these privacy enhancements in particular to the PAKE protocols currently proposed for TLS [ABC+06].

1.3 Retaining Non-tightly Reduced Privacy Properties of Secure Encryption Schemes in the Real-World

The asymptotic approach to proving the security of encryption schemes has two remarkable benefits: (1) unprecedented precision in defining goals and capabilities of adversary attacks and countermeasures, and (2) generality of security definitions and proofs; i.e., independence of application area, adversary strategies and technology advances in computing and communications. In short, the undeniable appeal of asymptotic security definitions and reduction proofs lies in their long-lasting value. Detailed accounts of these benefits are found in a recent book by Katz and Lindell [KL08] and in earlier work by Bellare and Rogaway [Bel98, Rog04b].

However, it is generally known that the asymptotic approach fails to account for practical attacks enabled by non-tight proof results, since it does not distinguish between tight and non-tight proofs. In contrast, the concrete security approach [BDJR97, Bel98] recognizes this difference between proof results and prescribes precise bounds on the non-tightness factors (defined in Chapter 5). However, in Chapter 5, we show that these bounds can be circumvented in the context of symmetric key encryption schemes.

1.3.1 Our Contributions

In Chapter 5, we show that privacy properties of symmetric-key encryption schemes that are obtained by non-tight reduction proofs are not retained in the real world when those schemes are implemented with *standard* block ciphers. In particular, we illustrate this by introducing "multi-key hiding" property of symmetric-key encryption schemes. Intuitively, key hiding property captures the key privacy concern: ciphertexts produced by encryption with the same key cannot be distinguished from those produced by encryption with different keys. The "multi-key hiding" property defined in Chapter 5, allows the adversary to more than 2 oracles where as the ordinary "key hiding" property allows only 2 oracles (of encryptions with the same key or encryptions with different keys). This property was parenthetically suggested by Abadi and Rogaway [AR00] who observed that it is indistinguishable from the ordinary "key hiding" property, from the point of view of adversary power; i.e., an adversary gains no extra power by accessing more than 2 oracles instead of just two. We show that these observations while correct in the asymptotic and formal-methods approach, do not hold in practice.

To show this, we prove the non-tight reduction of showing that IND$-CPA implies "multi-key hiding" property. Then, we use the notion of network adversary initially introduced by Gligor *et al.* [Gli08, GPS09] to conduct key-collision attacks and show that "multi-key hiding" property of IND$-CPA secure encryption scheme is broken in practice, while the ordinary "key hiding" property of the scheme withstands the attacks. To realize the key-collision attacks, the network adversary takes advantage of (1) the lack of a "security parameter" to strengthen the security of a real-world encryption scheme during much of its lifetime, and the longevity of standard block-cipher parameters (e.g., the lifetimes of two-key triple DES and AES-128 [BBB$^+$07] are measured in decades rather than a few years); and (2) the continuous availability of multiple encryption oracles and attack nodes in the Internet at essentially zero marginal cost during the lifetime of block-cipher parameters. The salient feature of the network attacks is that the adversary amplifies its advantage in attacking these schemes *quadratically* by increasing the amount of (commercially available) computational resources *only linearly*, while the unit cost of the dominant resource (storage) continues drop by 37 - 50% every year [GH03, Gil08].

Our results also show that the adversary goals of "existential key recovery" (e.g., some arbitrary key can be recovered) and "key recovery" (e.g., a specific challenge key can be recovered) are not equivalent. In the real world this means that, while all US national and international standard encryption schemes (modes) [Dwo01] are secure against "key recovery" attacks, some (e.g., nonce-based counter mode and CBC implementations) fail to exhibit both the "existential key recovery" security and "multi-key hiding" (e.g., no bit of a set of keys may be leaked) property, whenever these modes are implemented with standard block ciphers (i.e., two-key 3DES, AES-128). We illustrate several practical encryption schemes that are vulnerable to our network-adversary attacks. This suggests that either these schemes replace vulnerable US standards or the 128 bit AES key size be increased in the near future. We also illustrate some symmetric-key encryption schemes withstanding key-collision attacks, present common characteristics of such schemes and suggest them as solutions enhancing key privacy in the real-world.

A preliminary version of this work appeared previously [GPS09].

1.4 Thesis Organization

In Chapter 2, we introduce notations used throughout the dissertation, the adversarial model for defining security and privacy models, and basic primitives for constructing solutions. We also discuss the notion of password-based authenticated key exchange (PAKE), previous works related to PAKE and the efficiency of existing solutions. Our main work is contained in Chapters 3, 4 and 5 as described earlier.

Chapter 2

Preliminaries

2.1 Notations

Here we introduce some notations that are commonly used throughout the paper. Let $|x|$ denote the length of string x. If S is a set, $x \overset{r}{\leftarrow} S$ means x is an element chosen uniformly at random from S. Also, $x_1, ..., x_m \overset{r}{\leftarrow} S$ means $x_1 \overset{r}{\leftarrow} S; x_2 \overset{r}{\leftarrow} S; ...; x_m \overset{r}{\leftarrow} S$. $\{0,1\}^*$ is the set of all finite, binary strings and $\{0,1\}^n$ is the set of all binary strings of length n. $\mathsf{MAPs}(X, Y)$ is the set of all functions mapping from set X to set Y.

PPT stands for probabilistic polynomial-time. A function ϵ is *negligible* if for all positive t, there exists an n_t such that $\epsilon(n) < 1/n_t$ for all $n > n_t$.

2.2 The Adversarial Model

For our problems considering secure communications, we do not assume any security for communication channels; in particular, we do not assume any confidentiality or authenticity for communication channels. We assume that the adversary is given complete control over communication channels; more specifically, the adversary is able to :

- eavesdrop on messages between parties.

- modify, insert or drop messages, in particular, the adversary may change the content of messages or change the source of messages (e.g., impersonate the source).

- simply deliver messages (i.e., forward messages).

Our adversarial model over communication channels is referred to "man-in-the-middle" adversaries.

For the adversary attacking encryption schemes, the adversary is given ciphertexts and may try to reveal plaintexts, or discover secret keys. No matter what the adversary's strategies and goals are, the adversary can have several different levels of capabilities and can be categorized as follows [WT02]:

1. Ciphertext-only attack: this type of adversary is only able to obtain ciphertexts.

2. Predictable plaintext attack (PPA): in this case, the adversary is given a copy of a ciphertext and a set of candidate plaintexts such that the ciphertext is an encryption of one of the candidate plaintexts. Therefore, the adversary, given a ciphertext C, may not know exactly what is the corresponding plaintext, but can predict possible candidates of plaintexts corresponding to the ciphertext C. A good example is weather. If the adversary knows that a given ciphertext is an encryption of a weather description, the adversary can predict the plaintext might be "sunny", "rainy", "cloudy", or something else depending on the word format of terminologies used to describe the weather.

3. Known plaintext attack (KPA): in this case, the adversary can obtain a ciphertext and the corresponding plaintext. For example, the adversary knows that the letter always starts with “Dear Officer” and obtains its corresponding ciphertext [Bel97, HS93].

4. Chosen plaintext attack (CPA): in this case, the adversary is allowed to access to the encryption oracle for a limited amount of time, without the secret key being revealed. The adversary can submit a plaintext chosen by herself and obtain the corresponding ciphertext.

5. Chosen ciphertext attack (CCA): in this case, the adversary is allowed to access the decryption oracle; the adversary submits a ciphertext chosen by herself and obtains the corresponding plaintext.

2.3 Basic Primitives

We now review definitions of basic cryptographic primitives for constructing the solutions in this dissertation.

Pseudorandom Functions [GGM86]. Informally, pseudorandom function family is a set of functions, each of which is identified by a random key. Given the key, it is easy to compute the function. However, without the key, the function looks like a random function even when you can observe the input-output behavior of the function.

Definition 2.1. Let $F : \{0,1\}^* \times D \rightarrow R$ be a family of functions. Then, F is a *pseudorandom function family* if for any PPT algorithm A, the following is negligible in n:

$$\left|\Pr\left[k \overset{r}{\leftarrow} \{0,1\}^n : A^{F_k(\cdot)} = 1\right] - \Pr\left[g \overset{r}{\leftarrow} \mathsf{MAPs}(D,R) : A^{g(\cdot)}(1^n) = 1\right]\right|$$

where $\mathsf{MAPs}(D,R)$ denotes the set of functions mapping from D to R.

Symmetric-Key Encryption [GM84, BDJR97]. Intuitively, an encryption scheme provides confidentiality of a message when a party A sends the message to another party B. In particular, anyone E except B cannot reveal the contents of the message even if E is eavesdropping on the communications between A and B. A *symmetric-key* encryption scheme is an encryption scheme in which two parties (A and B) share a private key k in advance. Below, we define a symmetric encryption scheme and provide its security definitions under different levels of adversarial capabilities.

Definition 2.2. A symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three PPT algorithms such that:

- The randomized *key generation* algorithm $\mathcal{K}$ takes a security parameter 1^n as input and returns a key k. We denote it by $k \overset{r}{\leftarrow} \mathcal{K}(n)$.
- The *encryption* algorithm $\mathcal{E}$ takes a security parameter 1^n, a message $m \in \{0,1\}^*$ and a key k as input and outputs a ciphertext C. We denote it by $C \leftarrow \mathcal{E}_k(m)$.
- The deterministic *decryption* algorithm $\mathcal{D}$ takes a key and a ciphertext $C\{0,1\}^*$ and returns a string $m \in \{0,1\}^* \cup \{\perp\}$. We denote it by $m \leftarrow \mathcal{D}_k(C)$.

The scheme is required to have *correctness*, satisfying that for any key $k \xleftarrow{r} \mathcal{K}(1^k)$, if $C \leftarrow \mathcal{E}_k(m)$ and $m' \leftarrow \mathcal{D}_k(C)$, then $m' = m$ (i.e., $\mathcal{D}_k(\mathcal{E}_k(m)) = m$) for any message $m \in \{0,1\}^*$.

Definition 2.3. The symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has *indistinguishability under a chosen-plaintext attack* (or, is *IND-CPA secure*) if for every PPT adversary A, the following is negligible in n:

$$\left| \Pr \left[\begin{array}{c} k \xleftarrow{r} \mathcal{K}(1^n); (m_0, m_1) \leftarrow A^{\mathcal{E}_k(\cdot)}(1^n); \\ b \leftarrow \{0,1\}, C \leftarrow \mathcal{E}_k(m_b) \end{array} : A^{\mathcal{E}_k(\cdot)}(m_b) = b \right] - 1/2 \right|$$

Definition 2.4. The symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has *indistinguishability under a chosen-ciphertext attack* (or, is *IND-CCA secure*) if for every PPT adversary A, the following is negligible in n:

$$\left| \Pr \left[\begin{array}{c} k \xleftarrow{r} \mathcal{K}(1^n); (m_0, m_1) \leftarrow A^{\mathcal{E}_k(\cdot), \mathcal{D}_k(\cdot)}(1^n); \\ b \leftarrow \{0,1\}, C \leftarrow \mathcal{E}_k(m_b) \end{array} : A^{\mathcal{E}_k(\cdot), \mathcal{D}_k^*(\cdot)}(m_b) = b \right] - 1/2 \right|$$

where $\mathcal{D}_k^*(\cdot)$ is the decryption oracle $\mathcal{D}_k(\cdot)$ except answering to a query C.

Message Authentication Code (MAC) [BKR00]. Message authenticate codes allow two parties, sharing a private key k, to communicate each other in an authenticated way. In particular, in a communication, a party sends a tag along with a message so that the other party can verify the validity of the message by using the tag and their sharing key k. For this, it is required that the adversary who obtains many tags corresponding to her choice of messages is not able to forge a valid tag for a new message. Below, we provide a definition of message authentication codes and its security definition.

Definition 2.5. A message authentication code (MAC) $\Pi = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ consists of three PPT algorithms such that:

- The randomized *key generation* algorithm $\mathcal{K}$ takes a security parameter 1^n as input and returns a key k. We denote it by $k \overset{r}{\leftarrow} \mathcal{K}(n)$.
- The *authentication* algorithm $\mathcal{S}$ takes 1^n, the key k and message $m \in \{0,1\}^*$ and outputs a tag $\tau \in \{0,1\}^*$. We denote it by $\tau \leftarrow \mathcal{S}_k(m)$.
- The deterministic *verification* algorithm $\mathcal{V}$ takes a message m, a tag τ and the key k and outputs a bit. We denote it by $b \leftarrow \mathcal{V}_k(m, \tau)$.

The scheme is required to be *correct* satisfying for every $k \overset{r}{\leftarrow} \mathcal{K}(1^n)$, if $\tau \leftarrow \mathcal{S}_k(m)$, then $\mathcal{V}_k(m, \tau) = 1$ holds for all $m \in \{0,1\}^*$.

Definition 2.6. A message authentication code $\Pi = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is *secure under adaptive chosen message attack* if for any PPT algorithm A, the following is negligible in n:

$$\Pr[k \leftarrow \mathcal{K}(1^n); (m, \tau) \leftarrow A^{\mathcal{S}_k(\cdot)}(1^n) : \mathcal{V}_k(m, \tau) = 1]$$

where m was not queried to the oracle $\mathcal{S}_k(\cdot)$.

Signature Scheme [GMR88]. A signature scheme provides authenticity of messages. Informally, a message sender A is given a pair of keys (pk, sk) and makes pk public and keeps sk in secret. A signs a signature σ of a message m with its secret key sk. Then, anyone who is given message m, its associated signature σ and A's public key pk, can verify if m is the message signed by A. For this, it is required that no one except A can forge a new signature that is valid under A's public key pk.

It should hold even when the adversary is allowed to obtain many valid signatures corresponding to her choice of messages. Below, we provide a formal definition of signature schemes and its security definition of unforgeability under adaptive chosen message attacks.

Definition 2.7. A signature scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ consists of three PPT algorithms such that:

- The randomized *key generation* algorithm $\mathcal{K}$ takes a security parameter 1^n as input and returns the public key pk and the secret key sk. We denote it by $(pk, sk) \overset{r}{\leftarrow} \mathcal{K}(1^n)$.
- The *signing* algorithm $\mathcal{S}$ takes 1^n, the secret key sk, and a message $m \in \{0,1\}^*$ and outputs a signature $\sigma \in \{0,1\}^*$. We denote it by $\sigma \leftarrow \mathcal{S}_{sk}(m)$.
- The deterministic *verification* algorithm $\mathcal{V}$ takes a public key pk, a message m and a signature σ and outputs a bit. We denote it by $b \leftarrow \mathcal{V}_{pk}(m, \sigma)$.

The scheme is required to have the *correctness* property satisfying that for every $(pk, sk) \overset{r}{\leftarrow} \mathcal{K}(1^n)$, if $\sigma \leftarrow \mathcal{S}_{sk}(m)$, then $\mathcal{V}_{pk}(m, \sigma) = 1$ holds for any $m \in \{0,1\}^*$.

Definition 2.8. The signature scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is *existentially unforgeable under adaptive chosen message attacks* if for any PPT algorithm A, the following is negligible in n:

$$\Pr[(pk, sk) \leftarrow \mathcal{K}(1^n); (m, \sigma) \leftarrow A^{\mathcal{S}_{sk}(\cdot)}(1^n, pk) : \mathcal{V}_{pk}(m, \sigma) = 1]$$

where m was not queried to the oracle $\mathcal{S}_{sk}(\cdot)$.

2.4 Password-based Authenticated Key Exchange (PAKE).

Password-based authenticated key exchange allows two parties holding only short, human-memorable passwords to establish a secure session key of high-entropy when they share the same password. Such a key exchange is authenticated in a sense that it is secure against man-in-the-middle adversaries. While on-line attackers can guess a password with non-negligible probability, prevention of on-line attackers is straightforward with other mechanism (e.g., access block after consecutive log-in failures), it is not easy to prevent off-line attacker from enumerating all possible passwords of small space into execution transcripts. Therefore, essentially, major security property of password-based authenticated key exchange is security against off-line dictionary attackers. The formal definition of PAKE is provided in Section 4.4.1.

2.4.1 Previous Works

Clearly, "standard" shared-key authentication protocols (e.g., CHAP) are not suitable for PAKE since such protocols allow a passive eavesdropper who monitors even a single execution to mount an off-line dictionary attack and recover a low-entropy password. Lomas et al. [LGSN89a, GLNS93] (see also [HK99, Boy99]) gave the first password-based protocols resistant to such attacks, but in a "hybrid" model where a client and server share a password and the client additionally stores the server's public key. The seminal work of Bellovin and Merritt [BM92, BM93] was the first to consider a pure, password-only model and to propose protocols for this setting. Formal definitions for this problem were given by [BPR00, BMP00, GL06b, CHK$^+$05], and by now numerous provably-secure protocols are known in both the random oracle [BPR00, BMP00, MPS00, Mac02, BCP03] and the so-called "standard" [GL06b, KOY01, JG04, CHK$^+$05, GL06a, Gen08] cryptographic models.

2.4.2 Efficiency of PAKE

Although very efficient PAKE constructions exist [BPR00, BMP00], they rely on the idealized assumptions such as the ideal cipher and random oracle model. Those solutions only provide heuristic security when the random oracle is replaced by a public function such as SHA-1.

The KOY protocol by Katz *et al.* [KOY01] and the GL protocol by Gennaro and Lindell — a generalization of the KOY protocol [GL03] — are PAKE constructions in the common reference string model and do not require any idealized assumptions.

According to the efficiency analyze in [KOY01], each user only needs roughly 7-8 exponentiation computations. The cost is around 4 times greater than standard Diffie-Hellman key exchange that provides no authentication (i.e., no security against man-in-the middle attackers).

Very recently, Gennaro [Gen08] provided ways of improving the efficiency of the KOY protocol and the GL protocol. They pointed out that both of the KOY and the GL protocols use one-time signatures to provide authentication (against man-in-the-middle attack) which increases the bandwidth requirement for the message transmission. They improve the efficiency of those protocols by replacing one-time signatures with faster and shorter message authentication codes. Consequently, assuming a security parameter of 128, such an improvement saves as much as 12 Kbytes of bandwidth; while one-time signature schemes require around 12 Kbytes key and signature transmission, only 256 bits transmission is necessary for the MAC.

Chapter 3

A New Privacy-Enhanced Matchmaking Protocol

3.1 Outline of the Chapter

In Section 3.2 we explore a variety of related problems and explain the differences between these problems and ours. In particular, we argue that solutions to these related problems are insufficient to solve our problem. In Section 3.3 we introduce preliminaries and assumptions. In Section 3.4 we present security properties necessary to provide user privacy and wish privacy. Then, enhanced with these security properties, we give a new definition for matchmaking protocols. In Section 3.5 we construct an efficient solution satisfying our security definition.

3.2 Related Work

Secret Handshakes. The problem of secret handshakes is directly related to our problem. Secret handshakes allow two parties, which are suspicious about each other's affiliation, to securely recognize each other only if they have the same affiliation [BDS$^+$03, CJT04, TX06]. When compared to our problem, one can easily see secret handshakes is a specific instance of a privacy-enhanced matchmaking protocol (in other words, the latter is more general problem than the former). All secret handshake problems studied to date assume that each party uses classical cryptographic (i.e., high-entropy) keys that are distributed by a group manager prior to any execution of the protocol. In contrast, the use of low-entropy secrets (i.e., wishes) in our problem is an important practical requirement. Hence, in any secret-handshake setting where members in a same group are sharing a low-entropy password, our problem can provide a practical solution. Moreover, a secret handshake implemented from our solution can enjoy different flavors of communications as follows:

- with full anonymity: users communicate each other as long as they are convinced that they belong to a same group *without* ever being traced (i.e., identified), or

- with privacy-preserving entity authentication: once users are convinced they belong to the same group, they identify each other prior to further communication. or

- with traceability and anonymity: users can be traced by the group manager while full anonymity is preserved among users.[1]

However, existing solutions of secret handshakes do not fit into our problem as we cannot assume the use of high-entropy secrets, which is a fundamental requirement of all secret-handshake solutions.

[1]If we equip our protocol with a group signature scheme, we can implement a secret handshake protocol based on low-entropy passwords that fully satisfies the security properties of the extant notion of secret handshakes [BDS$^+$03, CJT04, TX06].

Set Intersection. Set intersection allows two or more parties, each having a set of elements, to securely learn the intersection of their sets without revealing elements not in the intersection [KS04, FNP04]. The major difference between our problem and set intersection is that set intersection is not necessarily an *exact matching.* Therefore, by engaging in an interaction with an honest user on an input of a set including all possible elements, an adversary can determine the honest user's input with probability one. Therefore, set interaction cannot provide a secure solution for our problem.

Trust-Negotiation. Trust negotiation allows a client to access a server's resources without having to reveal all the client's credentials and disclose the complete server's access policy, provided by the server's policy is satisfied [BS00, WSJ00, SWY01, YWS01, YW03, WL04]. Within an appropriate setting, our problem can be applied to each step of gradual negotiation to see whether each of a client's credentials exactly satisfies each access check of the server's policy. Furthermore, our solution can enhance client and server privacy so that their identities are not revealed until the last step of trust-negotiation is satisfied. Their identities are also kept anonymous to passive eavesdroppers. However, trust-negotiation solutions do not consider user (i.e., clients and servers) anonymity.

Other Privacy-Preserving Problems. Several problems have been introduced in the privacy-preserving area of access control. Hidden credentials [HBStKO03, BHS04], oblivious envelopes [LDB03, NT06, LL05, LL06] and policy-based encryptions [BM05, BMC06] are relevant examples. However, they focus mainly on the privacy of entity's attributes, for example, affiliation, policy, etc., and do not consider all users' privacy concerns. In particular, all problems assume that users know each other's identity, while in our problem users' identities are not revealed unless they have a common (i.e., a matching) wish. For similar reasons, it is doubtful whether generic two-party secure computation protocols [Yao86, GMW87, Gol04] can provide a solution of our problem; i.e., to date, all generic two-party secure computation protocols are carried out in settings where identities of two parties are known to each other (Nevertheless, the possibility of applying generic secure two-party computation protocols to solve anonymous communication problems represents an interesting open research problem).

3.3 Preliminaries and Assumptions

Anonymous Communication Channels. Like most other privacy-preserving protocols, privacy-enhanced matchmaking requires the use of anonymous communication channels. Use of ordinary communication channels is inadequate because anonymity and hence identity privacy (e.g., linking user actions) can be simply broken via eavesdropping on communication messages. Among other measures, anonymous communication relies on pseudonym-naming – a feature commonly provided by most privacy protocols. In practice, low-latency anonymous channels exist (e.g., Tor, JAP [DMS04, HFW]).

Untrusted Matchmaker. A matchmaker publishes description of matchmaking, roles and wishes. Also, the matchmaker binds a pseudonym to a user's address of anonymous communication channel and signs, distributes and revokes pseudonyms. However, the matchmaker is not trusted with the privacy of users. We assume that the matchmaker functions correctly.

Protocol Users and Secret Wishes. Let $\mathcal{U}$ be a fixed set of users who may participate in protocol executions. Although $\mathcal{U}$ is public information, we assume that users start communicating without knowing any information about each other's real identity but only know pseudonyms which are generated from a set of pseudonyms $\mathcal{I}$. Let $\mathcal{W}$ be a pre-defined set of publicly known wishes. We assume that a wish is a low-entropy secret and hence that $1/|\mathcal{W}|$ is small but non-negligible. For simplicity and clarity, we assume that a user chooses a wish uniformly at random. However, even when any arbitrary relation exists between wishes and users, and such relations are known to the adversary, security definitions can be adjusted appropriately as long as the following assumption holds: for each wish $w \in \mathcal{W}$, there exist at least two users U_1 and U_2 such that they are equally likely to use wish w as an input.

3.4 Security of Privacy-Enhanced Matchmaking

We separate security requirements of our protocol into two classes namely those addressing on-line and off-line adversaries. Their goals and the means of countering them are different. While on-line attacks that try to break the protocol through "on-line" interaction have a non-negligible probability of success in discovering low-entropy secrets, their handling is provided by attack detection and prevention of further protocol executions (discussed in Section 3.4.1 below). In contrast, off-line adversaries are substantially more challenging since such adversaries' off-line attacks can neither be detected nor blocked. For example, off-line adversaries can launch *dictionary* attacks by trying all possible wishes from $\mathcal{W}$ on information that is obtained via passive eavesdropping. Hence, by separating the two types of adversary, we can focus primarily on handling the more potent *off-line adversaries.* [2]

Definitions. An honest user is allowed to execute an unlimited number of protocol instances. Further, a user has a unique pseudonym for each execution (i.e, for each session). Therefore, without loss of generality, we assume that a pseudonym assigned to each execution represents the instance of the execution. Although not specifically stated, an instance of a user execution of our protocol is always performed with a new user pseudonym (Also, we use the notions of sessions and instances interchangeably).

An input of the protocol consists of a user wish, a user pseudonym, (real) user identity, and a partner's pseudonym. For simplicity, we say a user U *uses* a wish w if U takes w as an input secret of the protocol. We say a user A *accepts* a user B if A outputs B at the end of the protocol execution, and it means that A has recognized and authenticated B as a matching-wish partner. We say users A and B *interact* when they are informed of each other's pseudonym and engage in a protocol execution.

[2]We choose to model our security requirements using game conditions [DDM+06] rather than realizing ideal functionality. Our choice is motivated by the fact: (1) sometimes, additional message steps are necessary to realize ideal functionality [Can01, CK01, CHK+05] and (2) sometimes, it is impossible to realize ideal functionality without extra set-up assumptions [Can01, CKL03, DDM+06].

Security Properties

- *Impersonation resistance* counters on-line adversary attacks against user authentication.
- *Detector resistance* counters on-line adversary attacks against user and wish privacy, and match authentication (i.e., unless a match occurs, privacy of users' identities and their wishes are preserved).
- Security properties that counter off-line adversary attacks, namely *matching-result privacy, user unlinkability and wish unlinkability* provide privacy resistance to off-line dictionary attacks and forward privacy of users' identities and their wishes. In particular, *wish unlinkability* captures forward privacy of wishes and *user unlinkability* captures forward privacy of users' identities.

Figure 3.1: Overview of Our Security Properties and Attacks Countered

We say that an adversary $\mathcal{A}$ is *given an interaction with an honest user* U *who is running the protocol on input* w, when (1) an instance of U is initiated with inputs of wish w, its pseudonym I, its real identity U and $\mathcal{A}$'s (i.e., partner's) pseudonym I' and I is known to $\mathcal{A}$; and further, (2) whenever upon receiving a message from $\mathcal{A}$, the next message of the instance is computed according to the protocol and sent to $\mathcal{A}$.

Concrete Security Properties. The security goals for private matchmaking are supported by several concrete security properties that counter both on-line and off-line adversary attacks. The concrete properties are summarized in Figure 3.1 and defined below.

3.4.1 Security Properties that Counter On-line Adversaries

An on-line adversary can use a private matchmaking protocol to detect the identity of a honest user by guessing correctly a user's wishes with small but non-negligible probability (e.g., the probability of a correct guess can be lowered, but only to a limited degree, by extending the size of the wish space). By requiring that the adversary present his/her non-anonymous credentials to an honest user *after* any wish match, the protocol ensures that the user can detect an unwarranted match (or an on-line attack); i.e., a match whereby the adversary cannot present valid identity and wish credentials. Upon detection of an on-line attack, the user can request the revocation of the adversary's (anonymous) credentials from the matchmaker. Using a signed transcript of the adversary-user interaction, the matchmaker requests revocation from the certification authority which issued the adversary's anonymous credentials. A valid user revocation request, would cause the matchmaker to deny issue of a valid (signed) pseudonym to the adversary since the adversary could no longer produce the necessary (anonymous) credentials to the matchmaker after revocation. Thus, further, on-line, anonymous wish guessing by an adversary is blocked. Of course, an honest user would not initiate the matchmaking protocol unless the adversary (or any honest user) produces a matchmaker-signed pseudonym.

Limiting an on-line adversary's protocol execution after an unwarranted wish match requires an initial user interaction with a trusted certification authority. We use an anonymous credential system (e.g., [Cha85, CL01]) so that the user (or adversary) can prove the validity of his/her credentials without revealing his/her identity to the matchmaker. This ensures that user privacy is protected with respect to the matchmaker. All the matchmaker knows is the identity of a certification authority it trusts. Upon receiving a valid anonymous credential from the user, the matchmaker produces a signed pseudonym for a single protocol execution. The matchmaker also keeps a log of the user's proof transcripts (T) along with the corresponding pseudonym (p).

Anonymous credentials are revoked as follows: if the matchmaker receives a report (i.e., signed transcript by a reporter and encrypted with CA's public key) that a user with a pseudonym p guessed a wish but lacked appropriate credentials, the matchmaker finds the proof-transcript T corresponding to the pseudonym p and forwards the report along with T to the certification authority with a signed request to revoke the credential of the user identified in T. The certification authority verifies the validity of the report (e.g., verifies the signatures) and revokes the user's credentials. Note that at no point of the revocation protocol does the matchmaker discover the identities of the user and adversary.

In the above revocation scenario, we did not distinguish between an adversary's non-anonymous identity and wish credentials required by an honest user upon a wish match. In the rest of this chapter, we assume that the adversary is only required to produce a valid non-anonymous identity credential. An attack in which an adversary fails to produce such a credential to an honest user after a match would be significantly more likely than one in which the adversary produces a valid non-anonymous identity and invalid non-anonymous wish credentials. Nevertheless, we note that requiring verification of the adversary's (or any user's) non-anonymous wish credentials upon a wish match does not introduce any additional protocol interaction or complexity, and for this reason we ignore this case for the balance of the chapter.

In the rest of this section, we define two security properties that counter on-line adversaries, namely impersonation resistance and detector resistance. Essentially, the former captures entity authenticity and the latter captures identity privacy.

Impersonation Resistance. Intuitively, impersonation resistance requires that an adversary who is not a legitimate user cannot authenticate itself as a legitimate user to any honest user. This property should hold no matter what secret wish the adversary uses in the impersonation attack (e.g., even when wishes are matching, the adversary should not be able to impersonate a legitimate user).

Definition 3.1. Formally, we say a matchmaking protocol has *impersonation resistance* if, for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following experiment is negligible:

1. $\mathcal{A}$ selects a victim user V and a target user T from $\mathcal{U}$ and a wish w from $\mathcal{W}$ ($\mathcal{A}$ will try to impersonate V to T).[3]

2. Then, $\mathcal{A}$ is given an interaction with T who is running the protocol on input w.

In the experiment, if T accepts V as a matching partner, we say $\mathcal{A}$ *wins.*

[3]We allow $\mathcal{A}$ to choose the wish w, because we want impersonation resistance property to hold even when the adversary impersonating V uses a same wish that T uses.

Detector Resistance. Intuitively, detector resistance captures the identity-privacy concern: given a single interaction of the adversary $\mathcal{A}$ with an honest user, $\mathcal{H}$, adversary $\mathcal{A}$ cannot learn the real identity of $\mathcal{H}$ unless $\mathcal{A}$ and H execute the interaction on a same wish. We model this as an indistinguishability property.

Definition 3.2. We say a matchmaking protocol has the *detector resistance* property if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following experiment is negligibly close to $\frac{1}{2} + \frac{1}{2|\mathcal{W}|}$:

1. A random coin b is flipped. Two random users U_0 and U_1 are selected from $\mathcal{U}$ and a random wish w is chosen from $\mathcal{W}$.

2. If $b = 0$, $\mathcal{A}$ is given an interaction with U_0 who is running the protocol on input w. If $b = 1$, $\mathcal{A}$ is given an interaction with U_1 who is running the protocol on wish w.

3. When the interaction is complete, $\mathcal{A}$ is given the real identities of users, (U_0, U_1) and w.

4. Finally, $\mathcal{A}$ outputs b' (guessing whether $\mathcal{A}$ has an interaction with U_0 or U_1) and if $b' = b$, we say $\mathcal{A}$ *wins*.

3.4.2 Security Properties that Counter Off-line Adversaries

An off-line adversary is eavesdropping on honest executions and then trying off-line dictionary attacks on the obtained information. In this adversarial model, we introduce three relevant security properties, namely matching-result privacy, wish unlinkability, and user unlinkability.

Matching-result Privacy. Intuitively, when given a transcript of an honest execution between two users, the adversary cannot learn anything about the matching result of the execution; i.e., whether two users engaged in the execution on a common wish.

Definition 3.3. We say a matchmaking protocol has *matching-result privacy* if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following experiment is negligibly close to $\frac{1}{2}$:

1. Two random users U_1 and U_2 are selected from $\mathcal{U}$. A coin bit b is flipped.
   - If $b = 0$, a random wish w is chosen. Then, an honest interaction between users U_1 and U_2, both on input wish w, is executed and the execution transcript is given to $\mathcal{A}$.
   - If $b = 1$, two random wishes w_1 and w_2 are chosen from $\mathcal{W}$. Then, an honest interaction between U_1 and U_2, on input wishes w_1 and w_2, respectively, is executed, and the execution transcript is given to $\mathcal{A}$.[4]
2. Finally, $\mathcal{A}$ outputs a bit b' and if $b' = b$, we say $\mathcal{A}$ *wins.*

Wish Unlinkability. Wish unlinkability captures forward privacy of wishes. Intuitively, wish unlinkability requires that the adversary cannot tell in which executions w has been used as an input wish.

[4]A stronger notion of matching-result privacy is possible by letting the adversary know secret wishes and it is achievable by our construction. However, the current notion is sufficient for our purposes.

Definition 3.4. We say a matchmaking protocol has *wish unlinkability* if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following experiment is negligibly close to $\frac{1}{2}$:

1. Two different wishes w and w' are randomly selected from $\mathcal{W}$ and given to $\mathcal{A}$. Four users U_0, U_1, U_2 and U_3 are chosen from $\mathcal{U}$. A coin bit b is flipped.

   - If $b = 0$, an honest interaction between users U_0 and U_1 on input wish w, and another honest interaction between U_2 and U_3 on input wish w are executed and the execution transcripts are given to $\mathcal{A}$.
   - If $b = 1$, an honest interaction between U_0 and U_1 on input wish w and another honest interaction between U_2 and U_3 on input wish w' are executed, and the execution transcripts are given to $\mathcal{A}$.

2. Finally, $\mathcal{A}$ outputs a bit b', and if $b' = b$, we say $\mathcal{A}$ wins.

User Unlinkability. User unlinkability captures forward privacy of users' identities. Intuitively, user unlinkability requires that, when given a transcript of an execution run by a particular user whose real identity is U, the adversary cannot detect whether a new execution transcript belongs to the user U. It should hold even though the adversary has learned wishes used in the executions.

Definition 3.5. We say a matchmaking protocol has *user unlinkability* if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following experiment is negligibly close to $\frac{1}{2}$:

1. Four different users U, U_0, U_1, U_2 are randomly selected from $\mathcal{U}$ and two wishes w, w' are randomly selected from $\mathcal{W}$. U, w and w' are given to $\mathcal{A}$. A coin bit b is flipped.
   - If $b = 0$, an honest interactions between users U and U_0 on input wish w, and another honest interaction between U and U_1 on input wish w' are executed and the execution transcripts are given to $\mathcal{A}$.
   - If $b = 1$, an honest interaction between U and U_0 on input wish w, and another honest interaction between U_2 and U_1 on input wish w' are executed and the execution transcripts are given to $\mathcal{A}$.
2. Finally, $\mathcal{A}$ outputs a bit b', and if $b' = b$, we say $\mathcal{A}$ wins.

Given all the security properties, we define a privacy-enhanced matchmaking protocol.

Definition 3.6. We say a matchmaking protocol is a *privacy-enhanced* if the protocol has impersonation resistance, detector resistance, matching-result privacy, wish unlinkability and user unlinkability.

3.5 Protocol Design

We design a privacy-enhanced matchmaking protocol in a multi-step modular way. First, we take a password-based authenticated key exchange (PAKE) protocol π satisfying certain properties that are useful in building our solution. Then, we generalize passwords into low-entropy secrets (i.e., wishes) and add *perfect blindness*

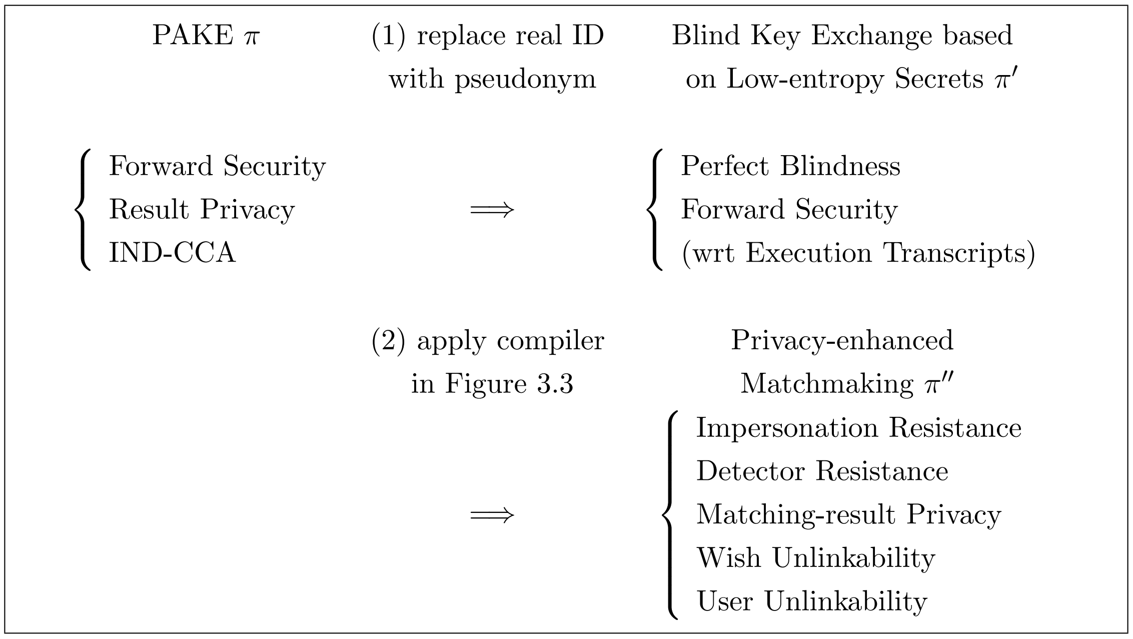

Figure 3.2: Transformations to obtain Privacy-enhanced Matchmaking π'' from PAKE π

by simply replacing user identity field with pseudonym. It will result in a protocol named "blind key exchange based on low-entropy secrets" or BKE-LS in short. Finally, we transform a BKE-LS to a privacy-enhanced matchmaking protocol by adding back entity authentication (which was removed by adding perfect blindness) in a way of providing entity privacy (i.e., confidentiality). For an overview, our procedure to obtain a solution is illustrated in Figure 3.2. We describe each step in detail in the following sections. Note that we omit the revocation protocol for on-line adversaries and assume that such an adversary is limited to a single unwarranted wish match (viz., Section 3.4 above).

3.5.1 Relevant PAKE Security Properties

The PAKE security properties relevant to our protocol are forward security, result privacy, and tight IND-CCA of session key encryption.[5] For these properties, we only focus on *off-line dictionary attackers* which are given transcripts of executions between honest players. We show that these properties provided by a password-based key exchange protocol (PAKE) and hence a PAKE proven secure in the non-corruption model is sufficient to our solution.

Forward Security [BPR00, KOY01]. Intuitively, forward security implies that corruption of a user's password does not break the security of sessions keys used prior to the corruption. This notion has already been introduced in the authenticated key exchange problem where a long-standing belief has been that forward security in the weak corruption model (where the adversary is allowed to corrupt a user's long-term key, or password)[6] is strictly stronger than security in the non-corruption model (where corruption of long-term key, or password, is not allowed). However, in a password-only (i.e., low-entropy secret) setting, we show that any PAKE protocol secure in the non-corruption model also has forward security in the weak corruption model.

Theorem 3.1. *PAKE security in the non-corruption model implies forward security in the weak corruption model.*

[5]We assume that the reader is familiar with the definition of secure PAKE protocols and related notation. For details, we refer the reader to references [BPR00, KOY01].

[6]Here, we only consider the *weak* corruption which, in contrast with the *strong* corruption model, does not allow the adversary to have a complete control over users.

The proof of this theorem is provided in Section 3.6.

Result Privacy. Intuitively, result privacy captures the following property: when the adversary passively observes an interaction between two honest users where the adversary does not know whether the users' passwords are equal, the adversary should not be able to tell whether the two honest users have accepted the same session key. If the adversary can learn that the two honest users have *not* accepted the same session key, then the adversary knows that the users' passwords are different. This notion of result privacy has not received attention before, because for the authentication problem, (1) it is natural for two parties to share the same password (e.g., in advance, by registration), to interact with each other, and (2) two parties who have already had a successful interaction are likely to have further communication, so the success of the result matching would become known to the adversary, anyway. In contrast, the notion of matching-result privacy is an important security property of our problem. Also, in further contrast with traditional PAKE applications, subsequent communication between two users who had a successful match of wishes is also supposed to be anonymous. Therefore, in our problem the result of an interaction between two users cannot possibly become trivially learnable information by an adversary. Hence, result privacy is a relevant property for a PAKE protocol whenever that protocol is used as a building block for privacy-enhanced matchmaking.

Definition 3.7. We say a protocol has *result privacy* if, for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following game is negligibly close to $\frac{1}{2}$: a coin b is flipped; if $b = 0$, a transcript of an honest execution between two random users such that their passwords are different is given to $\mathcal{A}$. If $b = 1$, a transcript of an honest execution between two random users such that their passwords are same is given to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess bit b' and wins if $b' = b$.

PAKE protocols *without* explicit authentication (i.e., with only implicit authentication) satisfy the result privacy as shown by the following theorem.

Theorem 3.2. *Any PAKE protocol with implicit authentication satisfies result privacy.* [7]

The proof of this theorem is similar to that of Theorem 3.1 (except for some technical details) and hence is omitted.

Tight IND-CCA of Session Key-based Encryption. It is well-known (e.g., [BPR00]) that a common session key established between two parties via authenticated key exchange, allows them to have a secure future communication enhanced with either authenticity or confidentiality (or both). For example, by applying the common session key to a symmetric key encryption scheme that has indistinguishability against chosen ciphertext attack (IND-CCA), two parties can communicate each other without losing confidentiality.

[7] So far, our result privacy notion only considered security against passive eavesdroppers explicitly. However, result privacy against active adversaries (i.e., impersonators) is clearly satisfied by the definition of *on-line adversaries*. Intuitively, the definition of on-line attack captures that once the adversary carried out an on-line attack against an honest user by guessing a password and engaging in an execution with the user, the adversary cannot tell whether the guess was correct until the adversary corrupts either the user or the session key that the user has accepted in protocol execution [KOY01, BPR00].

Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ denote any IND-CCA symmetric key encryption scheme. Informally, *tight IND-CCA* with respect to $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ implies that no PPT adversary can distinguish a ciphertext of m_0 from a ciphertext of m_1, where the ciphertexts are encrypted with a session key sk (i.e., $\mathcal{E}_{\mathsf{sk}}(m_b)$) and messages m_0 and m_1 are chosen by the adversary. In particular, we want the highest probability that the adversary breaks this property to be negligibly close to the probability that the adversary distinguishes a real session key from a random key. In PAKE protocol definitions [KOY01, BPR00], this corresponds to the event that the adversary *succeeds* with probability negligibly close to $\frac{1}{2} + \frac{1}{2N}$, where N is the size of low-entropy secret set, whenever the protocol is a secure.[8] More formally, we define a new experiment where adversary $\mathcal{A}$ is given all the oracles except the Test oracle ; viz., the experiment of the PAKE security definition [BPR00, KOY01]. Additionally, we define a new oracle $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}$ as follows:

- $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}(m_0, m_1, \Pi_U^i)$: Upon receiving two messages m_0 and m_1 and an instance Π_U^i from $\mathcal{A}$, a bit b is flipped and $\mathcal{E}_{\mathsf{sk}_i}(m_b)$ is given to $\mathcal{A}$ where sk_i is the session key of Π_U^i.

Finally, in the experiment, $\mathcal{A}$ outputs a bit b' and wins if $b' = b$.

Definition 3.8. We say a protocol π has *tight IND-CCA with respect to* an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the game of π with the $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}$ and given encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is negligibly close to $\frac{1}{2} + \frac{1}{2N}$, where N is the size of low-entropy secret set.

[8]This is a standard result of secure PAKE protocols; viz., [BPR00, KOY01].

Authenticated key exchanges based on high-entropy secrets (e.g., symmetric-key based key exchange and public-key based key exchange) easily imply the *tight IND-CCA* property, because the probability that $\mathcal{A}$ breaks the protocol is negligible. However, this is *not* trivially true in the case of password-based key exchange where passwords are *low-entropy secrets* because the probability that $\mathcal{A}$ breaks the protocol is non-negligibly high. However, it can be shown that most of existing solutions satisfy the tight IND-CCA property in a *non-black box* way.

3.5.2 Generalizing Passwords as Low-entropy Secrets and Adding Perfect Blindness

In this section, we modify a password-based authenticated key exchange to obtain a blind key exchange based on low-entropy secrets (BKE-LS). Our main task is providing a perfect blindness by breaking the binding between secrets (e.g., passwords, wishes) and user IDs. Blind key exchange based on low-entropy secrets is obtained by adding perfect blindness to a password-based authenticated key exchange so that it will provide *no* entity authentication. Here, we focus on providing *anonymity*; however, we do add back entity authentication as the final step of our protocol design (viz., the next section).

In the password-based authenticated key exchange protocol, authentication is provided upon the assumption that there is a binding between *user and user ID* (a user has a unique ID value), and a binding between *user and password* (each user has one password). (Typically these bindings are the result of the user registration process.) Therefore, a user identified by an user ID can be authenticated by password verification. In contrast, in our problem the low entropy secret, namely the "wish", is not used for user authentication. In particular, a user's wish is not necessarily fixed or registered in advance. Here, we generalize passwords as low-entropy secrets and we call them "wishes". We allow a user to use a different secret (i.e., a wish) for each execution and remove the restriction that the low-entropy secret has to be initialized prior to protocol execution (as in the case of passwords).

Finally, to provide *perfect blindness*, we remove the binding between user IDs and secret wishes by breaking the connection between user and user ID. In particular, we let a user have a new pseudonym instead of its (real) user ID for the ID field in each execution of the protocol. Hence the user ID field does not reveal anything about either the user or the secret (i.e., wish) used.[9]

To provide wish unlinkability in our solution, we introduce the notion of forward security with respect to execution transcripts. Intuitively, for any particular secret wish w, the adversary should not be able to tell whether an execution transcript has resulted from input w.

Definition 3.9. We say a key exchange protocol has *forward security with respect to transcripts* if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the following game is negligibly close to $\frac{1}{2}$: (1) Two different secrets s_0, s_1 are randomly selected from $\mathcal{W}$ and given to $\mathcal{A}$. (2) A coin bit b is flipped and two users U_1 and U_2 are selected. If $b = 0$, an execution between U_1 and U_2 on s_0 is simulated and the

[9]In fact, the real user ID is never used in an execution of the BLK-LS protocol. However, the real user ID will be added in a privacy-preserving way later in the last step of our protocol.

transcript is given to $\mathcal{A}$. If $b = 1$, an execution between U_1 and U_2 on input secret s_1 is simulated and the transcript is given to $\mathcal{A}$. (3) Finally, $\mathcal{A}$ outputs b' and we say $\mathcal{A}$ wins if $b' = b$.

Theorem 3.3. *If a protocol π is a secure PAKE protocol, a BKE-LS protocol π' obtained from π has forward security with respect to transcripts.*

The proof of this theorem is similar to the of Theorem 3.1 and hence is omitted.

3.5.3 Final Step of Building a Privacy-enhanced Matchmaking Protocol

The compiler transforming a PAKE protocol π into a BKE-LE protocol π' and then into a PMM protocol π'' is illustrated in Figure 3.3. In this section, we briefly describe the last transformation from a BKE-LE protocol to a privacy-enhanced matchmaking protocol. The compiler essentially adds secure authentication between two parties, U and U'. User U runs BKE-LS π' until it computes a session key sk. Then, U computes a digital signature σ on transcripts of the execution of π' (i.e., ordered concatenation of all the messages sent and received during the execution) and its real identity information including its real identity, its public key and the certificate of the public key. Further, user U encrypts all the transcript, its real identity and the signature σ with key sk, and sends the ciphertext to party U' with whom U interacted during the execution of π. Upon receiving a ciphertext from U', U decrypts it with key sk and, if the plaintext is valid, U verifies that (1) the decrypted transcript is the same as the original, and (2) the digital signature is valid using public key of U'. If the verification is all correct, U accepts U' as a matching partner. Otherwise, U accepts no one.

Theorem 3.4. *If π is a secure PAKE protocol, then protocol π'' obtained by applying the compiler of Figure 3.3 to π is a secure privacy-enhanced matchmaking protocol.*

Proof. We give a sketch of the proof that each security property of privacy-enhanced matchmaking protocol is satisfied.

Impersonation Resistance. If there exists an adversary $\mathcal{A}$ that can break the impersonation resistance property of π'' with non-negligible probability δ, then we can easily construct an algorithm $\mathcal{F}$ that breaks the underlying signature scheme Σ with a probability at least δ. Basically, $\mathcal{F}$ simulates a view for $\mathcal{A}$ and outputs a forged signature σ', whenever $\mathcal{A}$, impersonating V to an honest player T, outputs a forged, but valid signature σ' for a uncorrupted user V. Then, the probability:

$$\Pr[\mathcal{F} \text{ forges a valid signature } \sigma' \text{ with respect to } V\text{'s public key}]$$

is at least $\Pr[T \text{ accepts } V]$, which is equal to $\delta(k)$. Since we assumed that $\delta(k)$ is non-negligible, it contradicts the assumption of security of the underlying digital signature scheme Σ.

Detector Resistance. If there exists an adversary $\mathcal{A}$ that can break the detector resistance property of π'' (*nb.* in a single interaction) with probability $\frac{1}{2} + \frac{1}{2|\mathcal{W}|} + \delta$ for a non-negligible function $\delta(k)$, then we can construct an algorithm $\mathcal{B}$ that breaks the *tight IND-CCA* property of π. $\mathcal{B}$ is given $\mathsf{Execute}, \mathsf{Send}, \mathsf{Reveal}$ and $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}$ and proceeds as follows:

Compiler

Let k be a security parameter. Let $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme which is existentially unforgeable against adaptive chosen-message attack. Let $\{PK_{U_i}, SK_{U_i}\}_{U_i \in \mathcal{U}}$ be a list of public/secret key pairs generated from $\mathsf{Gen}(1^k)$, and assume $\mathcal{U}$, $\{PK_{U_i}\}_{U_i \in \mathcal{U}}$ is publicly-known. Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a symmetric-key encryption scheme that is IND-CCA secure.

The Protocol π': In π, let ID_U be a variable indicating the identity of user U and pw_U be a variable indicating the password of U, and pid be a variable indicating partner ID. Given the input values, namely a wish w, a pseudonym I and a partner's pseudonym pI, user U verifies if the partner's pseudonym pI is valid (signed by the matchmaker). If it is valid, user U sets $ID_U = I$, $pw_U = w$, and $\mathsf{pid} = pI$ and runs protocol π on those inputs.

The Protocol π'': In π', if U terminates accepting a session key sk, U keeps sk. Otherwise, U obtains a random key r through $\mathcal{G}(1^k)$, and sets $\mathsf{sk} = r$. Given sk, U performs the following additional steps:

1. Let $\mathcal{T}$ be a concatenation of messages that U has sent and received during the execution of π'. U computes:
   (a) a signature σ by signing a message $\mathcal{T}||U$, where $||$ denotes a concatenation of messages (i.e., $\mathsf{Sign}_{SK_U}(\mathcal{T}||U)$).
   (b) a ciphertext C by encrypting a plaintext $M = \mathcal{T}||U||\sigma||\mathsf{info}$ with sk (i.e., $\mathcal{E}_{\mathsf{sk}}(M)$), where info is U's information that includes U's public key and the certificate of the public key.
2. U sends ciphertext C to a partner whose pseudonym is pI.
3. Upon receiving a ciphertext C' from partner pI, U decrypts it with sk, obtains $\mathcal{T}'||U'||\sigma'||\mathsf{info}'$, and proceeds as follows:
   (a) If there is no public key for U' or $\mathcal{T}' \neq \mathcal{T}$, U terminates with a private output $\perp$. Otherwise, U verifies σ' by computing $\mathsf{Vrfy}_{PK_{U'}}(\mathcal{T}'||U', \sigma)$.
   (b) If the signature is not valid, U terminates with a private output $\perp$. Otherwise U terminates with a private output U' (i.e., U accepts U' as a matching partner).

Figure 3.3: Compiler to be applied to PAKE protocol π to yield privacy-enhanced matchmaking protocol π''.

1. $\mathcal{B}$ uses its own oracle Send to initiate an instance Π' for a new random identity I' and simulates $\mathcal{A}$'s view until the instance outputs a session key sk in the execution.

2. To simulate the last outgoing message (of π''), $\mathcal{B}$ carries out the following actions:

   (a) Obtain the secret pw' of I' by calling $\mathsf{Corrupt}(I')$.

   (b) Choose two different users U_0 and U_1 from $\mathcal{U}$ at random.

   (c) For each case of $b = 0$ and $b = 1$:

      i. Compute a signature σ_b by signing a message $\mathcal{T}||U_b$, where $\mathcal{T}$ is the transcript of Π'.

      ii. Compose a message $m_b = \mathcal{T}||U_b||\sigma_b||\mathsf{info}_b$ where info_b denotes user U_b's information.

   (d) Then, obtain a challenge ciphertext $\mathcal{C}$ by calling $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}(m_0, m_1, \Pi')$ and finish the interaction by sending the last message $\mathcal{C}$ to $\mathcal{A}$.

3. Finally, $\mathcal{B}$ gives (U_0, U_1) and pw' to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

The simulation by $\mathcal{B}$ is perfect from $\mathcal{A}$'s perspective for the following two reasons. First, the parts of π' are simulated by asking queries to Send oracle. Second, for the part of π'' (i.e., producing the ciphertext $\mathcal{C}$), $\mathcal{B}$ itself learns the secret of I' via $\mathsf{Corrupt}$ oracle query and so $\mathcal{B}$ computes a correct form of plaintext message (which is perfect since U_0 and U_1 are totally independent from I', the pseudonym used in π') and obtains a correct form of ciphertext $\mathcal{C}$ via the $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}(m_0, m_1, \Pi')$

query. Also, since $\mathcal{B}$ queries $\mathsf{Corrupt}(I')$ only after $\mathcal{B}$ finishes queries to the Send oracle, Π' (i.e., instance of I') is fresh. Moreover, by the definition of on-line attacks [BPR00, KOY01], $\mathcal{B}$ makes only one on-line attack.

For the analysis, let Enc_0 denote the case that $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}$ oracle returns encryption of m_0 and Enc_1 denote the case that $\mathsf{Test}_{\mathsf{IND\text{-}CCA}}$ oracle returns encryption of m_1. Then, since we have

$$\begin{aligned} \Pr[\mathcal{B}=0|\mathsf{Enc}_0] &= \Pr[\mathcal{A}=0|\mathsf{Enc}_0], \text{ and} \qquad (3.1)\\ \Pr[\mathcal{B}=1|\mathsf{Enc}_1] &= \Pr[\mathcal{A}=1|\mathsf{Enc}_1], \qquad (3.2) \end{aligned}$$

the probability that $\mathcal{B}$ wins equals the probability that $\mathcal{A}$ wins, which is non-negligibly higher than $\frac{1}{2}+\frac{1}{2|\mathcal{W}|}$ (by the assumption), and it contradicts the fact that π has the *tight IND-CCA* property with respect to $(\mathcal{G},\mathcal{E},\mathcal{D})$. (If π has the *tight IND-CCA* property, an on-line attack can be successful only with a probability negligibly close to $\frac{1}{2}+\frac{1}{2|\mathcal{W}|}$).

Security against off-line Adversaries. Matching-result privacy, wish unlinkability and user unlinkability are clearly satisfied by security properties of the underlying PAKE (and so BKE-LS) protocol. In particular, matching-result privacy is guaranteed by result privacy of π. Wish unlinkability is preserved due to forward security with respect to execution transcripts and perfect blindness of π'. Finally, user unlinkability is obtained by perfect blindness of π', forward security and tight IND-CCA property with respect to $(\mathcal{G},\mathcal{E},\mathcal{D})$ that π has. $\square$

3.6 PAKE security implies forward security

In this section, we show that any password-based authenticated key exchange protocol secure in the standard model (or in the non-corruption model as opposed to the corruption model where password corruption is allowed) is also forward secure in weak-corruption model (where compromise of a password is allowed but complete control over users is not allowed). The formal definition of PAKE is provided in Section 4.4.1. Recall that we only consider off-line dictionary attackers for forward security.

Theorem 3.1. *If a protocol π is a secure password-based authenticated key exchange protocol in the non-corruption model, π is forward secure against off-line attackers in the weak-corruption model.*

Proof. Assume that there exists an adversary $\mathcal{A}$ breaking forward security of π in the weak-corruption model. Because we consider only off-line dictionary attackers for forward security, $\mathcal{A}$ is not allowed to access Send oracle. Then, by the assumption, $\mathcal{A}$ attacks π *in the weak corruption model* and succeeds in the experiment with probability $\frac{1}{2}+\delta(k)$, for a non-negligible function $\delta(k)$. Given $\mathcal{A}$, we can construct an adversary $\mathcal{A}'$ that attacks protocol π *in the non-corruption model* by eavesdropping on the executions of π, and then outputs a pair of a password and a user, (pw, U), for some user U that $\mathcal{A}'$ has chosen, such that probability of pw being U's correct password is non-negligibly higher than $\frac{1}{|\mathcal{W}|}$. The existence of $\mathcal{A}'$ is sufficient to show that protocol π is an insecure password-based authenticated key exchange protocol in the non-corruption model. The reason for this is as follows: informally, if there exists an adversary $\mathcal{M}$ who can correctly guess the password of any user of $\mathcal{M}$'s choice with probability non-negligibly higher than $\frac{1}{|\mathcal{W}|}$, then we can construct an on-line adversary $\mathcal{O}$ who uses $\mathcal{M}$ to break π in the non-corruption model, with *one* on-line attack, and achieves an advantage non-negligibly higher than $\frac{1}{|\mathcal{W}|}$. Basically, $\mathcal{O}$ simulates the view of $\mathcal{M}$ and when $\mathcal{M}$ outputs (pw, U) for some user U, $\mathcal{O}$ carries out an on-line attack against U with password pw and asks for a Test query for the instance. Given a challenge key as a response to a Test query, if the key is the same as the key that $\mathcal{O}$ computed in the on-line attack, $\mathcal{O}$ outputs 1. Otherwise, $\mathcal{O}$ outputs 0. Then, if pw was a correct password for U, $\mathcal{O}$ always succeeds. Otherwise, $\mathcal{O}$ succeeds with probability exactly $\frac{1}{2}$. Therefore, the advantage of $\mathcal{O}$'s in breaking protocol π in the non-corruption model with *one* on-line attack is non-negligibly higher than $\frac{1}{|\mathcal{W}|}$ (the advantage of $\mathcal{O}$ obtained by using $\mathcal{M}$ is the difference between the probability that $\mathcal{M}$ guesses a password correctly and the probability $\frac{1}{|\mathcal{W}|}$). Then, it leads a contradiction and the proof is complete.

Now, let's see how $\mathcal{A}'$ can guess a password of a user with probability non-negligibly higher than $\frac{1}{|\mathcal{W}|}$, by using $\mathcal{A}$. Adversary $\mathcal{A}'$, playing in the non-corruption model, has access to the $\mathsf{Execute}$ and Reveal oracles, and $\mathcal{A}$, playing in the weak-corruption model, has access to the $\mathsf{Execute}, \mathsf{Reveal}, \mathsf{Corrupt}$ and Test oracles. Adversary $\mathcal{A}'$ proceeds as follows:

1. Let L be a maximum number of users that $\mathcal{A}$ will ask for $\mathsf{Execute}$ query (L is polynomial in k since $\mathcal{A}$ is a PPT adversary). Choose an integer ℓ from $\{1, ..., L\}$ at random.

2. Whenever $\mathcal{A}$ asks a query in a form of $\mathsf{Execute}(C, i, S, j)$,[10] if C is the ℓ-th new user that has been queried in such a form, keep C as a target user T and answer to $\mathcal{A}$ by forwarding the same query to its own oracle and returning the response from the oracle. Otherwise, choose a random password (or find a stored password for C, if one exists), and simulate an execution according to π, and return the resulting transcript to $\mathcal{A}$.

3. Whenever $\mathcal{A}$ asks a query in a form of $\mathsf{Corrupt}(U)$, if U is not T, find a password chosen for U and answer with it (if there is no record, answer with a random password and record it). Otherwise (if U is T), choose a random password pw_1 and answer with pw_1.

4. Upon receiving a $\mathsf{Test}(\Pi^i_{U'})$ query from $\mathcal{A}$, adversary $\mathcal{A}'$ proceeds as follows:

   - If U' is not T, $\mathcal{A}'$ selects a random password pw and outputs (pw, T). Let $\mathsf{NoUse}_{\mathcal{A}}$ denote this event.

   - Otherwise, if U' is T, $\mathcal{A}'$ proceeds as follows:

     (a) Flips a random coin b. If $b = 0$, $\mathcal{A}'$ chooses a random session key r and provides it to $\mathcal{A}$.

     (b) If $b = 1$, $\mathcal{A}'$ sends a $\mathsf{Reveal}(\Pi^i_{U'})$ query to its own oracle, obtains the real session key $\mathsf{sk}^i_{U'}$, and provides $\mathcal{A}$ with it.

     (c) If $\mathcal{A}$ aborts, $\mathcal{A}'$ chooses a random password pw_2, different from pw_1, and outputs (pw_2, T). Let $\mathsf{Abort}_{\mathcal{A}}$ denote this event.

[10] S is a server who keeps all the passwords of clients C.

(d) Finally, when $\mathcal{A}$ outputs b', if $b' = b$, adversary $\mathcal{A}'$ outputs (pw_1, T). Otherwise, $\mathcal{A}'$ chooses a random password pw_3, different from pw_1, and outputs (pw_3, T).

Next, we analyze the probability that $\mathcal{A}'$ correctly guesses the password of a user. For a better understanding, we introduce some additional notation. We let $\mathsf{Succ}_{\mathcal{A}}$ (resp., $\mathsf{Succ}_{\mathcal{A}'}$) denote the event that $\mathcal{A}$ (resp., $\mathcal{A}'$) succeeds in the experiment of breaking forward security of π (resp., guesses a correct password of a user). Also, we let pw_1 (resp., pw_2, or pw_3) denote the event that T's password is equal to pw_1 (resp., pw_2, or pw_3).

Then, the success probability of $\mathcal{A}'$ is the following:

$$\begin{aligned}
\Pr[\mathsf{Succ}_{\mathcal{A}'}] &\geq \Pr[\mathsf{Succ}_{\mathcal{A}'}|\mathsf{NoUse}_{\mathcal{A}}] \times \Pr[\mathsf{NoUse}_{\mathcal{A}}] + \\
&\quad \Pr[\mathsf{Succ}_{\mathcal{A}'}|\overline{\mathsf{NoUse}_{\mathcal{A}}}] \times \Pr[\overline{\mathsf{NoUse}_{\mathcal{A}}}] \\
&\geq \frac{1}{|\mathcal{W}|} \times \left(1 - \frac{1}{L}\right) + \Pr[\mathsf{Succ}_{\mathcal{A}'}|\overline{\mathsf{NoUse}_{\mathcal{A}}}] \times \frac{1}{L} \\
&= \frac{1}{|\mathcal{W}|} + \frac{1}{L} \cdot \left(\Pr[\mathsf{Succ}_{\mathcal{A}'}|\overline{\mathsf{NoUse}_{\mathcal{A}}}] - \frac{1}{|\mathcal{W}|}\right) \qquad (3.3)
\end{aligned}$$

Then, we can bound the probability $\Pr[\mathsf{Succ}_{\mathcal{A}'}|\overline{\mathsf{NoUse}_{\mathcal{A}}}]$ as follows (*nb.*, for simplicity, we omit the conditional event $\overline{\mathsf{NoUse}_{\mathcal{A}}}$ for the right hand side) :

$$\begin{aligned}
\Pr[\mathsf{Succ}_{\mathcal{A}'}|\overline{\mathsf{NoUse}_{\mathcal{A}}}] &= \Pr[\mathsf{pw}_2 \wedge \mathsf{Abort}_{\mathcal{A}}] + \\
&\quad \Pr[\mathsf{Succ}_{\mathcal{A}} \wedge \mathsf{pw}_1 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}] + \\
&\quad \Pr[\overline{\mathsf{Succ}_{\mathcal{A}}} \wedge \mathsf{pw}_3 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}]
\end{aligned}$$

Next, we bound each term of the right hand side in the above equation. First, we bound the probability $\Pr[\mathsf{pw}_2 \wedge \mathsf{Abort}_{\mathcal{A}}]$ as follows:

$$\begin{aligned}
\Pr[\mathsf{pw}_2 \wedge \mathsf{Abort}_{\mathcal{A}}] &= \Pr[\mathsf{pw}_2 \wedge \overline{\mathsf{pw}_1} \wedge \mathsf{Abort}_{\mathcal{A}}] \\
&= \Pr[\mathsf{pw}_2 \wedge \overline{\mathsf{pw}_1}] \times \Pr[\mathsf{Abort}_{\mathcal{A}} | \overline{\mathsf{pw}_1} \wedge \mathsf{pw}_2] \\
&= \Pr[\mathsf{pw}_2] \times \Pr[\mathsf{Abort}_{\mathcal{A}} | \overline{\mathsf{pw}_1}] \\
&= \frac{1}{|\mathcal{W}|} \times \Pr[\mathsf{Abort}_{\mathcal{A}} | \overline{\mathsf{pw}_1}] \\
&= \frac{1}{|\mathcal{W}|} \times \left(1 - \Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \overline{\mathsf{pw}_1}]\right)
\end{aligned}$$

Second, we bound the probability $\Pr[\mathsf{Succ}_{\mathcal{A}} \wedge \mathsf{pw}_1 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}]$ as follows:

$$\begin{aligned}
\Pr[\mathsf{Succ}_{\mathcal{A}} \wedge \mathsf{pw}_1 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}] &= \Pr[\mathsf{Succ}_{\mathcal{A}} | \mathsf{pw}_1 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}] \times \\
&\quad \Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \mathsf{pw}_1] \times \Pr[\mathsf{pw}_1] \\
&= \left(\frac{1}{2} + \delta(k)\right) \times \\
&\quad \Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \mathsf{pw}_1] \times \frac{1}{|\mathcal{W}|}
\end{aligned}$$

Finally, the last term of probability $\Pr[\overline{\mathsf{Succ}_{\mathcal{A}}} \wedge \mathsf{pw}_3 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}]$ is bounded as follows:

$$\begin{aligned}
\Pr[\overline{\mathsf{Succ}_{\mathcal{A}}} \wedge \mathsf{pw}_3 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}] &= \Pr[\overline{\mathsf{Succ}_{\mathcal{A}}} | \mathsf{pw}_3 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}] \times \\
&\quad \Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \mathsf{pw}_3] \times \Pr[\mathsf{pw}_3] \\
&= \left(1 - \Pr[\mathsf{Succ}_{\mathcal{A}} | \mathsf{pw}_3 \wedge \overline{\mathsf{Abort}_{\mathcal{A}}}]\right) \times \\
&\quad \Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \overline{\mathsf{pw}_1}] \times \frac{1}{|\mathcal{W}|} \\
&\geq \frac{1}{2|\mathcal{W}|} \times \Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \overline{\mathsf{pw}_1}]
\end{aligned}$$

Let p denote the probability $\Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \overline{\mathsf{pw}_1}]$ and q denote the probability $\Pr[\overline{\mathsf{Abort}_{\mathcal{A}}} | \mathsf{pw}_1]$.

Then, by combining three probabilities that we computed so far, we have:

$$\begin{aligned}\Pr[\mathsf{Succ}_{\mathcal{A}'}|\overline{\mathsf{NoUse}_{\mathcal{A}}}] &\geq \frac{1}{|\mathcal{W}|} \times (1-p) + \\ &\quad \left(\frac{1}{2} + \delta(k)\right) \times q \times \frac{1}{|\mathcal{W}|} + \frac{1}{2|\mathcal{W}|} \times p \\ &= \frac{1}{|\mathcal{W}|} + \frac{q}{|\mathcal{W}|} \times \delta(k) + \frac{(q-p)}{2|\mathcal{W}|} \end{aligned} \tag{3.4}$$

In the event of pw_1, the simulated view for $\mathcal{A}$ is perfect. Therefore, the probability that $\mathcal{A}$ aborts in the conditional event of pw_1 (i.e., q) is negligibly close to 1 (i.e., $q \approx 1$). Also, no matter how close the probability p is to q, q is greater than or equal to p. Therefore, by applying Equation (3.4) into Equation (3.3), we obtain:

$$\begin{aligned}\Pr[\mathsf{Succ}_{\mathcal{A}'}] &\geq \frac{1}{|\mathcal{W}|} + \frac{1}{L} \times \frac{q}{|\mathcal{W}|} \times \delta(k) \\ &\geq \frac{1}{|\mathcal{W}|} + \frac{1}{2|\mathcal{W}|L} \times \delta(k)\end{aligned}$$

which is non-negligibly higher than $\frac{1}{|\mathcal{W}|}$ since L is polynomial in k. This completes the proof. $\square$

Chapter 4

Enhancing Password Privacy of Password-Based Authenticated Key Exchange in the Real World

4.1 Outline of the Chapter

In Section 4.2 we discuss the significance of our attacks in practice. In Section 4.3 we describe the details of the attacks with analytical and experimental evidence. In Section 4.4 we propose a new definition of security. In Section 4.5 we analyze our attacks and conclude with recommendations on ways to bound the probability of failure of PAKE protocols. Our recommendations can be applied in particular to the PAKE protocols currently proposed for TLS [ABC$^+$06].

4.2 Background of Our Attacks

Practical Significance. The significance of the results reported herein, particularly of the *timeout-delay* and *multi-domain* attacks, goes beyond merely providing practical attacks against protocols proven secure in a well-accepted theoretical setting. The security exposures created by these attacks cannot be overlooked given that the effective space of user-chosen passwords, N, is generally small. For example, the NIST Electronic Authentication Guideline - Special Publication 800-63 (Appendix A.3 - Examples), April 2006, recommends that, for "level 1" password security, the password strength should limit an adversary success to $Q^*/N = 2^{-10}$, whereas at "level 2," $Q^*/N = 2^{-14}$, *during a password's lifetime.* Our simulations show (viz., Figure 4.3 below) that the bound of $Q^* = 5$ failed-login attempts of a PAKE protocol proven secure can be circumvented to allow an adversary $200 - 500$ login attempts per domain in a 5 second interval, which would be non-compliant with the NIST guidelines. In a five-domain attack this can be amplified to 2500 attempts in 5 seconds, which would obviously be non-compliant. Even synchronization-delay attacks, if amplified in multiple domains, could result in non-compliance (e.g., with "level 1") password-security guidelines.

4.3 Delay-Based Attacks

In this section we illustrate the inadequacy of counting login failures to bound the security of a PAKE protocol. That is, we show on-line attacks against PAKE protocols for which setting up a specific bound of Q^* for the number of login failures does *not* result in a failure probability of $\varepsilon \leq Q^*/N$ (where N is the size of the password space).

We show further that our attacks can be easily launched in current client-server applications that run in both distributed systems and the Internet. In particular, we argue that the ease of such attacks is enhanced by current multi-tiered applications that require large-scale, multi-threaded constructions of the client-server model for use over the Internet. This represents a significant concern as these protocols have been recently standardized [IEE05].

4.3.1 Counting Attack Queries: Login Requests or Failed Logins?

In a typical Internet-level deployment, PAKE protocols would run in an distributed environment where a multi-tasked server responds to multiple client requests concurrently. Multiple server instances of the PAKE protocol can be distributed over a number of servers controlled by load balancers, such as layer 4-7 (L4-L7) switches in the Internet, in the same manner as in most large-scale, personalized services are implemented currently. In fact, most large-scale, personalized services implemented by Google, Yahoo, Microsoft Hotmail, and Skype follow this model.

Counting Login Requests. In any environment where concurrent login requests are possible, counting such requests as attack queries would require either synchronization of *all* PAKE servers for real-time enforcement of bound Q^* or synchronization of load balancing (L4-L7) switches in the Internet to filter out clients that issue concurrent PAKE requests to individual accounts. Either could lead to substantial, user-visible login delays. For this reason, in such environments it is preferable to limit concurrent login requests to the same account early, at the *client side*, before they reach PAKE servers or Internet switches, by challenging them using CAPTCHAs [vABHL03].

However, exclusive reliance on CAPTCHAs to limit the number of concurrent login requests to the same account is an insufficient solution. An (human) adversary could request multiple logins via multiple clients, redirect the CAPTCHA challenges for those clients to himself, respond to those challenges correctly outside the http sessions, and then pipe the responses concurrently into the corresponding sessions. Note that CAPTCHA challenges do not time out before their http sessions since human-level response delays are not indicative of *automated* attacks, which CAPTCHAs attempt to prevent. Further, two or more human users could collude and amplify concurrent attacks against a PAKE protected account protected by CAPTCHAs. In either case, a concurrent attack would certainly exceed the given bound on Q^*.

The only practical alternative for the enforcement of a bound Q^* on the number of on-line attacks against a user's account would be to require *server-side* enforcement of a bound on *failed login attempts*, which are invariably categorized as an adversary's attack queries. Reasonable bounds of this type can account for legitimate-user errors and, at the same time, allow the authentication of concurrent client requests. Almost all large-scale deployment of personalized services, including by Google, Yahoo, Microsoft Hotmail, and Skype, and *all* theoretical PAKE models, also use this approach.

Counting Login Failures. Counting login failures causes, by definition, a *time delay* between (1) the instance an adversary query is issued and (2) the instance when the authentication of that query is completed *and* counted as a login failure. It is this time delay that can be exploited by an adversary to launch concurrent attacks. However, different provably secure PAKE protocols with similar security bounds react differently to concurrent attacks. We partition the PAKE protocol space into two large classes depending on which entity, client or server, initiates

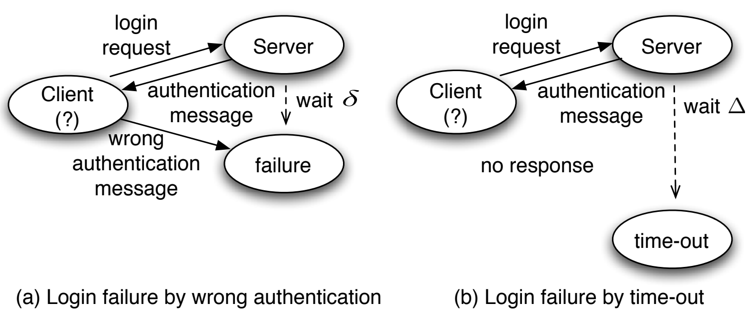

Figure 4.1: Login failures : Incorrect authentication vs. Time-out.

the authentication exchange and illustrate the reaction to two different types of concurrent attacks.

4.3.2 Timeout-Delay Attack

We consider in this section protocols in which the server authenticates first. At a high level, such protocols have the following structure: the client sends some information to which the server responds; at this point the client can tell whether it is interacting with the legitimate server (holding the same password) or not. Then the server sends some information to the client (this can, of course, be piggy-backed on the previous exchange). The server than either (1) receives a response and, if the response is incorrect, counts this session as a a login failure (viz., Figure 4.1-(a)) or (2) waits for a response until a time-out occurs (viz., Figure 4.1-(b)). The crucial point is that by aborting after running the first phase, an adversary gets a password guess that is not counted as a failure until the time-out occurs. In the latter case, the adversary can open multiple client-server instances within a server's timeout interval (denoted by Δ in Figure 4.1-(b)). This, in turn, means that the adversary receives multiple concurrent authentication queries, each of them allowing her to verify a separate password guess, *without raising the login failure counter.*

Client C		**Server** S
(password pw)		(password pw)
	Public: g, h, e, p, g, H, F	
$x \overset{R}{\leftarrow} \mathbb{Z}_q$		
$X \leftarrow g^x$		
$Y \leftarrow h^x g^{pw}$	$\xrightarrow{X, Y, C}$	$\lambda_1, \lambda_2 \overset{R}{\leftarrow} \mathbb{Z}_q$
		$\mu \leftarrow g^{\lambda_1} h^{\lambda_2}$
		$Y' \leftarrow Y g^{-pw}, \sigma \leftarrow X^{\lambda_1} Y'^{\lambda_2}$
		$r \leftarrow F_\sigma(3), \omega \leftarrow E_e[\Sigma; r]$
		$\Sigma \leftarrow H(\mu, X, Y', S, C)$
$Y' \leftarrow h^x, \sigma \leftarrow \mu^x, r \leftarrow F_\sigma(3)$	$\xleftarrow{\mu, \omega, S}$	
$\Sigma \leftarrow H(\mu, X, Y', S, C)$		
Verify $\omega = E_e[\Sigma; r]$		
Abort if verification fails.		
Else: $\tau \leftarrow F_\sigma(2), sk \leftarrow F_\sigma(1)$	$\underbrace{\xrightarrow{\tau}}$	
	(a) No Reply with τ	(b) In case Reply (τ is received):
		Verify $\tau = F_\sigma(2)$
		Abort if verification fails.
		Else: $sk \leftarrow F_\sigma(1)$
		(c) In case of No Reply (τ not received):
		May do nothing
		Wait for Δ until timeout

Figure 4.2: Timeout-delay attack against the JG protocol: a single protocol instance.

Within the time-out delay, an adversary may initiate multiple interactions with distinct server instances, each enabling a different password guess. For example, in the normal (non-attack) mode of operation a client's reply to a server's authentication message arrives to the server after a delay δ, as shown in Figure 4.1-(a). In contrast, in an attack, the client's authentication-message reply does not arrive before a timeout period Δ, if at all, as shown in Figure 4.1-(b). Since, by definition, $\delta << \Delta$, an adversary has numerous chances to verify password candidates, one per client-server instance, without responding to authentication messages initiated by a server instance in each protocol run. Meanwhile each server instance must wait until the timeout internal is exhausted to count its client's lack of reply as a login failure and increase the failure count. Note that the time delay within which an adversary can open many protocol instances and verify multiple password candidates can be substantially larger than Δ; i.e., it can be $\Delta + \gamma$, where γ denotes the time interval from the adversary's protocol-initiation message to the server's reply with an authentication message to the adversary.

Timeout-delay attacks succeed for *all* existing PAKE protocols in which server-authentication occurs first. Figure 4.2 illustrates an attack on the JG protocol [JG04].[1]

An adversary can pose as a client in the JG protocol and open many concurrent sessions similar to the one illustrated in Figure 4.2. In each such session it can pick a random password from the password space and then run the client-side of the protocol honestly. Upon receiving the second message of the protocol from the server (computed using the correct password), the adversary can tell whether its password guess was correct. Message flow (a) of this figure shows that the adversary does not reply to the authentication message issued by a server instance with message τ. The server instance will then have to wait for some timeout interval Δ, as

[1]Although the notation we use in this figure is very close to the original one, we do not cover all protocol flows of the JG protocol. We refer the readers to [JG04] for the detailed description of this protocol.

shown in box (c) of the figure, rather than performing the normal authentication verification illustrated in box (b), to determine whether an adversary attempts to impersonate a legitimate client. Obviously, a server could not possibly conclude that an adversary impersonates a legitimate client *before* performing that verification. Since the precondition of that verification is provided by adversary replies modeled as Send queries, adversary impersonation attacks, called **ITri** are counted as Send queries in the JG proof, regardless of whether authentication verification succeeds. Hence, a server has to wait until a Send query arrives to determine that an attack in taking place. This enables the timeout delay attack described above despite the correct proof provided for the JG protocol in the standard model. Similar attacks can be generated using the PAK-Z+ [Mac02] protocol, whose security was proved correctly in the random-oracle model.

Figure 4.3 shows the experimental results of timeout-delay attacks against the JG and PAK-Z+ [JG04, Mac02] protocols. These experiments are performed on a Windows platform and are described in more detail in Section 4.7. We set the practical login-failure limit to $Q^* = 5$ and synchronize the counting of login failures across server instances[2] simply by using a global variable for multi-threads. We have an adversary attempt to open 1000 concurrent sessions for the same account, and vary the length of the time-out interval, Δ, in each experiment. As we increase Δ from 500 ms to 5000 ms in the experiment, a growing number of authentication messages that exceed the failure-limit bound $Q^* = 5$ are obtained by the adversary who could

[2]In the real world, there could be a time delay for synchronizing such a count over multiple server hosts in real time. See the following subsection.

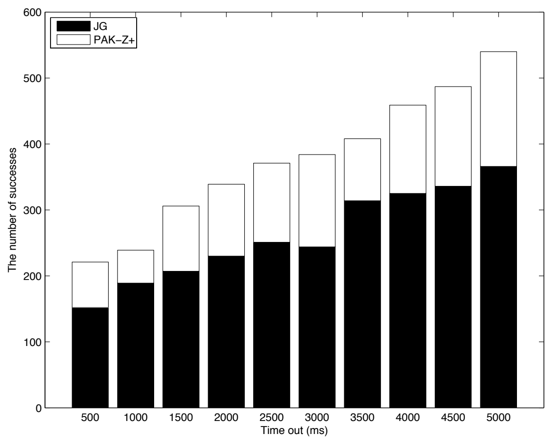

Figure 4.3: Experimental results of timeout-delay attacks against two provably secure PAKE protocols. (X-axis shows the different duration of time-out Δ in milliseconds, while y-axis shows the number of server authentication messages that allow password-candidate verification by an adversary.)

verify a correspondingly increased number of password candidates. For example, when $\Delta = 500\ ms$, protocol JG allows 152 guesses and PAK-Z+ 221 guesses on average, far exceeding the bound $Q^* = 5$. However, when $\Delta = 5000\ ms$, protocol JG allows 336 guesses and PAK-Z+ 540 guesses on average further exceeding the bound $Q^* = 5$. If the length of the time-out interval is set to a high value, or not set at all, then *all* adversary's concurrent requests for authentication to servers will be satisfied, thereby leading to a massive circumvention of the login-failure bound Q^*. In such attacks it is unlikely that any low-entropy password will be safe. The results of a wider range of experiment parameters for timeout-delay attacks against JG and PAK-Z+ protocols are shown in Section 4.7.

As we show in Section 4.3.4, the problem is magnified in multi-domain settings.

4.3.3 Synchronization-Delay Attack

A way to avoid timeout-delay attacks, is to require the client to send the initial authentication message to the server. This message allows the server to verify client's knowledge of the account password before the server replies with its own authentication message to the client. This authentication exchange, which is similar to the "client pre-authentication" in Kerberos version V and its derivatives (e.g., DCE), denies an adversary's client the possibility of avoiding to reply to a server's authentication message and exploiting the timeout delay in login-failure counting to launch concurrent attacks. Practical PAKE protocols that prescribe client initiation of the authentication exchange have been proposed in the past (e.g., KOY [KOY01]).

However this introduces a different type of complication. In processing a client's pre-authentication message, a server instance must first check whether the failure count for an account/password exceeds the bound Q^* and, if the bound is not exceeded, whether the client's pre-authentication message is valid (i.e., it reveals client's knowledge of the account password).[3] To check whether the current failure

[3]Note that if the failure-count check is performed *after* the validity check for the pre-authentication message, then adversary's clients could always verify multiple password guesses concurrently by observing a delay in the server instance reply. Such a delay would indicate a failed password guess since a correct password would not require a failure-count check and hence would not incur any synchronization delay. (Artificially delaying a server instance's authentication-message reply that corresponds to a correct password guess could not prevent an adversary's verification of multiple concurrent guesses that exceed bound Q^* since one of adversary's clients would eventually receive the positive reply among all negative ones.)

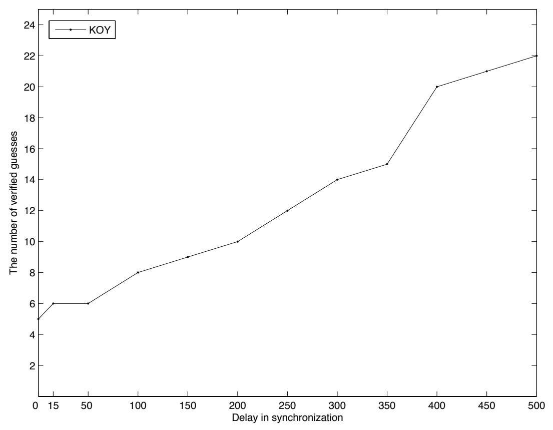

Figure 4.4: Experimental results of synchronization-delay attacks against the KOY protocol. (X-axis shows the delay in synchronization in milliseconds, while y-axis shows the number of server authentication messages that allow password-candidate verification by an adversary.)

count for a password exceeds bound Q^* requires the *exclusive locking* of the failure-count variable even for concurrent *read* operations – not just for concurrent *write* or *read and write* operations – by multiple clients. (Shared-read operations would allow multiple concurrent guesses of password by an adversary's clients before a server instance would be able to signal that the failure count has exceeded bound Q^*.) Unfortunately, placing exclusive locks on a failure-count variable associated with a password to ensure correct counting incurs non-negligible *synchronization delays* whenever replicas of server registries, which store account/password information, are concurrently accessible to PAKE protocol instances. While an exclusive-lock request propagates to all registry replicas, concurrent password guesses can still be made by an adversary's clients, since their server instances can still read failure-count copies of yet-to-be-updated registry replicas.

In Figure 4.4 we illustrate the experimental results of our synchronization-delay attacks again the Katz, Ostrovsky, and Yung (KOY) protocol [KOY01]. These experiments are also performed on a Windows platform. We set the practical login-failure limit to $Q^* = 5$ but allow possible delays in synchronizing the login failure counts across server instances. We have an adversary attempt to open 1000 concurrent sessions for the same account and vary the delay in each experiment. As we increase the delay from 0 ms to 800 ms[4] in the experiment, a growing number of authentication messages that exceed the failure-limit bound $Q^* = 5$ are obtained by the adversary who could verify a correspondingly increased number of password candidates. When the delay is set as 15 ms, the protocol at last loses the failure-limit bound by allowing one more instance to send out the server authentication message than the bound. As we can see in Figure 4.4, the KOY protocol allows 20 guesses when the delay is 400 ms and 26 guesses when it is 800 ms. We expect the number of verified guesses will increase more as we enlarge the delay in synchronization.

Simple, Obvious Solutions are Ineffective. Simple, obvious solutions to remove synchronization delays by serializing access to account/password information (e.g., by testing and updating a failed-login counter in a, possibly distributed, critical section), do not work in large-scale deployment of PAKE protocols. For example, restricting the access of concurrent protocol instances to a *single server* that maintains a critical section for the failed-login counter and its bound Q^* could serialize

[4]The experiments are done with 5 ms scales between 0 and 50 ms interval, while done with 100 ms scales between 100 and 800 ms interval for visuality. We set the upper end of the delay interval as 800 ms by assuming the average message delay is between 50 and 200 ms. Note that the delay interval would be typically $\{2T, 4T\}$ where T is the average message delay [CLLZ05].

access to the failed-login counter and thus eliminate exploitable synchronization delays, but would be highly impractical. Although the server availability could be assured by mirroring the server's content, in Internet-scale deployment login responsiveness would drop below any acceptable level since the *single server* would have to respond to *all* login requests; e.g., delays would reach 1.5 - 2 minutes at login rates that often exceed 200 requests per second and typical server response rates of about 500 ms per login request. Similarly, small-scale solutions where a single server allows only a single login session at a time would obviously not work in Internet-scale deployment. Alternate solutions that seek to serialize access to the failed-login counter and its bound Q^* in a critical section distributed across multiple servers, which would also eliminate exploitable synchronization delays, would not improve login responsiveness: the *lower bound* on the access latency of all distributed critical-section implementations resilient to failure is between 2T and 4T [CLLZ05], where T is the average message delay in the Internet (currently between 50 and 200 ms), plus the delay in accessing and verifying password information on any server replica. At login rates that often exceed 200 requests per second such implementations would also limit user-login responsiveness by causing 1.5 - 2 minute delays (e.g., 4*200 ms latency per request plus 100ms request processing would mean that, again, a typical request would be processed in 500ms.) Other synchronization solutions, possibly based on hierarchical account locking, may be effective. (The evaluation of such solutions are beyond the scope of this paper, however.)

4.3.4 Multi-Domain Attacks

Enforcement of the bound Q^* on the adversary's queries to different user accounts in $n > 1$ different domains having the same (or related) passwords is all but impossible since an adversary can launch a concurrent attack against that password *without* exceeding bound Q^* in any single domain. Thus an adversary could effectively issue nQ^* attack queries, far exceeding any reasonable security bound Q^* for large n. Unfortunately, it would be virtually impossible to enforce bound Q^* across multiple domains, since (1) it would be impossible to know all domains where a user may have an account unless users would reveal this information, which users might not do for privacy reasons; and (2) it would be necessary to authenticate every cross-domain bound update. This would not only amplify adversary-exploitable synchronization delays but would also cause unacceptable login delays for such users. An adversary could further amplify her attack by verifying $n\dot{m}$ passwords using n clients per domain in m domains and exploiting either timeout-based or synchronization-based delays to launch concurrent attacks. Of course, multi-domain attacks could be eliminated if users would resist the temptation to choose the same password in multiple domains.

4.4 Security Definitions for PAKE Protocols

As illustrated above, in practice, the number of on-line adversary attacks (i.e., password guesses) far exceeds the bound of consecutive log-in failures. As we discussed in the Introduction, these attacks have been overlooked in all PAKE protocol analyses to date because the current adversary models neither count login failures nor enforce a bound on them. The current formal definition for PAKE security simply correlates the probability of success of the adversary to the number of online attacks s/he can launch.

We conclude that we would be better served by a definition that *concretely* bounds the adversary's advantage with a hard bound ε. We then analyze protocols in light of *both* timeout-based and synchronization-based delays to make sure the probability of failure stays below ε.

We stress that in doing so, the current theoretical definition remains crucially important, as it allows us to tightly bound the failure probability to the number of *actual* queries made by an adversary. The second part of the task (which has been overlooked so far) is to make sure to get a correct estimate on the number of such queries.

Note that we do not claim to make fundamental changes to the definition of security of PAKE protocols, but rather to point out that more concrete definitions are necessary for security engineers to understand and ensure the security of PAKE protocols more exactly in the real world.

4.4.1 The Formal Definition

We briefly review the security notions for PAKE protocol models [BPR00, KOY01].

Protocol Entities and Identifiers. Each protocol participant is either a client (denoted by C) or a server (denoted by S). Each client selects a secret password from a small set $\mathsf{Password}$ of size N and registers it with a server. For simplicity, we assume that the distribution of the password is uniform. Each participant can run as many protocol instances as s/he likes, and each client instance is denoted by C^i and server instances by S^j, for any integers i and j.

Each protocol instance is denoted by U^i, for any i, and has a unique session identifier denoted by sid_U^i. Given a participant in a protocol instance U^i, we denote by pid_U^i the identifier of a partner with whom the participant intends to establish a shared session key, sk_U^i. After a participant computes a session key, the participant may accept it by setting a boolean variable $\mathsf{accept}_U^i = true$. Acceptance of a session key occurs only once per protocol instance, and the lifetime of that key ends when the session terminates.

Basic Oracle Queries. The adversary has complete control over the communication network and his behavior is modeled via queries to a set of oracles, briefly reviewed below.

- $\mathsf{Execute}(C^i, S^j)$ runs an execution of the protocol between new instances C^i and S^j and outputs the transcript of the protocol execution. This models passive eavesdropping.

- $\mathsf{Send}(U^i, m)$ sends message m to instance U^i (if there is no instance U^i, initiates a new one) and returns the response according to the protocol. This models active attacks.

- $\mathsf{Reveal}(U^i)$ outputs the session key of instance U^i, if U^i has accepted a session key.

- $\mathsf{Test}(U^i)$ measures the adversarial advantage in attacking the protocol and is queried for an accepted session key in instance U^i. A random bit b is flipped; if $b = 0$, a random session key is returned. If $b = 1$, the session key of instance U^i is returned.

Partnering. We say instances C^i and S^j are *partnered* if (1) $\mathsf{sid}_C^i = \mathsf{sid}_S^j \neq NULL$, and (2) $\mathsf{pid}_C^i = S$ and $\mathsf{pid}_S^j = C$.

Freshness. We say an instance is fresh if the instance has accepted and neither of the instance and the partnered instance have been queried via Reveal.

Correctness. We always assume a *correct* protocol, where this is defined as follows: if two *partnered* instances C^i and S^j accept, then they must both conclude with the same session key.

Defining Security. We say the adversary *wins in the* Test *experiment* if the adversary correctly guesses the bit b used in the Test oracle and the instance queried at the Test oracle is fresh. More formally:

We say an adversary *wins in the* Test *experiment* if the following conditions are satisfied: (1) the adversary queries Test oracle for a fresh instance U^i which has accepted a session key sk_U^i; (2) the adversary outputs a bit b' and b' is equal to b used in the Test oracle. This event is denoted by Succ and the advantage of the adversary $\mathcal{A}$ attacking protocol P is defined as:

$$\mathsf{Adv}_{\mathcal{A},P}(k) = 2 \cdot \Pr[\mathsf{Succ}] - 1, \tag{4.1}$$

where k is the security parameter and the probability is taken over the random coins used by the oracle queries and the adversary. Now, we can provide a new security definition for PAKE protocols as follows:

Definition 4.1. We say a protocol P is a *secure password-based authenticated key exchange protocol* if for all password spaces of size N and for all polynomial-time adversaries $\mathcal{A}$ making at most Q Send queries we have that $\mathsf{Adv}_{\mathcal{A},P}(k) \leq \frac{Q}{N} + negl(k)$ where $negl(\cdot)$ is a negligible function.

4.4.2 A Concrete Security Definition

Notice that the above definition simply bounds the probability of failure of the protocol as a function of number of queries made by the adversary, and the size of the dictionary space. It does not put a "hard" concrete bound on such failure probability.

What follows is a concrete-security version of the above definition. We consider a specific adversary $\mathcal{A}$ which runs in time T, and consider its advantage in a specific instantiation of the protocol (notice that we remove "asymptotic" security parameters and negligible functions here). We also impose a bound ε on the acceptable probability that the protocol can fail.

Definition 4.2. We say a protocol P is a (T, ε)*-secure password-based authenticated key exchange* protocol if for all probabilistic adversaries $\mathcal{A}$ which run in time T, we have that $\mathsf{Adv}_{\mathcal{A},P} \leq \varepsilon$.

In practice one fixes an reasonable bound on the computation time T of the adversary (say sufficient not to break Diffie-Hellman on groups of 160-bit prime order), and an acceptable probability of failure ε and determines under which conditions the protocol satisfies Definition 4.2. In particular one could try to set the size N of the password space so as to match the level of security desired.

If a protocol is proven secure according to the theoretical Definition 4.1, then bounding the probability of failure with ε in a concrete instantiation means that one must bound $Q/N \leq \varepsilon$.[5] Usually this is done by bounding the number Q^* of failed login attempts allowed before locking a user's account. Since this would seem to bound the number of possible queries of an adversary, it would therefore seems that setting $Q^* \leq \varepsilon \cdot N$ would suffice. But as we have shown in the previous section this is not that simple, and in some cases it is not going to work.

In the next section we show better ways to analyze PAKE protocols, in the face of timeout-delay and synchronization-delay attacks, so that a more realistic estimate of the failure probability ε can be achieved.

[5]Actually it must be $Q/N \leq \varepsilon - \varepsilon'$ where ε' is the value assumed by the "asymptotic" negligible function $negl(k)$ when the protocol is instantiated with a concrete security parameter k. One can estimate ε' by carefully analyzing the proof of security of the protocol and using appropriate key-sizes (for example in the KOY protocol one would look at the time T of the best current algorithm for solving the Decisional Diffie-Hellman Assumption for the concrete parameters chosen by the protocol). In the rest of this chapter ignore the factor of ε', as it is not relevant to the discussion at hand and is anyway many order of magnitude smaller than ε.

4.5 Analysis of the Delay-Based Attacks and Recommendations

In this section, we formally analyze both timeout-delay and synchronization-delay attacks and show how to keep the failure probability bound by ε even in the presence of such attacks. In Section 4.5.3 we then show a generic transformation as countermeasures that can be taken to neutralize or minimize such attacks.

4.5.1 Timeout-based Attacks

In this section, we distinguish between instances representing on-line attacks and log-in failures and present their relationship.

On-line Attacks. Although the notion of on-line attacks is essential to the security definition of any PAKE protocol, previous work has defined it only in the context of the implicit authentication model [BPR00, KOY01]. Here we define the notion of on-line attacks for the explicit authentication model.

Intuitively, a protocol instance is an on-line attack if it satisfies two conditions, namely (1) the adversary actively sent a message to the instance, and (2) the adversary can tell whether a password guess in the instance is successful.

Log-in Failures. Intuitively, we say a protocol instance represents a *log-in failure* if the instance rejects the session key, regardless of the cause for rejection; i.e., incorrect authentication message received, timeout, number of login attempts exceeded. Note that an instance can be a log-in failure *and* an on-line attack at the same time. Conversely, we say a protocol instance represents a *log-in success* if the instance accepts the session key. Ultimately, all instances become either a log-in success or

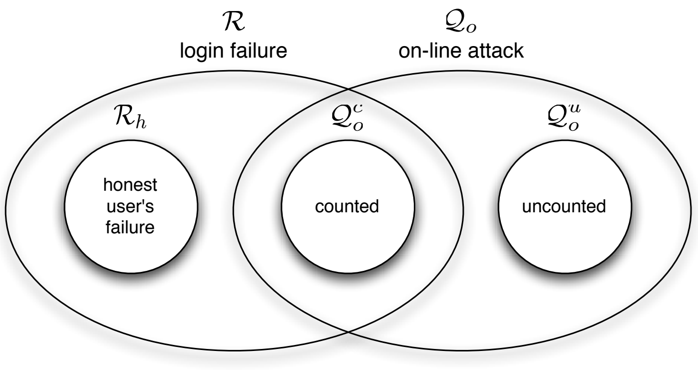

Figure 4.5: Sets of instances representing login failures and online attacks. ($r = |\mathcal{R}|$, $r_h = |\mathcal{R}_h|$, $q_o = |\mathcal{Q}_o|$, $q_o^c = |\mathcal{Q}_o^c|$, and $q_o^u = |\mathcal{Q}_o^u|$)

a log-in failure, due to the use of timeouts.

On-line Attacks vs. Log-in Failures. The difference between on-line attacks and login failures is illustrated in Figure 4.5 and explained below.

Let $\mathcal{R}$ denote set of protocol instances representing log-in failures, and $\mathcal{Q}_o$ denote set of instances representing on-line attacks. Then, $\mathcal{R}$ comprises log-in failures caused either by an honest user (i.e., via an incorrect password) or by an adversary (i.e., via a password-guessing attack). Clearly, instances of the latter must be included in the count of on-line attacks. We let $\mathcal{R}_h$ denote set of instances of the former (i.e., honest user's log-in failures) and let $\mathcal{Q}_o^c$ denote set of instances of the latter. In particular, instances of the latter represent both of log-in failures and on-line attacks. We call them *counted on-line attacks.*

Furthermore, in contrast with the counted on-line attacks, there may exist on-line attacks that do not represent log-in failures, which we call *uncounted on-line attack.* We also have:

$$q_o = q_o^c + q_o^u. \tag{4.2}$$

Timeout-based Attacks. In principle, the advantage of an adversary in breaking a PAKE protocol is proportional to the number of on-line attacks launched by that adversary. The security of a PAKE protocol is preserved only when the number of on-line attacks allowed is bounded. In practice, the only method to bound the number of such attacks is to count consecutive log-in failures for each client account. Hence, in our model, $q_o \leq r$.

However, as we have seen earlier (Equations 4.1, 4.2 and Figure 4.5) for r to be close to q_o, we need to have a small r_h and q_o^u. Fortunately, without loss of generality, we can assume that r_h is small since honest users are unlikely to make continuous login mistakes). However, as illustrated in the experiments shown in Section 4.3 , q_o^u can be very large, and this enables concurrent attacks. (Without loss of generality, we only need to consider a restricted concurrency model for user logins on the client side. A client bounds user-login attacks by counting login failures within the range of those counted by the server side.)

We consider now the cases when q_o^u can become large. Mutual authentication can be achieved in two different ways: Either the server sends the authentication message first, or the client sends the authentication message first. These two different forms of explicit authentication have different practical consequences; i.e., the former allows the existence of *uncounted on-line attacks* while the latter case does not. Clearly, *timeout-based attacks* are possible whenever $q_o^u > 0$; i.e., instances representing uncounted-online attacks exist. Therefore if client authentication precedes server authentication timeout-delay attacks can be prevented. As a practical example, we illustrate the introduction of client pre-authentication in extant PAKE protocols, such as those suggested for TLS integration, in the recommendations Section 4.5.3 below.

4.5.2 Representation of Synchronization-based Attacks

Client pre-authentication, however, does not prevent synchronization based, on-line attacks. These attacks are possible due to *inherent delays* in updating copies of failed-login counters in server replicas. To capture this fact, we introduce the following variables.

Let $r[C]$ be the global counter of login failures for all server replicas that share account information for a client C. Let $r[C]_i$ be a counter of login failures of a server replica S_i. Upon a new session requested by C, each server replica S_i updates $r[C]_i$ by setting it to $r[C]$. Additional sessions cannot be initiated by C whenever $r[C]_i$ reaches Q^*. The global counter r_C to be increased by 1 whenever a client session fails; i.e., a session key is not accepted. Clearly, we have:

$$r[C]_i \leq r_C. \tag{4.3}$$

Synchronization-based attacks are possible whenever $r_C - r[C]_i \geq 0$. This difference can be non-negligibly large when several server replicas are used since the synchronization delays increase.

In other words, to neutralize the effects of synchronization-delay based attacks one needs to estimate the number of protocol instances, δ, that an adversary can invoke during the maximum synchronization delay Δ of a system. If the PAKE protocol is secure according to Definition 4.1 then the probability of success of the adversary can only be bounded by $\frac{Q+\delta}{N} + negl(k)$. At this point the only way to reduce this probability to below the acceptable ε threshold of Definition 4.2 is to try to minimize δ (by minimizing Δ) or increasing the size of the required password space (e.g., via password complexity rules), to $N' \geq N$.

4.5.3 Recommendations for Existing PAKE Protocols

In this section, we provide a generic method of enhancing (or fixing) PAKE protocols so that they achieve *security against delay-based attacks*, in the client-server model. An exceptional case that a single session is only allowed at a time is out of concerns. First, we consider PAKE protocols that are provably secure in the implicit authentication model by Bellare, *et al.* [BPR00]. Then, we consider PAKE protocols that are provably secure in the explicit authentication model [BPR00, BR93]. We then describe a fixed PAKE protocol for SSL/TLS.

In fact, this method is a sort of simple, low-tech fix, but should work as a generic transformation for enhancing the security against delay-based attacks in the client-server model. Note that anyway protocols where the client authenticates first already exist, e.g., KOY [KOY01] and even in the IEEE standardization [IEE05].

4.5.3.1 Enhancement 1

First we consider PAKE protocols that are proven secure in the implicit authentication model. Our enhancement is essentially the transformation by Bellare, Rogaway and Pointcheval [BPR00] that adds mutual authentication to a PAKE pro-

Enhancement 1

Let H be the random hash function[a]

Client to Server Authentication: In π, if client C accepts $\mathsf{sk'}_C$ and terminates, do the following:

1. compute $\tilde{\mathsf{sk}}_C = H(C||S||\mathcal{T})$ where $\mathcal{T}$ denotes the protocol transcript resulted from the execution of π.
2. compute $\mathsf{sk}_C = H(\tilde{\mathsf{sk}}_C||0), \mathsf{auth}_C = H(\tilde{\mathsf{sk}}_C||1)$, and $\mathsf{auth}_S = H(\tilde{\mathsf{sk}}_C||2)$.
3. send an additional flow, auth_C and set $\mathsf{compute} = true$.

Server to Client Authentication: Upon receiving $\mathsf{auth'}_C$ from C, in π, if server S accepts $\mathsf{sk'}_S$ and terminates, do the same thing analogously as follows:

1. compute $\tilde{\mathsf{sk}}_S = H(C||S||\mathcal{T})$ where $\mathcal{T}$ denotes the protocol transcript resulted from the execution of π.
2. compute $\mathsf{sk}_S = H(\tilde{\mathsf{sk}}_S||0), \mathsf{auth}_C = H(\tilde{\mathsf{sk}}_S||1)$, and $\mathsf{auth}_S = H(\tilde{\mathsf{sk}}_S||2)$.
3. verify if $\mathsf{auth'}_C = \mathsf{auth}_C$, and if it is true, do the following: (a) send an additional flow, auth_S and, (b) set $\mathsf{compute} = true, \mathsf{accept} = true, \mathsf{determine} = true$ and terminate.
4. Otherwise, set $\mathsf{compute} = true, \mathsf{accept} = false, \mathsf{determine} = true$ and terminate.

Client verification: Upon receiving $\mathsf{auth'}_S$ from S, verify if $\mathsf{auth'}_S = \mathsf{auth}_S$, and do the following:

1. if the verification comes out true, set $\mathsf{accept} = true, \mathsf{determine} = true$ and terminate.
2. otherwise, set $\mathsf{accept} = false, \mathsf{determine} = true$ and terminate.

$\diamond$ Restricted concurrency is allowed to client-side protocol executions and full concurrency is allowed to server-side protocol executions.

[a]As described in [BPR00], a pseudo random function is sufficient for adding authentication, for simplicity, we follow them to assume the random oracle model for H.

Figure 4.6: Enhancing PAKE protocol π with implicit authentication into a protocol π'' secure against timeout-delay attacks.

Enhancement 2

Let π be a PAKE protocol satisfying (a) it is proven secure in the explicit authentication model; (b) in π, server-to-client authentication step proceeds client-to-server authentication step; and (c) both of client and server keep the password (symmetric setting)[a]

Initialization: In π, if client C is supposed to send the first message to initialize a communication with server S, send the first flow, $\mathsf{Hello}(C, S, \mathsf{init})$.

Switching roles: C runs a protocol execution following server-side of π and S runs a protocol execution following client-side of π.

$\diamond$ Restricted concurrency is allowed to client-side protocol executions and full concurrency is allowed to server-side protocol executions.

[a]This is the opposite to the asymmetric setting where server keeps image of password for a certain function.

Figure 4.7: Switching roles of client and server of protocol π to have a protocol π' secure against timeout-delay attacks.

tocol. However, we carefully fix the order of authentication so that client-to-server authentication occurs prior to server-to-client authentication.[6] The enhancement is illustrated in Figure 4.6 in detail.

4.5.3.2 Enhancement 2

Next, we consider PAKE protocols which are proven secure in the explicit authentication model. In these protocols, mutual authentication is composed of server-to-client authentication prior to client-to-server authentication. Such a composition makes PAKE protocol vulnerable to concurrent attacks when full concurrency is

[6]In [BPR00], a transformation for adding mutual authentication is described without carefully specifying whether the client or server authenticates first.

Client C		**Server** S
(password pw)		(password pw)
$\left.\begin{array}{r}\texttt{accept}, \texttt{compute}\\ \texttt{determine}\end{array}\right\} \leftarrow \mathsf{false}$		$\left.\begin{array}{r}\texttt{accept}, \texttt{compute}\\ \texttt{determine}\end{array}\right\} \leftarrow \mathsf{false}$
	Choose ciphersuite:	
choose $N_c \overset{R}{\longleftarrow} \{0,1\}^*$	$\xrightarrow{\texttt{ClientHello:}\ (N_c, C, ...)}$	
	$\xleftarrow{\texttt{ServerHello:}\ (N_s, ...)}$	choose $N_s \overset{R}{\longleftarrow} \{0,1\}^*$
	Compute Diffie-Hellman secret:	
	$\texttt{ServerKeyExchange:}(S, Y^\star)$ $\xleftarrow{\texttt{ServerHelloDone}}$	choose $y \overset{R}{\longleftarrow} \mathbb{Z}_q^*$, compute $Y \leftarrow g^y$, encrypt $Y^\star \leftarrow Y \times U^{pw}$
choose $x \overset{R}{\longleftarrow} \mathbb{Z}_q^*$, compute $X \leftarrow g^x$, decrypt $Y \leftarrow Y^\star / U^{pw}$	$\xrightarrow{\texttt{ClientKeyExchange:}\ (X)}$	
$\mathsf{compute} \leftarrow \mathsf{true}$	$Z = Y^x = X^y$	$\mathsf{compute} \leftarrow \mathsf{true}$

Compute pre-master secret and authentication key:

$\mathsf{PreMasterSecret} = \mathsf{Hash}(C,\ S,\ pw,\ X \| Y^\star \| Z)$

$\mathsf{AuthKey} = \mathsf{PRF}_1(\mathsf{PreMasterSecret},\ N_c \| N_s)$

Compute authenticators:

Auth_C=$\mathsf{MAC.Sign}_{AuthKey}$(“client finished”,...) Auth_S=$\mathsf{MAC.Sign}_{AuthKey}$(“server finished”,...)

	$\texttt{Authenticator}:\ \mathsf{Auth}_C$ $\xrightarrow{[\mathsf{ChangeCipherSpec}]}$	
		$\texttt{determine} \leftarrow \texttt{true}$ Abort if verification fails. Else: $\texttt{accept} \leftarrow \texttt{true}$
	$\texttt{Authenticator}: \mathsf{Auth}_S$ $\xleftarrow{[\mathsf{ChangeCipherSpec}]}$	
$\mathsf{determine} \leftarrow \mathsf{true}$ Abort if verification fails. Else: $\mathsf{accept} \leftarrow \mathsf{true}$		

Compute master secret and key material as in standard TLS:

$\mathsf{MasterSecret} = \mathsf{PRF}_2(\mathsf{PreMasterSecert},\ N_c || N_s)$

$\mathsf{KeyBlock} = \mathsf{PRF}_3(\mathsf{MasterSecret},\ N_s || N_c)$

$\longleftarrow\!\!\!-\!\!\!-\!\!\!-$ *Secure Channel* $-\!\!\!-\!\!\!-\!\!\!\longrightarrow$

Figure 4.8: The full handshake for PAKE-TLS ciphersuites in the full concurrency model.

allowed for server executions. Our enhancement is simply to swap the roles of the parties so that the client authenticated first; see Figure 4.7.

4.5.3.3 Fully Concurrent PAKE in TLS

Provably secure PAKE in TLS proposed recently by Abdalla *et al.* [ABC$^+$06] shows a practical application of provably secure PAKE. This work represents a sound attempt to replace a widely used AKE with provably secure PAKE. However, this protocol is also vulnerable to concurrent attacks since the authentication message of server precedes that of client.

We can apply our enhancement 2 to this protocol, and then obtain the new provably secure PAKE in TLS, as illustrated in Figure 4.8. (Let us borrow the depicting style of this Figure from [ABC$^+$06].) The original protocol of [ABC$^+$06] switches the roles of participants and lets the authentication message of client precede that of server. The recent standardization [BWNH$^+$03] of PAKE lets the `ClientHello` message include the ID of client in its extension field.

4.6 Conclusion

We have shown that existing formal definitions for PAKE do not provide a good model for the real-world security of a given implementation of a PAKE protocol. In particular, the definition – while theoretically sound – does not tightly address real-world concerns, and naive solutions to bridge this gap do not work. We hope our results and recommendations will influence the standardization of PAKE protocols going forward, and the better understanding of security engineers with regard to the security of PAKE protocols in the real world.

Taking a broader viewpoint, we can also ask: for what other cryptographic protocols do current formal definitions not adequately match practical needs?

4.7 Simple Experiments with Delay-Based Attacks

The delay-based attacks on PAKE protocols show that even provably secure protocols cannot bound on-line attacks as assumed in the previous adversary models. In other words, these models are fragile in the sense that a rigid bound cannot be obtained in practice. In this section, we illustrate practical attacks for the generic scenarios presented in Section 4.3.

Figure 4.9 shows the experimental result of concurrent attacks on two provably secure PAKE protocols, JG and PAK-Z+ [JG04, Mac02], with a wider spectrum of waiting time Δ. We perform the experiments in the Windows XP SP2 PC platforms having P4 2.6GHz CPUs and 2GB memory. The server implementation is multi-threaded as usual, while our adversary client is also multi-threaded for launching concurrent attacks. In the server threads of our experiments, every login failure (including wrong password and time-out) of the same account is counted and shared as a global variable. This is an emulation of multiple servers[7]. In a real-world attack, each client might be distributed in a *Trinoo* style. In these experiments,

[7]In reality, there would be a synchronization delay for such a count over multiple server hosts but this delay is an order of magnitude smaller than a timeout delay and would not affect the results of our experiments; this simple experiment illustrates our claim clearly.

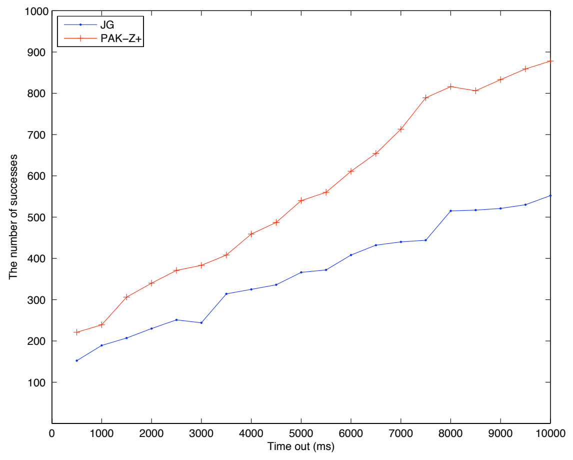

Figure 4.9: Expanded experimental results of concurrent attacks on provably secure PAKEs.

we emulate the timeout delay attack within a single host. Each client thread precomputes the challenge messages off-line with different password guesses, and sends them out concurrently on-line. Each of invoked server instances then responds with authentication messages for respective guesses unless the number of consecutive login failures exceeds the limit Q^*.

In Figure 4.9, we set the synchronized bound Q^* to 5 and distinct timeout delays Δ in a larger range than in Section 4.3. We implement the timeouts by asynchronous sockets; i.e., a non-blocking socket in Visual Studio 6.0. In each experiment, we have an adversary open 1000 concurrent sessions for the same account while we vary Δ. As we increase Δ from 500 ms to 10000 ms, a growing number of authentication messages that exceed bound Q^* significantly are obtained by the adversary. For example, when $\Delta = 500ms$, the JG protocol allows 152 guesses and PAK-Z+ protocol 221 guesses, on the average. However, when $\Delta = 5000ms$, JG allows 366 guesses and PAK-Z+ 540 guesses on the average, and when $\Delta = 10000ms$,

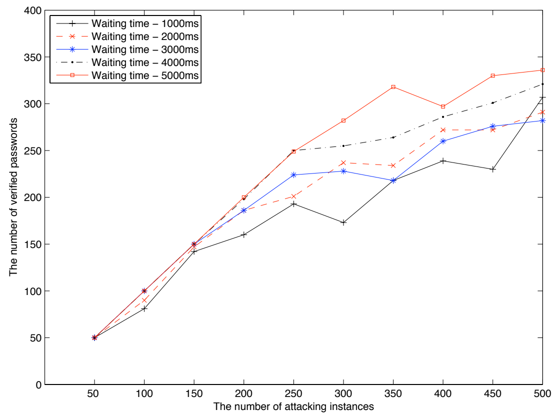

Figure 4.10: Experiment of JG protocol.

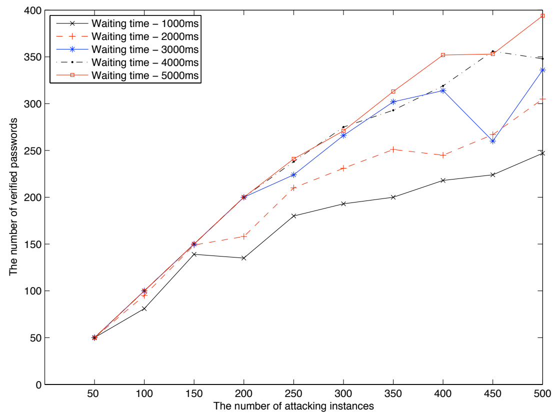

Figure 4.11: Experiment of PAK-Z+ protocol.

JG allows 552 guesses and PAK-Z+ 878 guesses on the average. Here the average is computed over 10 consecutive experiments.

Figure 4.9 shows that if the time-out duration is set loosely or never set at all, then almost all of concurrent requests cause servers to send out authentication messages. We observe that the difference between JG and PAK-Z+ results from the distinct computational loads placed on the server. (JG needs more computation than PAK-Z+ on a server when manipulating authentication messages, and this delays the handling of many requests until the failure count exceeds the security bound.) Interestingly, when we reduce the computation load on the server, by making the adversary send fewer requests concurrently, the results obtained are nearly linear between $\Delta = 1000$ and 5000. Figure 4.10 and Figure 4.11, respectively, illustrate the experimental results with fewer attack instances. Here, the number of verified passwords (i.e., successes of concurrent attacks) is almost linear in the number of attack instances when the number of attack instances is relatively small.

Chapter 5

Retaining Non-tightly Reduced Privacy Properties of Secure Encryption Schemes in the Real-World

5.1 Outline of the Chapter

In Section 5.2.2 we introduce security properties of encryption schemes and present important claims and theorem for security properties obtained by non-tight reduction proofs. In Section 5.3 we demonstrate specific attacks against secure encryption schemes. In particular, we describe details of our attacks, argue the feasibility of these attacks in practice, and show vulnerability examples. We also give examples of schemes withstanding our attacks and analyze their characteristics. In Section 5.4 we discuss related work. In Section 5.5 we provide full proofs of claims appeared in Section 5.2.

5.2 Security Properties of Encryption Schemes

Security-Property Definitions. In the asymptotic approach, an adversary is defined in terms of the possible attacks it can launch under a given computational model. An attack consists of a well-defined *goal*, such as the ability to distinguish encryptions of an adversary-supplied plaintext from those of a random string of the same length as that of the plaintext, and a set of *attack capabilities.* The attack capabilities enable an adversary to obtain ciphertexts of predictable, known or chosen plaintexts [HS93, Bih96, Bel97, SM00], and to use chosen ciphertexts [BDJR97, Rog04a]. The capabilities are typically denoted by PPA (predictable plaintext attacks), KPA (known plaintext attacks), CPA (chosen plaintext attacks) and CCA (chosen ciphertext attacks). Thus, a *security property* of an encryption scheme can be expressed as the pair <denial of an adversary goal - attack capability>.

For example, a strong adversary goal in attacking an encryption scheme expresses the ability to distinguish ciphertexts produced by encryption from random bits [AR00, Rog04b]. The *security property* expressing the ability of an encryption scheme to withstand an attack with such a goal using chosen plaintexts is denoted by "indistinguishability from random bits in a (adaptive) chosen-plaintext attack," or IND\$-CPA. Similarly, the ability of an encryption scheme to withstand an attack that attempts to distinguish between encryptions of real and random plaintexts, or between encryptions of two different plaintexts, of the same length, using chosen plaintexts is denoted by IND-CPA.

Formal definitions of an adversary's advantage in attacking different security properties of an encryption scheme are provided by the work of Bellare and Rogaway [BDJR97, Bel98, Rog04b, BR05]. The reader is referred to this body of work for in-depth coverage of these definitions. Examples of formal definitions for several security properties are given in Section 5.2.2 below. We present these definitions as background to the illustration of non-tight reduction results and their implications.

Security of an encryption scheme, $\Pi(n)$, is formally defined in the asymptotic approach by introducing the notion of an adversary A's advantage in attacking a security property of that scheme, *sec-prop*, and requiring that the advantage be bounded from above by a *negligible function*, $\varepsilon(n)$[1], where n is the security parameter; i.e.,

$$\mathsf{Adv}^{\text{sec-prop}}_{\Pi(n)}(A) \leq \varepsilon(n).$$

Reductions. Let A, B be security properties of an encryption scheme. Suppose that the advantage of *any* probabilistic, polynomially time (PPT) adversary B in attacking security property B, in q_B queries and time t_B, is known (i.e., explicitly assumed, proven) to be bounded by ε_B from above. Let A denote some PPT adversary program whose advantage in attacking security property A, in q_A queries and time t_A, exceeds ε_A. Then, the *reduction* B$\Rightarrow$A is a new PPT adversary program that gains an advantage greater than ε_B in attacking security property B by invoking an adversary A, possibly a polynomially-bounded number of times.

5.2.1 Non-tight Reductions

A reduction is said to be *tight* if $\varepsilon_A \approx \varepsilon_B$ and $t_A \approx t_B$ ($q_A \approx q_B$).[2] A reduction is *non-tight* if $\varepsilon_A \gg \varepsilon_B$ or $t_A \ll t_B$ ($q_A \ll q_B$). In typical tight reductions among properties of encryption schemes, $\varepsilon_A(n) \leq c \cdot \varepsilon_B(n) + \texttt{negligible(n)}$, where c is a small positive constant and $\texttt{negligible(n)}$ is a negligible function in the security parameter n. In contrast, results of non-tight reductions are illustrated by the following two examples:

[1]A function $\varepsilon(n)$ is negligible if for every polynomial $P(\cdot)$ there exists an N such that for all integers $n > N$ (i.e., for sufficiently large values of n) $\varepsilon(n) < \frac{1}{P(n)}$.

[2]More precisely, by $a \approx b$ we mean that $|a - b| \leq c \cdot min(a, b)$ for a small positive constant c.

1. $\varepsilon_A(n) \leq P(n) \cdot \varepsilon_B(n) + \texttt{negligible(n)}$, and

2. $\varepsilon_A(n) \leq c \cdot \varepsilon_B(n) + Q(n) \cdot 2^{-L(n)}$,

where c is a small positive constant, $P(n), Q(n)$ are positive polynomials, and $L(n)$ is the size of a block cipher. Of course, other inequalities for non-tightness follow, such as

3. $\varepsilon_A(n) \leq P(n) \cdot \varepsilon_B(n) + Q(n) \cdot 2^{-L(n)}$.

A source of non-tight reduction results of type 1 is the application of the hybrid techniques to proofs of indistinguishability; viz., Claim 5.1 where $P = p + 1$ and p is the number of oracles for chosen-plaintext encryption available to an adversary. Hybrid reduction proofs are typically used when a basic cryptographic primitive that exhibits property B is applied multiple times to obtain property A. For example, non-tight reductions appear in cases when property B is "there exists a PPT adversary that distinguishes a single sample from distribution X from a single sample of distribution Y with non-negligible success" and A is "there exists another PPT adversary that distinguishes a polynomial number, $P > 2$, of independent samples efficiently obtained from distribution X from P independent samples efficiently obtained from distribution Y with non-negligible success."

Multiple sources of non-tight reduction results of type 2 appear in typical proofs. For example, in Claim 5.2 below, $Q = p$ and p is the number of oracles for chosen-plaintext encryption available to an adversary. Also, in reductions from a block cipher modeled as a secure family of pseudorandom functions (PRF) to a security property (e.g., real-or-random, left-or-right indistinguishability) of a probabilistic encryption scheme, such as CBC$ or CTR$ [BDJR97], it is possible that $Q(n) \cdot 2^{-L(n)} = q^2(n) \cdot 2^{-L(n)} \gg c \cdot \varepsilon_B(n)$, where q is the number of queries to the block cipher used by that scheme. This would increase non-tightness almost proportionally to $Q(n)$. Furthermore, the non-tightness of this example is (slightly) increased in practice, when the block cipher is implemented with a family of secure pseudo-random permutations (PRPs), as in the case of AES and 3DES. In this case, switching from a PRF to a PRP family increases Q by a $\frac{q^2}{2^{l+1}}$ term, where l is the size of the block cipher.

An example of non-tight reduction of type 3 is provided by results of the Theorem 5.1 of Section 5.2.2. Here, $P = p + 1$ and $Q = p$ and p is the number of oracles for chosen-plaintext encryption available to an adversary.

Security Relevance. Non-tightness of a reduction proof is security-irrelevant in the asymptotic approach. The closure properties of negligible functions (viz., Fact 5.1 below) imply that a distinction between tight and non-tight reductions need not be made to guarantee a security property of an encryption scheme.

Fact 5.1. *(closure properties of negligible functions [KL08, Chapter 3])*

1. *If* $\texttt{negligible}_i(n), i = 1, 2,$ *are negligible functions in the parameter* n*, then* $\texttt{negligible}_3(n) = \texttt{negligible}_1(n) + \texttt{negligible}_2(n)$ *for a sufficiently large* n*.*

2. *If* $\texttt{negligible}_1(n)$ *is a negligible function in the parameter* n*, then* $\texttt{negligible}_2(n) = P \cdot \texttt{negligible}_1(n)$ *is negligible for a sufficiently large* n *whenever* P *is a pos-*

itive polynomial.

Regardless of how large positive polynomials $P(n)$ and $Q(n)$ may become as the result of a non-tight reduction of types 1 – 3 illustrated above, the negligibility of $\varepsilon_A(n)$ is assured by a sufficiently large security parameter n. Furthermore, *any* increase in P and Q, which increases the non-tightness of a reduction, can be offset by increasing the security parameter n, which causes an automatic decrease of $\varepsilon_B(n)$ and $2^{-L(n)}$. In short, in the asymptotic approach, the non-tightness of a reduction proof B $\Rightarrow$ A is irrelevant to the security property A obtained from that reduction. The only relevance of non-tightness in practice is on a scheme's performance, as a larger security parameter typically requires a larger key and, possibly, cipher-block size, which implies lower block-cipher performance [Bel98].

The key assumption that underlies the asymptotic approach, namely that an immediate increase of the security parameter offsets any increase of non-tightness factors, proves to impractical, as shown in Section 5.3.2 below. In the real world, a security parameter does not exist for symmetric-key block ciphers, and block-cipher parameters (i.e., the key and block sizes) cannot be increased for decades, not years. Thus, the results of non-tight reduction proofs become security-relevant in practice.

5.2.2 Security Properties Obtained by Tight and Non-Tight Reductions

In this section, we review literature definitions of properties obtained by tight and non-tight reductions, which we use in our illustrations. The proofs of the claims and theorem of this section are found in the Appendix.

Definition 5.1. (IND\$-CPA) [AR00, Rog04a] Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let n be a security parameter, and let A be a chosen plaintext attack adversary. Define

$$\mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A) \stackrel{\text{def}}{=} \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : A^{\mathcal{E}_k(\cdot)}(n) = 1\right] - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : A^{\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right]$$

Encryption scheme $\Pi(n)$ is IND\$-CPA secure if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A) \leq \varepsilon(n)$, where $\varepsilon(n)$ is a negligible function of n.

Definition 5.2. (KH-CPA) [Fis99] Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let n be a security parameter, and let A be a chosen plaintext attack adversary. Define

$$\begin{aligned}\mathsf{Adv}^{\text{KH-CPA}}_{\Pi(n)}(A) \stackrel{\text{def}}{=} & \Pr\left[k, k' \xleftarrow{r} \mathcal{K}(n) : A^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)}(n) = 1\right] \\ & - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : A^{\mathcal{E}_k(\cdot), \mathcal{E}_k(\cdot)}(n) = 1\right]\end{aligned}$$

Encryption scheme $\Pi(n)$ is *key-hiding* in a chosen-plaintext attack (or KH-CPA) if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}^{\text{KH-CPA}}_{\Pi(n)}(A)$ is negligible (as a function of n).

The "key hiding" property, KH, conveys a sense of key privacy as ciphertexts produced by encryption with the same key cannot be distinguished from those produced by encryption with different keys.

The "multi-key hiding" property, KH_p, defined below, allows the adversary to access $p > 2$ oracles. This property was parenthetically suggested by Abadi and

Rogaway [AR00] who observed that it is indistinguishable from the ordinary "key hiding" property, KH, from the point of view of adversary power; i.e., an adversary gains no extra power by accessing $p > 2$ oracles instead of just two. It is interesting to note that a similar observation regarding the power of an adversary was made for the formal approach to adversary definition, as illustrated by the Dolev-Yao model [DY83]. Micciancio [Mic05] shows that the Dolev-Yao adversary needs no more than two parties to break a cryptographic protocol, and hence the availability of a larger number of protocol parties adds no power to that adversary.

The attacks presented in the next section show that these observations, while correct in the asymptotic and formal-methods approach, do not hold in practice.

Definition 5.3. (KH$_p$-CPA) Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let n be a security parameter, and let A be a chosen plaintext attack adversary. Define

$$\begin{aligned}\mathsf{Adv}^{\mathrm{KH}_p\text{-CPA}}_{\Pi(n)}(A) &\stackrel{\mathrm{def}}{=} \Pr\left[k_1, k_2, ..., k_p \stackrel{r}{\leftarrow} \mathcal{K}(n) : A^{\mathcal{E}_{k_1}(\cdot), \mathcal{E}_{k_2}(\cdot), ..., \mathcal{E}_{k_p}(\cdot)}(n) = 1\right] \\ &\quad - \Pr\left[k \stackrel{r}{\leftarrow} \mathcal{K}(n) : A^{\mathcal{E}_k(\cdot), ..., \mathcal{E}_k(\cdot)}(n) = 1\right]\end{aligned}$$

Encryption scheme $\Pi(n)$ is "multi-key hiding" in a chosen-plaintext attack with p oracles (or KH$_p$-CPA) if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}^{\mathrm{KH}_p\text{-CPA}}_{\Pi(n)}(A)$ is negligible (as a function of n).

Definition 5.4. (KR-CPA) [BR05, Chapter 5.12] Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let n be a security parameter, and let A be a chosen plaintext attack adversary. Define

$$\mathsf{Adv}^{\mathrm{KR\text{-}CPA}}_{\Pi(n)}(A) \stackrel{\mathrm{def}}{=} \Pr\left[k \stackrel{r}{\leftarrow} \mathcal{K}(n), A^{\mathcal{E}_k(\cdot)}(n) = k' : k' = k\right]$$

Encryption scheme $\Pi(n)$ is secure against key-recovery in a chosen-plaintext attack (or has the KR-CPA property) if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}_{\Pi(n)}^{\text{KR-CPA}}(A)$ is negligible (as a function of n).

The following property is informally described in [KM06] and formally defined here:

Definition 5.5. (EKR$_p$-CPA) Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let n be a security parameter, and let A be a chosen plaintext attack adversary. Define

$$\mathsf{Adv}_{\Pi(n)}^{\text{EKR}_p\text{-CPA}}(A) \overset{\text{def}}{=} \Pr\left[k_1, ..., k_p \overset{r}{\leftarrow} \mathcal{K}(n), A^{\mathcal{E}_{k_1}(\cdot), ..., \mathcal{E}_{k_p}(\cdot)}(n) = k' : k' \in \{k_1, ..., k_p\}\right]$$

Encryption scheme $\Pi(n)$ is secure against "existential-key-recovery" in a chosen-plaintext attack with p oracles (or has EKR$_p$-CPA property) if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}_{\Pi(n)}^{\text{EKR}_p\text{-CPA}}(A)$ is negligible (as a function of n).

Note that the EKR$_p$ property is exactly the KR property when $p = 1$.

Fact 5.2. [IND\$-CPA $\Rightarrow$ KR-CPA] *[Rog04a] [BR05, Chapter 5.12] Let* $\Pi(n)$ *be an* IND\$-CPA *secure encryption scheme. That is, for every PPT adversary A attacking the* IND\$-CPA *property of* $\Pi(n)$*, there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\mathsf{Adv}_{\Pi(n)}^{\text{IND\$-CPA}}(A) \leq \varepsilon(n)$$

Then, for every PPT adversary B attacking the KR *property of* $\Pi(n)$*, we have:*

$$\mathsf{Adv}_{\Pi(n)}^{\text{KR-CPA}}(B) \leq \varepsilon(n) + \frac{1}{2^{l(n)}}$$

where $l(n)$ is the block size used in the encryption scheme $\Pi(n)$.

Claim 5.1. [IND\$-CPA $\Rightarrow$ KH$_p$-CPA] *Let* $\Pi(n)$ *be an* IND\$-CPA *secure encryption scheme. That is, for every PPT adversary A attacking the* IND\$-CPA *property of* $\Pi(n)$*, there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A) \leq \varepsilon(n).$$

Then, for every PPT adversary B attacking KH$_p$-CPA *property of* $\Pi(n)$*, we have:*

$$\mathsf{Adv}^{\text{KH}_p\text{-CPA}}_{\Pi(n)}(B) \;\; \leq \;\; (p+1)\cdot\varepsilon(n).$$

Claim 5.2. [KH$_p$-CPA $\Rightarrow$ EKR$_p$-CPA] *Let* $\Pi(n)$ *be an encryption scheme. Then, for any PPT adversary A attacking the* EKR$_p$-CPA *property of* $\Pi(n)$*, there exists a PPT adversary B attacking the* KH$_p$-CPA *property of* $\Pi(n)$ *such that:*

$$\left|\mathsf{Adv}^{\text{EKR}_p\text{-CPA}}_{\Pi(n)}(A) - \mathsf{Adv}^{\text{KH}_p\text{-CPA}}_{\Pi(n)}(B)\right| \;\; \leq \;\; \frac{p}{2^{l(n)}}$$

where $l(n)$ is the block size of encryption scheme $\Pi(n)$*.*

An obvious corollary of this claim can be summarized as follows:

KH$_p$-CPA security $\Rightarrow$ KH-CPA security $\Rightarrow$ KR-CPA security.

Also, it is easy to show that KR-CPA $\not\Rightarrow$ KH-CPA security.[3]

By Claims 5.1 and 5.2, we easily obtain the following theorem.

Theorem 5.1. [IND\$-CPA $\Rightarrow$ EKR$_p$-CPA] *Let* $\Pi(n)$ *be an* IND\$-CPA *secure encryption scheme. That is, for every PPT adversary A attacking the* IND\$-CPA

[3]In constructing a counterexample, use an IND\$-CPA secure scheme that appends the least significant bit of the encryption key to each ciphertext. This scheme preserves the KR-CPA security but not KH-CPA.

property of $\Pi(n)$, *there exists a negligible function* $\varepsilon(\cdot)$ *such that*

$$\mathsf{Adv}_{\Pi(n)}^{\text{IND\$-CPA}}(A) \leq \varepsilon(n)$$

Then, for every PPT adversary B *attacking the* EKR_p-CPA *property of* $\Pi(n)$, *we have:*

$$\mathsf{Adv}_{\Pi(n)}^{\text{EKR}_p\text{-CPA}}(B) \quad \leq \quad (p+1)\cdot\varepsilon(n) + \frac{p}{2^{l(n)}}$$

5.3 Attacks against Secure Encryption Schemes

In this section we describe specific attacks against the "existential key recovery" (EKR_p) property of IND\$-CPA secure schemes. Essentially, our attacks are known/predictable plaintext attacks and so the success of these attacks against the EKR_p property automatically implies breaking the EKR_p property *under chosen plaintext attacks* (i.e., EKR_p-CPA). Detailed attacks are presented later in Section 5.3.3.

We stress that our attacks break the EKR_p property and KR_p property while they do not break the IND\$ and KH and KR properties. In effect the attack shows that:

1. The properties (EKR_p and KH_p) obtained by non-tight reduction are *not* retained in the real-world.

2. The adversary *does* obtain more power by accessing $p > 2$ oracles instead of just two (as mentioned in Section 5.2.2)

3. Universal property is *not* equivalent to existential property; namely, "existential key recovery" (EKR$_p$) and "key recovery" (KR) properties are not equivalent.

In practice, this means that, while all US national standard encryption schemes (modes) [Dwo01] are IND-CPA secure, and hence secure against "key recovery" attacks, some fail to exhibit not just "existential key recovery security" but also "multi-key hiding," whenever these modes are implemented with standard block ciphers. In Section 5.3.3, we illustrate practical attacks against two schemes that are proved IND\$-CPA secure. An immediate consequence of our attacks is that either the vulnerable IND\$-CPA schemes must be replaced with schemes that withstand such attacks, or the key lengths of standard block ciphers [BBB$^+$07] must be increased.

5.3.1 Overview of Attack Strategies

EKR$_p$ *Attack.* In contrast with the asymptotic approach where non-tight reduction results have no vulnerability significance, in the real-world, an adversary can exploit such results to attack properties of encryption schemes proven secure in the asymptotic approach.

The network adversary's strategy is to increase the non-tightness factor $P = p + 1$ (viz., Theorem 5.1) to a level necessary for non-negligible advantage. At that level, these security properties are vulnerable and lose all their value. To increase P to that level, the network adversary needs to find an abundance of predictable/known plaintext-ciphertext pairs. How to do that has been known and demonstrated in practice for more than six decades (viz., discussion below). Briefly, the adversary precomputes a table containing the ciphertexts and keys used to encrypt a known constant plaintext. The adversary harvests known/predictable plaintext-ciphertext pairs from P continuously-available Internet oracles and derives ciphertexts that encrypt the known constant. Finally the adversary searches the precomputed table for a collision between the derived ciphertexts and the table entries. A collision reveals the key that encrypted the harvested ciphertext. In Section 5.3.3, we give the details of such attacks, compute the adversary's advantage, and illustrate the vulnerability of some secure encryption schemes implemented with standard block ciphers.

5.3.2 What Enables Network Attacks in Practice?

As shown in Section 5.2.1 above, the asymptotic approach assumes that the security parameter of can be increased at any time to offset an attack that increases the non-tightness factors P and Q. In this section we show that in practice (1) neither a security parameter nor block-cipher parameters can be increased during the lifetime of an attack; and (2) the non-tightness factors P and Q can be increased at zero, or near-zero, marginal cost by the adversary. This enables successful attacks with non-negligible advantage (viz., Section 5.3.3).

Lack of a Security Parameter. In the real world, a security parameter cannot be used to offset increases in the non-tightness factors P and Q for obvious reasons: neither block ciphers nor encryption schemes have a security parameter [KL08, Rog04a]. Furthermore, block-cipher parameters, namely the key and block sizes, cannot be increased to compensate for a possible weakness of an encryption scheme in an attack, since these sizes are kept constant for decades, not years. For example, based on the NIST Key Management guidelines regarding the longevity of different key sizes[BBB+07], the 112-bit key size of 3DES will have lasted for *over three decades* and its 64-bit block size for *over five decades* by year 2030, when this block cipher will be phased out from use. Similarly, the 128-bit AES block and key sizes would have lasted for *at least three decades* by year 2030 and beyond. Block-cipher parameters are often hard-coded in both object code and hardware, and thus their Internet-wide replacement becomes an extremely costly exercise. For all practical purposes, the key and block sizes of a cipher are constant during the lifetime of any network-adversary attack.

Continuous Availability of Multiple Oracles. Consider a network of n_h nodes (hosts) each having r compatible communication links, where typically $r \geq n_h$ (i.e., two hosts may have multiple links between them). Each compatible link represents the use of the same protocol and secure encryption scheme, and each link uses an independently generated encryption key. In such a network, the adversary can use each of the $P = r \cdot n_h$ links as a freely available oracle for predictable/known plaintext-ciphertext pairs, whenever the communication protocol encrypts a predictable/known plaintext.

A vast and continuous supply of oracles that encrypt predictable/known plaintext is available in the Internet. For example, the headers of the TCP/IP protocol suite, which are universally deployed in the Internet, offer no fewer than 88 known bits [Bel97, MF01] that are encrypted by the IPsec suite using a different key per communication link. Application-level protocols that encrypt files (e.g., postscript file) produce globally-known *constant* headers encrypted in different keys [Bih96, Bih02], as do HTTP and e-mail applications. Similarly, message padding [Dwo01] offers predictable plaintext and sometimes known constant plaintext (e.g., for same-size messages) encrypted in different keys. In fact, oracles that produce encryptions of known (constant) plaintext in different keys have been exploited for more than six decades; i.e., since World War II. For example, messages encrypted with the Naval Enigma in different keys (e.g., by different U-boats, in different regions of the Atlantic ocean and Baltic sea), contained globally known constants like the sender's rank "Offizer" and weather report "Wettervorhersage Deutsche Bucht." [Bel97, HS93].

Oracles produced by standard protocols (e.g., known header/trailer fields) cannot be eliminated in practice, and are continuously available to an adversary at near-zero marginal cost (discussed below). Equally importantly, the number of oracles available in the Internet is not just large, but it grows *quadratically* in the number of the network nodes over time; i.e., $P \geq n_h^2$. This is very significant, as the number of network nodes, n_h, is expanding at a reasonably fast rate. This can be inferred from the real-life Internet experience: in 2008 the number of Internet users (and presumably nodes) grew by 89%. Between 1997 and 2008, Internet traffic doubled every 6 months. The Internet is expected to grow from nearly 1 billion hosts today to 2 billion hosts in 2011 [LB06]. Rapid growth in the number of Internet nodes can be anticipated for years to come given the anticipated growth of Internet-enabled phones.

Zero Marginal Cost. The network adversary exploits the availability of P oracles appropriate for the attack at zero, or near-zero, marginal cost. This means that once the adversary "pays" the initial cost of network access it only needs to passively collect[4] ciphertext for known/predictable plaintext from various freely available P oracles. However, it need not expend any resources to generate the ciphertexts; i.e., the adversary can simply find and exploit a large number of oracles in current protocols [SM00]. Furthermore, the adversary does not need to harvest, transfer and store all of the ciphertexts at once. Instead, the continuous availability of P oracles allows an adversary to harvest ciphertext, transfer it at Internet speed and buffer it incrementally at essentially zero cost [GH03, Gil08] during the lifetime of block-cipher parameters [BBB+07]. Hence, communication costs, whose unit costs are typically higher than those of storage, will be insignificant.

5.3.3 Vulnerability Examples: Nonce-based IND\$-CPA Schemes

Nonce-based IND\$-CPA *Encryption.* Rogaway introduced the notion of nonce-based schemes, illustrated their practical benefits, and proved their IND\$-CPA security. Briefly, a nonce is "a value, like a counter that is used only once within a ses-

[4]Recent studies indicate that a tier-1 Autonomous System (AS) can hijack 60-70% of the traffic destined to the top 100 websites [BFZ07], and companies such as Akamai claim to deliver 10-20% of all web traffic.

sion" [Rog04b]. Nonces can be used as initialization vectors, not just counters, and have a fundamental property: as long as they do not repeat on any message encryption, they could be chosen by an adversary. Hence, they need not be secret and need not be part of the definition of the encryption or decryption algorithms of secure schemes.

In this section we describe two attacks that succeed with non-negligible and uncomfortably-high probability against the EKR$_p$-KPA (and KH$_p$-KPA) security of two nonce-based IND\$-CPA schemes implemented with standard block ciphers (e.g., AES-128 and 2-key 3DES); i.e., schemes CTR1 and CTR2 of Figures 5.1 and 5.2. While the security of these schemes is broken with high probability, other nonce-based IND\$-CPA schemes such as CBC2 (viz., Figure 5.3) fare better, as their security properties are left intact. We note that neither the asymptotic nor the concrete security approach distinguish between the vulnerable and non-vulnerable schemes illustrated in Figures 5.1–5.3. All are provably secure in both approaches [BDJR97, Rog04b].

5.3.3.1 Key-Collision Attacks against EKR$_p$-KPA/-PPA Security

The attacks presented below fall into the broad category of "key-collision" attacks that use known and predictable plaintexts obtained with standard communication protocols. The differences between these attacks and those that have been successfully launched since World Was II [HS93], most recently by Biham and others [Bih96, Bih02, Gli98, MF01], are discussed in Section 5.4.

1. The EKR$_p$-KPA **Attack.** In the CTR1 scheme illustrated in Figure 5.1 below, the input to the encryption algorithm $E_K(\cdot)$, namely $(S+i)$ is known to an adversary. Assume that, in a communication protocol using this scheme, some plaintext block x_i in fixed message position i is known for P communication links and their keys. This is a common case, since communication protocols use highly formatted messages. For example, this block could be part of a message header containing the protocol and other identifiers; e.g., identifiers for the encryption, message authentication code, and compression algorithms used. Of course, knowledge of the plaintext block x_i can be acquired from the protocol definition and operational use.

When used with communication protocols such as the ones mentioned above, scheme CTR1 produces a large number of known plaintext-ciphertext pairs $< x_i, E_{K_j}(S+i) \oplus x_i >$ corresponding to the known value and fixed position of block $x_i, i = 0, \cdots m-1$, for all keys $K_j, j = 1, ..., P$. Now the adversary uses these pairs to construct *constant* plaintext-ciphertext pairs. Recall that in nonce-based encryption schemes the nonce, N, and hence S in CTR1, is initialized to the same known constant for all keys. This implies that the adversary can easily construct plaintext-ciphertext pairs $< (S+i), E_{K_j}(S+i) >$ where $(S+i)$ is a *constant* for all P keys, K_j.

In anticipation of harvesting ciphertexts for known plaintexts and constructing the constant pairs, the adversary enciphers the constant $(S+i)$ in T *distinct* keys $K_1, \cdots, K_T$ and precomputes a table comprising entries $< E_{K_1}(S+i), K_1 >, \cdots, < E_{K_T}(S+i), K_T >$ indexed by these ciphertexts.

The adversary searches this table with the harvested ciphertext $E_{K_j}(S+i)$ of unsuspecting network nodes (i.e., $r \cdot n_h$ oracles) to find key collisions. A collision would break EKR$_p$-KPA security since the adversary would discover one of the P keys used in the network.

Similar attacks also work against the CTR2 scheme (viz., Figure 5.2). For CTR2, assume that the adversary's known plaintext-ciphertext pair is block x_i at index i. Hence, the adversary can easily construct the known constant plaintext-ciphertext pairs $< N+i, E_{K_j}(E_{K_j}(N)+i) >$ where nonce N is initialized to a known *constant* for all P keys, K_j. Note that the ciphertext entries in the precomputed table require double encryption (with the same key, K_j) in the P distinct keys. This means that he adversary need only double its effort expended in breaking CTR1 to break CTR2.

2. The EKR$_p$-PPA **Attack.** Schemes CTR1 and CTR2 allow successful key-collision attacks even if the adversary can only *predict*, but not know with certainty, the value of x_i prior to actual attack. That is, the adversary would know that plaintext block x_i has one of the possible x_{i_j} values, $j = 1, \cdots, s$ for some (small) value of s. Informally, we say that such a plaintext x_i is *predictable* if the value of s is small compared with that of the block-cipher parameters.

In attacking CTR1, the adversary harvests the ciphertexts $E_{K_j}(S+i) \oplus x_i$ for keys K_j. For each ciphertext produced by some unknown key K, the adversary constructs s candidate pairs $< (S+i), E_K(S+i) \oplus x_i \oplus x_{i_j} >$, where one such pair is guaranteed to be the constant plaintext-ciphertext pair $< (S+i), E_K(S+i) >$. This is the case since, by definition, $x_i = x_{i_j}$ for one of the s values of j. Then the adversary searches the precomputed table (described above) using the s candidate

```
**Algorithm** CTR1.Encrypt$^N_K(M)$
$S \leftarrow N||0^{l/2}$
$m \leftarrow \lceil |M|/l \rceil$
$P \leftarrow E_K(S+0)||\cdots||E_K(S+m-1)$
$C \leftarrow M \oplus P$[first $|M|$ bits]
**return** $C$
```

```
**Algorithm** CTR1.Decrypt$^N_K(C)$
$S \leftarrow N||0^{l/2}$
$m \leftarrow \lceil |M|/l \rceil$
$P \leftarrow E_K(S)||\cdots||E_K(S+m-1)$
$M \leftarrow C \oplus P$[first $|C|$ bits]
**return** $M$
```

Figure 5.1: Scheme CTR1. The nonce space is Nonce=$\{0,1\}^{l/2}$ [Rog04a].

```
**Algorithm** CTR2.Encrypt$^N_K(M)$
$S \leftarrow E_K(N)$
$m \leftarrow \lceil |M|/l \rceil$
$P \leftarrow E_K(S+0)||\cdots||E_K(S+m-1)$
$C \leftarrow M \oplus P$[first $|M|$ bits]
**return** $C$
```

```
**Algorithm** CTR2.Decrypt$^N_K(C)$
$S \leftarrow E_K(N)$
$m \leftarrow \lceil |M|/l \rceil$
$P \leftarrow E_K(S)||\cdots||E_K(S+m-1)$
$M \leftarrow C \oplus P$[first $|C|$ bits]
**return** $M$
```

Figure 5.2: Scheme CTR2. The nonce space is Nonce=$\{0,1\}^l$ [Rog04a]. Unlike CTR1, CTR2 encrypts the nonce before using it as a counter. This scheme is similar to the XOR Scheme with Random Counters [Gli98], which outputs (S, C) at encryption.

ciphertexts. If one of these s ciphertexts collides with a ciphertext entry $E_{K_j}(S+i)$ in the precomputed table, the adversary will obtain the key collision between unknown key K and known key K_j in the table. The adversary repeats this process for all P unknown keys that encrypt predictable plaintext x_i. In attacking CTR2, the adversary uses predictable plaintexts and follows the attack for CTR1 after double encryption (with the same key, K_j) in the P distinct keys to construct the precomputed table.

Adversary Advantage in attacking EKR$_p$-KPA *security.* Let k be the length of the block-cipher keys used by the IND\$-CPA scheme Π = CTR1. Assume that each of the n_h nodes has r communication links. Let T be the number of entries in the table that the adversary precomputes and $P = r \cdot n_h \geq T$ be the number of oracles available to the adversary (and the non-tightness factor). In this case,

$$\mathsf{Adv}_{\Pi}^{\mathrm{EKR}_p\text{-KPA}}(A) = \frac{r \cdot n_h \cdot T}{2^k} = \frac{P \cdot T}{2^k} \geq \frac{T^2}{2^k}.$$

Thus the adversary can increase its advantage[5] *quadratically*, while increasing its resource expenditures only *linearly*, in the size of the precomputed table T. The adversary need only "pay" for storing T ciphertexts and their distinct keys in the precomputed table.[6]

Note that if $r \geq n_h$, then this advantage becomes:

$$\mathsf{Adv}_{\Pi}^{\mathrm{EKR}_p\text{-KPA}}(A) \geq \frac{n_h^2 \cdot T}{2^k}.$$

This shows that an adversary's advantage increases *quadratically*[7] over time in the number of Internet hosts, n_h, that use a vulnerable scheme with a protocol that produces predictable/known plaintext-ciphertext pairs, without the adversary expending any extra effort and resources.

The cost of harvesting P target ciphertexts from Internet sessions is relatively small, compared with the table precomputation and storage, and is adjustable as ciphertext harvesting can be done *incrementally and continuously*; i.e., ciphertexts for known or predictable plaintexts in the same message position encrypted in different keys are produced continuously by Internet protocols [Bel97]. This has two positive consequences for assuring near zero marginal cost to the adversary:

[5]A similar advantage can be derived for the attack against EKR_p-PPA security.

[6]If Π = CTR2, the advantage of the adversary is half the above value, as the adversary can build a table of size $T/2$ with the same resources as those for the CTR1 attack.

[7]This adversary earns its name as it takes advantage of the "network effect" suggested by Metcalfe's Law [Gil93].

1. Unlike a KR attack where the adversary focusses exclusively on a *single oracle at a time*, in an EKR$_p$ or KH$_p$ attack the adversary can use *any* stream of P oracles at *any time*. Hence, the adversary need not harvest and buffer large amounts of ciphertexts at any time, and can adjust its communication and buffering needs to ensure zero-marginal costs. Thus communication and buffering costs are not dominant in this attack.

2. Over time, the adversary's advantage (i.e., probability of a key collision) grows *quadratically* in n_h (since $r \geq n_h$) for the same initial investment of precomputing a table of T entries. Hence, the adversary's advantage grows naturally in time, without appreciable increase in the dominant cost (e.g., storage).

According to [GPS09], in practice, the parameters defining the adversary's advantage above are currently in the following Internet-scale ranges: $2^{24} \leq n_h \leq 2^{26}$, which represents only about 2 - 7% of the Internet hosts now, and $2^{26} \leq r \leq 2^{38}$ represents a range of per-year connections to various Internet sites, from moderately "popular" to highly visited ones (e.g., Craig's list, Google). Given the current speed of AES encryption with commercially available devices, the size of the table the adversary precomputes, T, can reach 2^{54} AES cipherblocks in one year of computing, using either one Helion Giga card [Hel08] on a PC or 64 quadcore, 2.6 GHz Apple Macs. Thus, for the implementation of $\Pi = \{\text{CTR1, CTR2}\}$ with the block cipher AES-128, we can use a variety of network and attack capabilities representing different points on the adversary's advantage curve. As an example, for

```
Algorithm CBC2.Encrypt$^N_{K_1,K_2}(M)$
if $|M| \notin \{l, 2l, 3l, ...\}$ then return $*$
Parse $M$ into $M_1 \cdots M_m$ where $|M_i| = l$
$C_0 \leftarrow E_{K_1}(N)$
for $i \leftarrow 1$ to $m$ do
    $C_i \leftarrow E_{K_2}(C_{i-1} \oplus M_i)$
return $C_1 \cdots C_m$
```

```
Algorithm CBC2.Decrypt$^N_{K_1,K_2}(C)$
if $|C| \notin \{l, 2l, 3l, ...\}$ then return $*$
Parse $C$ into $C_1 \cdots C_m$ where $|C_i| = l$
$C_0 \leftarrow E_{K_1}(N)$
for $i \leftarrow 1$ to $m$ do
    $M_i \leftarrow C_{i-1} \oplus E^{-1}_{K_2}(C_i)$
return $M_1 \cdots M_m$
```

Figure 5.3: Scheme CBC2. The nonce space is Nonce=$\{0,1\}^l$ [Rog04a].

$(r = 2^{28}, n_h = 2^{24}, T = 2^{48})$ and $(r = 2^{31}, n_h = 2^{27}, T = 2^{50})$ we obtain:

$$2^{-28} \leq \mathsf{Adv}^{\text{EKR}_p\text{-KPA}}_{\Pi-AES}(A) \leq 2^{-20}$$

for CTR1 and half that for CTR2. A similar attack against the implementation of Π with 112-bit 3DES (2TDES) would yield:

$$2^{-12} \leq \mathsf{Adv}^{\text{EKR}_p\text{-KPA}}_{\Pi-2TDES}(A) \leq 2^{-4}.$$

Clearly, the adversary advantage in these attacks is decidedly non-negligible.

Attack Cost Points. The adversary can adjust the attack cost points as a function of the dominant cost beyond taking advantage of yearly drops in memory costs (i.e., 37 - 50% per year). The dominant cost of this attack is the storage for the 2^{48} - 2^{50} AES block ciphers and keys. For AES, we estimate the wholesale cost to range between \$0.5 M - \$2 M in early 2009 (at \$65 per TB) and \$62.5 K - \$250 K in 2012 (at \$8 per TB).[8] In early 2009, the energy costs of running and cooling the storage would likely add \$4-\$9/TB[9]. The cost of the two-key 3DES attack is about half that of the AES. Note that the adversary can perform its attack by building

[8]Our cost figures are fairly conservative. More optimistic cost projections for the period 2008 - 2023 are \$38.56 per TB in 2009 and \$6.42 in 2012 [Gil08].

[9]At 10 W per active disk [RSRK07], each disk will consume 87.7 kWh in a year.

the precomputed table incrementally and still take advantage of the continuous availability of predictable/known plaintext. For example, the adversary can start the attack well before T reaches 2^{48} key-ciphertext entries and continue to build the table as storage costs decrease.

5.3.3.2 Breaking the EKR$_p$-CPA property implies breaking the KH$_p$-CPA property

By Claim 5.2, given an adversary A attacking the EKR$_p$-CPA with a non-negligible advantage, we can obtain an adversary B attacking the KH$_p$-CPA with a non-negligible advantage such as:

$$\mathsf{Adv}_{\Pi(n)}^{\text{EKR}_p\text{-CPA}}(A) - \frac{p}{2^{l(n)}} \quad \leq \quad \mathsf{Adv}_{\Pi(n)}^{\text{KH}_p\text{-CPA}}(B) \quad \leq \quad \mathsf{Adv}_{\Pi(n)}^{\text{EKR}_p\text{-CPA}}(A) + \frac{p}{2^{l(n)}}$$

Thus, if we consider the earlier example (in Section 5.3.3.1) of the implementation of $\Pi = \{\text{CTR1,CTR2}\}$ with the block cipher AES-128, for $(r = 2^{28}, n_h = 2^{24}, T = 2^{48})$ and $(r = 2^{31}, n_h = 2^{27}, T = 2^{50})$ we obtain:

$$2^{-28} - 2^{-80} \leq \ \mathsf{Adv}_{\Pi-AES}^{\text{KH}_p\text{-KPA}}(A) \ \leq 2^{-20} + 2^{-78}, \text{ or}$$

$$2^{-29} < \ \mathsf{Adv}_{\Pi-AES}^{\text{KH}_p\text{-KPA}}(A) \ < 2^{-19}$$

for CTR1 and half the value for CTR2. A similar attack against the implementation of Π with 112-bit 3DES (2TDES) would yield:

$$2^{-13} < \ \mathsf{Adv}_{\Pi-2TDES}^{\text{KH}_p\text{-KPA}}(A) \ < 2^{-3}.$$

Clearly, the adversary B's advantage is also decidedly non-negligible. However, such an implication does not apply for the case of the IND\$-CPA property. As shown in Theorem 5.1, the advantage of attacking EKR$_p$-CPA is bounded by the multiplication of the non-tightness factor p and the advantage of attacking IND\$-CPA. The advantage of attacking IND\$-CPA is *linear* (in p) and remains *negligible* while the advantages of attacking EKR$_p$ and KH$_p$ increase quadratically (in p) and are non-negligible.

5.3.3.3 Examples of Secure Schemes that withstand Key-Collision Attacks

Scheme CBC2 (viz., Figure 5.3) is not subject to the key-collision attacks described above. The reason is that this scheme (1) does not use unencrypted nonces directly as counters, and (2) does not make nonce encryptions available to an adversary. Thus knowledge of a nonce does not enable the adversary to collect ciphertexts for known- or predictable-plaintexts. Other schemes exist that have similar characteristics; e.g., stateful XECB [GD01]. Note that CBC2 uses two keys, while CTR1 and CTR2 only use one.

All schemes that withstand the key-collision attacks share the common trait: they randomize every input plaintext block using a pseudo-random value that remains unknown to the adversary before enciphering that block. Input randomization makes it impossible to construct constant plaintext - ciphertexts pairs, as required by the key-collision attacks above. Hence, the adversary would have to launch a *verifiable-plaintext* attack [LGSN89b, Gon90]; i.e., it must decrypt every harvested ciphertext using all T candidate keys, remove randomization, and check that the result is the known plaintext block. If the check passes for a candidate key, that key is a candidate for the encryption key. However, if the randomization value is unknown to the adversary, the verifiable-plaintext attack fails, since the randomization cannot be removed upon block decryption.

5.4 Related Work

Similarities and differences among various attacks become evident when one considers their definitions (e.g., differences in attack capabilities and strategy), scope (e.g., specific schemes and properties targeted), and cost of attack resources. We review related work in this light.

Key-Collision Attacks. The first attack described above is based on key collisions. These att acks, which require theability to exploit known- and predictable-plaintext capabilities, were successfully used against various ciphers during WW II [HS93]. Although similar capabilities were used (cal led "cribs" then),these attacks differ in scope and strategy from the ones presented in this paper, as they targeted only *stateless deterministic* encryption(now known to be broken in other ways).

The key-collision attacks described here also differ from Biham's atta cks [Bih96, Bih02]in three ways. First, Biham's attacks that can be used in practice (vs. his *theoretical* attacks)[10] have a different scope: they target *stateless deterministic* schemes, such as ECB, which cannot possibly be (IND$-CPA or IND-CPA)

[10]We note that Biham's attacks against probabilistic schemes and multiple encryptions are intended to be *theoretical*, as they could not possibly work in practice for standard block ciphers;e.g., two-key 3DES, AES-128.

secure. In contrast, our attacks are against properties of IND$-CPA schemes that have security proofs in both the asymptotic and concrete approaches. Second, in contrast to Biham's deterministic-scheme attack, our attacks work with *predictable* plaintext, not only wi th known, *constant* plaintext. Third, our attacks do not seek "birthday collisions" in keys. Such attacks would not be practical for today's standard key lengths; e.g. 128-bit AES keys. Instead, we tailor our attack parameters to obtain a non-negligible probability of collisions in these keys, since this is sufficient to show that these schemes are vulnerable.

Reduction Proofs. Recently, Koblitz and Menezes conjectured that non-tight reduction proofs for security-protocol properties defined in the asymptotic approach (viz., Section 5.2) leave unaddressed vulnerabilities to attack when these protocols are implemented in practice. They posed the following open problem: "Find an example of a natural and realistic protocol that has a plausible (non-tight) reductionist proof of security, and is also insecure when used with commonly accepted parameter sizes [KM06]." We answer this open question by demonstrating attacks against symmetric encryption schemes proven secure in the asymptotic approach and instantiated with standard block ciphers.

Koblitz and Menezes also define the goal of "existential key recovery" security informally and offer an intuitive explanation as to why it might be desirable that it be equivalent to that of "(universal) key recovery security". The attacks presented in this paper show that these two properties are not equivalent, despite of the intuitive appeal of such equivalence.

5.5 Proofs of Claims

5.5.1 Proof of Claim 5.1

Proof. To prove this claim, we first define an IND$_p-CPA property as follows:

Definition 5.6. (IND$_p-CPA) Let $\Pi(n) = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, let n be a security parameter, and let A be an adversary. Define

$$\begin{aligned}\mathsf{Adv}^{\text{IND\$}_p\text{-CPA}}_{\Pi(n)}(A) &\overset{\text{def}}{=} \Pr\left[k_1, ..., k_p \xleftarrow{r} \mathcal{K}(n) : A^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = 1\right] \\ &\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : A^{\$^{|\mathcal{E}_k(\cdot)|},...,\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right]\end{aligned}$$

Encryption scheme $\Pi(n)$ is IND$_p-CPA secure if for every probabilistic polynomial-time adversary A, $\mathsf{Adv}^{\text{IND\$}_p\text{-CPA}}_{\Pi(n)}(A)$ is negligible (as a function of n).

Then, we show the following lemma:

Lemma 5.1. For any PPT adversary B, we have:

$$\mathsf{Adv}^{\text{IND\$}_p\text{-CPA}}_{\Pi(n)}(B) \leq p \cdot \varepsilon(n)$$

We prove Lemma 5.1 by induction and a hybrid argument.

(1) For the base case where $p = 2$, we construct two adversaries A_1 and A_2 attacking IND$-CPA of $\Pi(n)$ as follows:

$\underline{A_1}$

1. A_1 chooses a random key k_1 from $\mathcal{K}(n)$ and whenever B sends a query m for the first oracle, returns $\mathcal{E}_{k_1}(m)$ to B.

2. Whenever B sends a query m for the second oracle, A_1 forwards m to A_1's own oracle and returns the answer to B.

3. When B outputs a bit b, A_1 outputs b.

Then, the advantage of A_1 is as follows :

$$\begin{aligned}\mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A_1) &= \Pr\left[k_1, k_2 \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot)}(n) = 1\right] \\ &\quad - \Pr\left[k_1 \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),\$^{|\mathcal{E}_{k_1}(\cdot)|}}(n) = 1\right]\end{aligned}$$

Similarly, we construct A_2 as follows:

$\underline{A_2}$

1. Whenever B sends a query m for the first oracle, A_2 forwards m to A_2's own oracle and returns the answer to B.

2. Whenever B sends a query m for the second oracle, A_2 chooses a random string of length $|\mathcal{E}_k(\cdot)|$ and returns it to B.

3. When B outputs a bit b, A_2 outputs b.

Then, the advantage of A_2 is as follows:

$$\mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A_2) = \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_k(\cdot),\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] - \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|},\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right]$$

Then, we have:

$$\begin{aligned}
\mathsf{Adv}^{\text{IND\$}_2\text{-CPA}}_{\Pi(n)}(B) &= \Pr\left[k_1, k_2 \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot), \mathcal{E}_{k_2}(\cdot)}(n) = 1\right] - \\
&\quad \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|}, \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&= \Pr\left[k_1, k_2 \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot), \mathcal{E}_{k_2}(\cdot)}(n) = 1\right] \\
&\quad - \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_k(\cdot), \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&\quad + \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_k(\cdot), \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&\quad - \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|}, \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&= \mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A_1) + \mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A_2) \leq 2 \cdot \varepsilon(n)
\end{aligned}$$

(2) For the inductive hypothesis where $p = \ell$, for any PPT adversary B, we assume the following:

$$\begin{aligned}
\mathsf{Adv}^{\text{IND\$}_\ell\text{-CPA}}_{\Pi(n)}(B) &= \Pr\left[k_1, ..., k_\ell \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot), ..., \mathcal{E}_{k_\ell}(\cdot)}(n) = 1\right] \\
&\quad - \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|}, ..., \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&\leq \ell \cdot \varepsilon(n)
\end{aligned}$$

Then, (3) for the inductive step where $p = \ell + 1$, we prove the following :

$$\begin{aligned}
\mathsf{Adv}^{\text{IND\$}_{\ell+1}\text{-CPA}}_{\Pi(n)}(B) &= \Pr\left[k_1, ..., k_{\ell+1} \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot), ..., \mathcal{E}_{k_{\ell+1}}(\cdot)}(n) = 1\right] \\
&\quad - \Pr\left[k \overset{r}{\leftarrow} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|}, ..., \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&\leq (\ell + 1) \cdot \varepsilon(n)
\end{aligned}$$

Again, to use a hybrid argument, we construct two adversaries — an IND\$-CPA adversary A_1 (working in much the same way as in the base case proof) and an adversary A_2 trying to break our inductive hypothesis. First, we describe how A_1 runs.

$\underline{A_1}$

1. A_1 chooses ℓ random keys $k_1, ..., k_\ell$ from $\mathcal{K}(n)$.

2. Whenever B sends a query m for any of i-th oracles where $i \in [1, \ell]$, A_1 returns $\mathcal{E}_{k_i}(m)$ to B.

3. Whenever B sends a query m for $(\ell + 1)$-th oracle, A_1 forwards m to its own oracle and returns the answer to B.

4. When B outputs a bit b, A_1 outputs b.

Then, the advantage of A_1 is as follows:

$$\begin{aligned}\mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A_1) &= \Pr\left[k_1, ..., k_{\ell+1} \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_{\ell+1}}(\cdot)}(n) = 1\right] \\ &\quad - \Pr\left[k_1, ..., k_\ell \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_\ell}(\cdot),\$^{|\mathcal{E}_{k_1}(\cdot)|}}(n) = 1\right]\end{aligned}$$

Next, we construct A_2 as follows:

$\underline{A_2}$

1. Whenever B sends a query m for any of the i-th oracles where $i \in [1, \ell]$, A_2 forwards m to its own i-th oracle and returns the answer to B.

2. Whenever B sends a query m for $(\ell+1)$-th oracle, A_2 chooses a random string of length $|\mathcal{E}_k(\cdot)|$ and returns it to B.

3. When B outputs a bit b, A_2 outputs b.

Then, the advantage of A_2 is as follows:

$$\begin{aligned}\mathsf{Adv}^{\text{IND\$}_\ell\text{-CPA}}_{\Pi(n)}(A_2) &= \Pr\left[k_1, ..., k_\ell \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_\ell}(\cdot),\$^{|\mathcal{E}_{k_1}(\cdot)|}}(n) = 1\right] \\ &\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : A^{\$^{|\mathcal{E}_k(\cdot)|},...,\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right]\end{aligned}$$

Now we let the advantage of B be expressed in terms of the advantages of A_1 and A_2, and then apply the inductive hypothesis as follows:

$$\begin{aligned}\mathsf{Adv}^{\text{IND\$}_{\ell+1}\text{-CPA}}_{\Pi(n)}(B) &= \Pr\left[k_1, ..., k_{\ell+1} \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_{\ell+1}}(\cdot)}(n) = 1\right] \\ &\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|},...,\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\ &= \Pr\left[k_1, ..., k_{\ell+1} \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_{\ell+1}}(\cdot)}(n) = 1\right] \\ &\quad - \Pr\left[k_1, ..., k_\ell \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_\ell}(\cdot),\$^{|\mathcal{E}_{k_1}(\cdot)|}}(n) = 1\right] \\ &\quad + \Pr\left[k_1, ..., k_\ell \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot),...,\mathcal{E}_{k_\ell}(\cdot),\$^{|\mathcal{E}_{k_1}(\cdot)|}}(n) = 1\right] \\ &\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|},...,\$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\ &= \mathsf{Adv}^{\text{IND\$-CPA}}_{\Pi(n)}(A_1) + \mathsf{Adv}^{\text{IND\$}_\ell\text{-CPA}}_{\Pi(n)}(A_2) \\ &\leq (\ell + 1) \cdot \varepsilon(n)\end{aligned}$$

Therefore, the induction holds for $p = \ell + 1$. The induction is complete.

Finally, we go back to our claim. For any PPT adversary B attacking the KH$_p$-CPA property of $\Pi(n)$, we can construct an adversary A_1 attacking the IND\$$_p$-CPA property (from Lemma 5.1) and an IND\$-CPA adversary A_2 (we skip the details of those constructions, due to the similarity to the previous case). Then, we apply

Lemma 5.1 and obtain:

$$\begin{aligned}
\mathsf{Adv}_{\Pi(n)}^{\mathrm{KH}_p\text{-CPA}}(B) &= \Pr\left[k_1, k_2, ..., k_\ell \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot), \mathcal{E}_{k_2}(\cdot), ..., \mathcal{E}_{k_\ell}(\cdot)}(n) = 1\right] \\
&\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_k(\cdot), ..., \mathcal{E}_k(\cdot)}(n) = 1\right] \\
&= \Pr\left[k_1, k_2, ..., k_\ell \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_{k_1}(\cdot), \mathcal{E}_{k_2}(\cdot), ..., \mathcal{E}_{k_\ell}(\cdot)}(n) = 1\right] \\
&\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|}, ..., \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&\quad + \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : B^{\$^{|\mathcal{E}_k(\cdot)|}, ..., \$^{|\mathcal{E}_k(\cdot)|}}(n) = 1\right] \\
&\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n) : B^{\mathcal{E}_k(\cdot), ..., \mathcal{E}_k(\cdot)}(n) = 1\right] \\
&\leq \mathsf{Adv}_{\Pi(n)}^{\mathrm{IND\$}_p\text{-CPA}}(A_1) + \mathsf{Adv}_{\Pi(n)}^{\mathrm{IND\$}\text{-CPA}}(A_2) \\
&\leq (p+1) \cdot \varepsilon(n)
\end{aligned}$$

□

5.5.2 Proof of Claim 5.2

Proof. Given a PPT adversary A attacking the EKR$_p$-CPA security of $\Pi(n)$, we construct a KH$_p$-CPA adversary B as follows.

B is given a set of p oracles that is either $(\mathcal{E}_{k_1}(\cdot), ..., \mathcal{E}_{k_p}(\cdot))$ for $k_1, ..., k_p \xleftarrow{r} \mathcal{K}(n)$, or $(\mathcal{E}_k(\cdot), ..., \mathcal{E}_k(\cdot))$ for $k \xleftarrow{r} \mathcal{K}(n)$. B must then determine which set of oracles it was given. B runs as follows:

<u>B</u>

1. If A requests a query m for i-th oracle, B forwards m to B's own i-th oracle, obtains an answer c, and returns c to A.

2. When A outputs a key k',

(a) B selects a message m' different from any query received from A.

(b) for each of p oracles given to B, B sends m' to the oracle and receives answer c.

(c) B decrypts c with k' and checks whether $\mathcal{D}_{k'}(c) = m'$. If the condition is true, B sets flag $f_i = \texttt{true}$. Otherwise, B sets flag $f_i = \texttt{false}$.

3. Then, B outputs a bit as follows:

- If $f_i = \texttt{true}$ for all $i \in [1,p]$, B outputs 0.
- Otherwise if $f_i = \texttt{true}$ for any $i \in [1,p]$, B outputs 1.
- Otherwise (i.e., $f_i = \texttt{false}$ for all $i \in [1,p]$), B outputs 0.

Clearly, the advantage of B is as follows:

$$\begin{aligned}\mathsf{Adv}^{\text{KH}_p\text{-CPA}}_{\Pi(n)}(B) &= \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : \begin{array}{c} \mathcal{D}_{k'}(\mathcal{E}_{k_i}(m')) = m' \text{ for } i \in [1,p] \\ \text{but not for all } i \in [1,p] \end{array}\right] \\ &\quad - \Pr\left[k \xleftarrow{r} \mathcal{K}(n), A^{\mathcal{E}_k(\cdot),...,\mathcal{E}_k(\cdot)}(n) = k' : \begin{array}{c} \mathcal{D}_{k'}(\mathcal{E}_{k_i}(m')) = m' \text{ for } i \in [1,p] \\ \text{but not for all } i \in [1,p] \end{array}\right]\end{aligned}$$

Let us consider the probabilities on the right hand side. Clearly, the second probability term is zero, since when A is given $(\mathcal{E}_k(\cdot), ..., \mathcal{E}_k(\cdot))$ for $k \xleftarrow{r} \mathcal{K}(n)$, we have:

either $f_i = \texttt{true}$ for all $i \in [1,p]$, *or*

$f_i = \texttt{false}$ for all $i \in [1,p]$

due to the correctness of the encryption scheme Π. Then, we have:

$$\begin{aligned}
\mathsf{Adv}_{\Pi(n)}^{\mathrm{KH}_p\text{-CPA}}(B) &= \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : \begin{array}{c} \mathcal{D}_{k'}(\mathcal{E}_{k_i}(m')) = m' \text{ for } i \in [1,p] \\ \text{but not for all } i \in [1,p] \end{array}\right] \\
&= \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : \begin{array}{c} \mathcal{D}_{k'}(\mathcal{E}_{k_i}(m')) = m' \text{ for } i \in [1,p] \\ \wedge k' \in \{k_1, ..., k_p\} \end{array}\right] \\
&\quad + \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : \begin{array}{c} \mathcal{D}_{k'}(\mathcal{E}_{k_i}(m')) = m' \text{ for } i \in [1,p] \\ \wedge k' \notin \{k_1, ..., k_p\} \end{array}\right] \\
&= \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : k' \in \{k_1, ..., k_p\}\right] \\
&\quad + \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : \begin{array}{c} \mathcal{D}_{k'}(\mathcal{E}_{k_i}(m')) = m' \text{ for } i \in [1,p] \\ \wedge k' \notin \{k_1, ..., k_p\} \end{array}\right]
\end{aligned}$$

where the last equation holds due to the correctness of Π. Then, since we have:

$$\mathsf{Adv}_{\Pi(n)}^{\mathrm{EKR}_p\text{-CPA}}(A) = \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \end{array} : k' \in \{k_1, ..., k_p\}\right],$$

by applying a union bound, we obtain:

$$\begin{aligned}
\left|\mathsf{Adv}^{\mathrm{EKR}_p\text{-CPA}}_{\Pi(n)}(A) - \mathsf{Adv}^{\mathrm{KH}_p\text{-CPA}}_{\Pi(n)}(B)\right| &= \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \\ (\mathcal{D}_{k'}(\mathcal{E}_{k_i}(m)) = m \text{ for } i \in [1,p]) \\ \wedge (k' \notin \{k_1, ..., k_p\}) \end{array} : \right] \\
&\leq \bigcup_i \Pr\left[\begin{array}{c} k_1, k_2, ..., k_p \xleftarrow{r} \mathcal{K}(n), \\ A^{\mathcal{E}_{k_1}(\cdot),\mathcal{E}_{k_2}(\cdot),...,\mathcal{E}_{k_p}(\cdot)}(n) = k' \\ (\mathcal{D}_{k'}(\mathcal{E}_{k_i}(m)) = m) \\ \wedge (k' \neq k_i) \end{array} : \right] \\
&\leq \sum_i \frac{1}{2^{l(n)}} = \frac{p}{2^{l(n)}}
\end{aligned}$$

□

Bibliography

[ABC$^+$06] Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, Bodo Moller, and David Pointcheval. Provably secure password-based authentication in TLS. In *ACM Symposium on Information, Computer, and Communications Security*, pages 35–45. ACM, 2006.

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP TCS*, pages 3–22, 2000.

[BBB$^+$07] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid (eds.). Recommendation for key management - part 1: General (revised). *NIST Special Publication 800-57*, March 2007. Table 4.

[BCP03] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Security proofs for an efficient password-based key exchange. In *ACM Conference on Computer and Communications Security*, pages 241–250. ACM Press, 2003.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of of symmetric encryption. In *Proceedings of the Symposium on Foundations of Computer Science*, 1997.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, pages 26–45, 1998.

[BDS$^+$03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, page 180, Washington, DC, USA, 2003. IEEE Computer Society.

[Bel97] Steven M. Bellovin. Probable plaintext cryptanalysis of the IP security protocols. In *Proceedings of the Symposium on Network and Distributed System Security*, 1997.

[Bel98] Mihir Bellare. Practice-oriented provable-security. In *Proc. of the Workshop on Information Security*, 1998.

[BFZ07] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. In *Proceedings of the ACM SIGCOMM*, 2007.

[BG85] R.W. Baldwin and W.C. Gramlich. Cryptographic protocol for trustable match making. In *IEEE Security and Privacy*, 1985.

[BHS04] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons. Concealing complex policies with hidden credentials. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 146–157, New York, NY, USA, 2004. ACM.

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their ralation to public-key cryptosystems. In *CRYPTO*, pages 283–298, 1998.

[Bih96] E. Biham. How to forge DES-encrypted messages in 2^{28} steps. TR CS0884, Technion Computer Science Department, 1996. (available at `http://www.cs.technion.ac.il/~biham/publications.html`).

[Bih02] E. Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. In *Information Processing Letters*, volume 84, pages 117–124, 2002.

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE Symposium on Research in Security and Privacy*, pages 72–84, 1992.

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[BM05] Walid Bagga and Refik Molva. Policy-based cryptography and applications. In *Financial Cryptography*, pages 72–87, 2005.

[BMC06] Walid Bagga, Refik Molva, and Stefano Crosta. Policy-based encryption schemes from bilinear pairings. In *ASIACCS*, page 368, 2006.

[BMP00] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology — Eurocrypt 2000*, pages 156–171. Springer-Verlag, 2000.

[Boy99] Maurizio Boyarsky. Public-key cryptography and password protocols: The multi-user case. In *ACM Conference on Computer and Communications Security*, pages 63–72. ACM Press, 1999.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology — Eurocrypt 2000*, pages 139–155. Springer-Verlag, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology — Crypto '93*, pages 232–249. Springer-Verlag, 1993.

[BR05] Mihir Bellare and Phillip Rogaway. *Introduction on Modern Cryptography*, 2005. (available at `http://www-cse.ucsd.edu/users/mihir/cse207/classnotes.html`).

[BS00] Piero Bonatti and Pierangela Samarati. Regulating service access and information release on the web. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 134–143, New York, NY, USA, 2000. ACM.

[BWNH+03] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) extensions. *IETF RFC 3546 — Standards Track*, 2003.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[Cha85] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[CHK+05] Ran Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie. Universally composable password-based key exchange. In *Advances in Cryptology — Eurocrypt 2005*, pages 404–421. Springer-Verlag, 2005.

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In *ASIACRYPT*, pages 293–307, 2004.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474, 2001.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT*, pages 68–86, 2003.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001.

[CLLZ05] Wei Chen, Shiding Lin, Qiao Lian, and Zheng Zhang. Sigma: A fault-tolerant mutual exclusion algorithm in dynamic distributed systems subject to process crashes and memory losses. In *PRDC '05: Proc. 11th Pacific Rim Intl. Symposium on Dependable Computing*, pages 7–14. IEEE Computer Society, 2005.

[DDM$^+$06] Anupam Datta, Ante Derek, John C. Mitchell, Ajith Ramanathan, and Andre Scedrov. Games and the impossibility of realizable ideal functionality. In *TCC*, pages 360–379, 2006.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[Dwo01] Morris Dworkin. *Special Publication 800-38A: Recommendation for block cipher modes of operation.* National Institute of Standards, U.S. Department of Commerce, December 2001.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[Fis99] Marc Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In *EUROCRYPT*, pages 432–445, 1999.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology — EUROCRYPT 2004.*, 2004.

[GD01] Virgil D. Gligor and Pompiliu Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *FSE*, 2001.

[Gen08] Rosario Gennaro. Faster and shorter password-authenticated key exchange. In *Theory of Cryptography Conference — TCC 2008*, pages 589–606. Springer-Verlag, 2008.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GH03] E. Grochowski and R. D. Halem. Technological impact of magnetic hard disk drives on storage systems. *IBM Syst. J.*, 42(2):338–346, 2003.

[Gil93] George Gilder. Metcalfe's law and legacy. *Forbes*, 1993.

[Gil08] Steve Gilheany. The decline of magnetic disk storage cost over the next 25 years, 2008. `http://www.berghell.com/whitepapers/Storage%20Costs.pdf` accessed 03/10/09.

[GL01] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In *CRYPTO*, pages 408–432, 2001.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *EUROCRYPT*, pages 524–543, 2003.

[GL06a] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. *ACM Trans. Information and System Security*, 9(2):181–234, 2006.

[GL06b] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. *J. Cryptology*, 19(3):241–340, 2006.

[Gli98] Virgil D. Gligor. Symmetric encryption with random counters. TR 3968, Department of Computer Science, University of Maryland, Dec. 1998.

[Gli08] Virgil D. Gligor. On the Evolution of Adversary Models in Cryptographic Protocols (Part II: the Fragility of Adversary Definitions), June 2008. Invited talk at FCS-ARSPA-WITS (http://profs.sci.univr.it/ vigano/fcs-arspa-wits08/).

[GLNS93] Li Gong, T. Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[Gol04] O. Goldreich. *Foundations of Cryptography: Volume 2 - Basic Applications*. Cambridge University Press, 2004.

[Gon90] Li Gong. Verifiable-text attacks in cryptographic protocols. In *INFOCOM*, pages 686–693, 1990.

[GPS09] Virgil D. Gligor, Bryan Parno, and Ji Sun Shin. From asymptotic proofs to network-adversary attacks against secure encryption schemes. CyLab Technical Report CMU-CyLab-09-003, Carnegie Mellon University, CyLab, Pittsburgh, PA, March 2009.

[HBStKO03] Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and tilarie K. Orman. Hidden credentials. In *WPES*, pages 1–8, 2003.

[Hel08] Helion Technology Limited. AES cores. `http://www.heliontech.com/aes.htm`, 2008.

[HFW] S. Kopsell H. Federrath and R. Wendolsky. Project: An.on - anonymity.on-line, jap. http://anon.inf.tu-dresden.de/.

[HK99] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. In *ACM Transactions on Information and System Security*, pages 230–268. ACM Press, 1999.

[HS93] F.H. Hinsley and A. Stripp. *Codebrakers: The inside Story of Bletchley Park*. Oxford Univ. Press, 1993.

[IEE05] IEEE P1363.2: Standard specifications for password-based public-key cryptographic techniques, 2005. See http://grouper.ieee.org/groups/1363/passwdPK/.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *FOCS*, pages 230–235, 1989.

[JG04] Shaoquan Jiang and Guang Gong. Password-based key exchange with mutual authentication. In *Selected Areas in Cryptography 2004*, pages 267–279. Springer-Verlag, 2004.

[KL08] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2008.

[KM06] Neal Koblitz and Alfred Menezes. Another look at "Provable Security". II. In *INDOCRYPT*, 2006.

[KOY01] Jonathan Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology — Eurocrypt 2001*, pages 475–494. Springer-Verlag, 2001.

[KS04] L. Kissner and D. Song. Private and threshold set-intersection. Tr, Carnegie Mellon University, 2004.

[KY00] Jonathan Katz and Moti Yung. Complete characterization of security notions for probabilistic private-key encryption. In *STOC*, pages 245–254, 2000.

[LB06] Thomas M. Lenard and Daniel B. Britton. *The Digital Economy Fact Book*. The Progress & Freedom Foundation, eighth edition, 2006.

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 182–189, New York, NY, USA, 2003. ACM.

[LGSN89a] T. Mark A. Lomas, Li Gong, Jerome H. Saltzer, and Roger M. Needham. Reducing risks from poorly chosen keys. In *ACM Symposium on Operating System Principles*, pages 14–18. ACM Press, 1989.

[LGSN89b] T. Mark A. Lomas, Li Gong, Jerome H. Saltzer, and Roger M. Needham. Reducing risks from poorly chosen keys. In *SOSP*, 1989.

[LL05] Jiangtao Li and Ninghui Li. Oacerts: Oblivious attribute certificates. In *ACNS*, pages 301–317, 2005.

[LL06] Jiangtao Li and Ninghui Li. A construction for general and efficient oblivious commitment based envelope protocols. In *ICICS*, pages 122–138, 2006.

[Mac02] Philip MacKenzie. The PAK suite: Protocols for password-authenticated key exchange, 2002. http://citeseer.ist.psu.edu/mackenzie02pak.html.

[Mea86] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE Symposium on Security and Privacy*, pages 134–137, 1986.

[MF01] David A. McGrew and Scott R. Fluhrer. Attacks on additive encryption of redundant plaintext and implications on internet security. In *Proceedings of the Workshop on Selected Areas in Cryptography*, 2001.

[Mic05] Daniele Micciancio. *The Dolev-Yao model, Lecture Notes, UC San Diego*, 2005. (available at `http://www-cse.ucsd.edu/classes/sp05/cse208/lec-dolevyao.html`).

[MPS00] Philip Mackenzie, Sarvar Patel, and Ram Swaminathan. Password-authenticated key exchange based on RSA. In *Advances in Cryptology — Asiacrypt 2000*, pages 599–613. Springer-Verlag, 2000.

[NT06] Samad Nasserian and Gene Tsudik. Revisiting oblivious signature-based envelopes. In *Financial Cryptography*, pages 221–235, 2006.

[NV04] Minh-Huyen Nguyen and Salil P. Vadhan. Simpler session-key generation from short random passwords. In *TCC*, pages 428–445, 2004.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[Rog04a] Phillip Rogaway. Nonce-based symmetric encryption. In *FSE*, pages 348–359, 2004.

[Rog04b] Phillip Rogaway. On the role definitions in and beyond cryptography. In *ASIAN*, pages 13–32, 2004.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, pages 433–444, 1991.

[RSRK07] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis. JouleSort: A balanced energy-efficient benchmark. In *ACM International Conference on Management of Data (SIGMOD)*, June 2007.

[SG08] Ji Sun Shin and Virgil D. Gligor. A new privacy-enhanced matchmaking protocol. In *15th Annual Network and Distributed System Security Symposium*, February 2008.

[SM00] Stuart G. Stubblebine and Catherine A. Meadows. Formal characterization and automated analysis of known-pair and chosen-text attacks. *IEEE Journal on Selected Areas in Communications*, 18(4):571–581, April 2000.

[SWY01] Kent E. Seamons, Marianne Winslett, and Ting Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *NDSS*, 2001.

[TX06] Gene Tsudik and Shouhuai Xu. A flexible framework for secret handshakes. In *Privacy Enhancing Technologies*, pages 295–315, 2006.

[vABHL03] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. *Advances in Cryptology - Eurocrypt*, pages 294–311, 2003.

[WL04] William H. Winsborough and Ninghui Li. Safety in automated trust negotiation. In *IEEE Symposium on Security and Privacy*, pages 147–160, 2004.

[WSJ00] W. H. Winsborough, K. E. Seamons, and V.E. Jones. Automated trust negotiation. *DARPA Information Survivability Conference and Exposition*, 1:88–102, 2000.

[WT02] Lawrence C. Washington and Wade Trappe. *Introduction to Cryptography: With Coding Theory*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

[YW03] Ting Yu and Marianne Winslett. A unified scheme for resource protection in automated trust negotiation. In *IEEE Symposium on Security and Privacy*, pages 110–122, 2003.

[YWS01] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *ACM Conference on Computer and Communications Security*, pages 146–155, 2001.

[ZN01] Kan Zhang and Roger Needham. A private matchmaking protocol, 2001. http://citeseer.nj.nec.com/71955.html.