Security Models and Proofs for Key Establishment Protocols

by

Eddie M. Ng

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2005

Author's declaration for electronic submission of a thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis we study the problem of secure key establishment, motivated by the construction of secure channels protocols to protect information transmitted over an open network. In the past, the purported security of a key establishment protocol was justified if it could be shown to withstand popular attack scenarios by heuristic analysis. Since this approach does not account for all possible attacks, the security guarantees are limited and often insufficient.

This thesis examines the provable security approach to the analysis of key establishment protocols. We present the security models and definitions developed in 2001 and 2002 by Canetti and Krawczyk, critique the appropriateness of the models, and provide several security proofs under the definitions. In addition, we consider the importance of the key compromise impersonation resilience property in the context of these models. We list some open problems that were encountered in the study.

Acknowledgements

I would like to sincerely thank my supervisor, Alfred Menezes, for his advice, guidance, patience and support. I would also like to thank my two readers, Doug Stinson and Edlyn Teske, for carefully reviewing my thesis. Their valuable feedbacks and suggestions are greatly appreciated.

I would like to thank the faculty and staff members of the C&O Department for making my graduate experience truly stimulating and rewarding. A special thanks goes to Marg Feeney for assisting me with various administrative tasks far beyond her duties.

Thanks to Wes, Patrick, Yi, Max, Bobby, Luis, James, Omran, Jenny, Lemmus and other friends and colleagues for making my stay at Waterloo enjoyable and memorable.

Finally, I would like to thank my family for their love and encouragement.

Contents

List of Figures

Chapter 1

Introduction

Key establishment protocols are procedures to derive a shared secret key by two or more parties over an openly distributed network. These protocols are important building blocks in cryptography, and in particular, are fundamentally required to build secure communications channels over insecure channels. In this thesis we examine security models and proofs for key establishment protocols. We use the provable security approach and emphasize mathematically rigorous models and proofs.

This thesis presents the Canetti-Krawczyk security model as the primary model for analyzing the security of key establishment protocols, and as such, draws heavily from the work of Canetti and Krawczyk [17, 18, 19]. Most of the thesis surveys the concepts, results and proofs from their work. Our new contributions are the analyses in Sections 4.1.1, 4.1.2, 4.1.4, 4.2.2, and 4.3.

We begin with a discussion of the key distribution problem in Section 1.1. Section 1.2 presents key establishment protocols as a solution to the problem. After presenting the necessary mathematical background in Section 1.3, Section 1.4 discusses approaches to analyzing the security of key establishment protocols. Section 1.5 documents notations used, and Section 1.6 provides the roadmap for the rest of the thesis.

1.1 The Key Distribution Problem

Providing security over open and large distributed networks has always been both intriguing and challenging. On one hand, the decentralized nature of these networks allows new devices and innovative services to be easily and cheaply deployed, nurturing them as common technology platforms with unprecedented business opportunities. On the other hand, these flexibilities present perfect opportunities for malicious individuals to perform disruptive and otherwise unethical tasks. Let us take the case of the Internet, for example. The original design of the Internet provides no reliability and security guarantees. Thus, for example, connection failures and data congestions are commonplace, as are router compromises and packet "sniffing" (eavesdropping of network data). Despite these concerns, the pervasive Internet that we now have has become the preferred medium to conduct electronic commerce and other online business transactions. Malicious users may attempt to obtain valuable business information or interrupt operations of these activities for psychological or commercial rewards.

Are there ways to enable "secure channels" over insecure networks? More precisely, are there ways to emulate a secure wire in an open environment, subject to adversarial actions? We may get a better idea on how to answer the question by first identifying specific goals that this secure communicating channel should achieve. Five primary goals are often suggested [39, Chapter 1]:

1. Data confidentiality — ensuring that data is kept secret from all except those who are authorized to see them.

2. Data integrity — ensuring that data is not tampered by unauthorized means.

3. Data origin authentication — proving the validity of the source of data.

4. Entity authentication — proving the validity of a party's identity.

5. Non-repudiation – preventing a party from denying previous actions or commitments.

Let us now examine data confidentiality. To achieve data confidentiality in a session established by party A with intended recipient B, one may use a cryptographic algorithm, called *symmetric encryption*, which given a plaintext message m, produces a ciphertext c that A can subsequently send to B over the network. This ciphertext has the property that it does not reveal any information about the original plaintext to anyone except A and B. This property can be achieved since the algorithm requires A and B to share a piece of secret information, known as a *shared session key*, that is fresh and unique for each session. Of course, one can use *public-key encryption* instead to avoid the use of shared session keys, at the expense of operating at a speed that is too inefficient to be practical for transferring large amount of data. Similar reasonings show that the other four goals of secure communications can be accomplished provided that shared session keys are readily available to each pair of parties. Therefore, it is of great interest to devise effective mechanisms to establish these shared session keys — the so-called key distribution problem.

1.2 A Solution: Key Establishment Protocols

One solution to the key distribution problem is to have a trusted party generate all session keys and physically transport these keys to the intended parties. However, there are at least four drawbacks which render this approach unscalable. First, if n parties are present in a system, then $\binom{n}{2}$ keys have to be generated initially, and each party has to store $n-1$ keys. Also, if a new party joins the system, then n new keys need to be generated and transported. Third, if two parties want to establish subsequent sessions, then new session keys need to be acquired from the trusted party. Finally, if the trusted party is compromised or is itself a malicious party, then all hopes of security are gone.

It is also possible to have the two parties meet in a secret location for a face-to-face key establishment, without the need for a trusted party. Suppose the parties want to agree upon a k-bit shared key. What they can do is toss a fair coin k times in sequence, recording the outcome (0 for head and 1 for tail) of each toss. The concatenation of the

outcomes can then be used as the shared key. Of course, this approach suffers from the same scalability problems as the previous approach, although it no longer relies on a trusted party. As a result, the only logical approach is to develop online mechanisms to solve the key distribution problem.

Key establishment protocols are procedures to securely establish shared session keys over distributed networks. We assume that a public-key infrastructure is in place, which may or may not be used by the key establishment protocols. There are two major techniques of key establishment, key transport and key agreement, which roughly correspond to the two previously mentioned physical techniques. There have been numerous proposals for key establishment protocols, but unfortunately many of them are vulnerable to unanticipated security problems. Since many cryptographic protocols cannot operate, let alone provide security, without initial secure key establishments, it is extremely crucial to understand how to analyze the security of key establishment protocols.

Before we give further details on how to solve this problem, let us first present the necessary mathematical background required for the rest of the thesis.

1.3 Mathematical Preliminaries

1.3.1 Algorithms and Complexity

In theoretical cryptography, computational problems and tasks are formulated using the Turing machine model of computation. When we speak of "solving a computational problem", we mean constructing a Turing machine, which we call an algorithm, whose output is the solution to the problem. An algorithm is called **probabilistic** if its operation depends on random tosses. A **polynomial-time algorithm** (also called an **efficient** algorithm) is an algorithm whose expected running time is $O(n^c)$, where c is a constant and n is the size of the input.

A computational problem is said to be **computationally intractable** if there exists no polynomial-time algorithm for solving it. Many computational problems are believed to

be computationally intractable, and are used as important building blocks in cryptography.

Reduction is a technique to compare the difficulty of two computational problems. Here we look at reductions that preserve polynomial-time solvability.

Definition 1.3.1 *Let P_1 and P_2 be two computational problems. We say that P_1* **poly-time reduces** *to P_2, written $P_1 \leq_p P_2$, if there exists a polynomial-time algorithm for solving P_1, which uses as an oracle a polynomial-time algorithm for solving P_2.*

Reduction is a powerful idea. Intuitively, if $P_1 \leq_p P_2$, then the computational difficulty of P_2 must be at least as hard as P_1. Consequently, if P_2 is solvable in poly-time, then so is P_1. Similarly, if P_1 is intractable, then P_2 must also be intractable.

1.3.2 Polynomial-time Distinguishability.

A discrete **random variable** X is an ordered pair (S, D), where S is a finite sample space and D is a probability distribution on S. If A is a (probabilistic) algorithm, then $A(x)$ denotes the probability distribution of the outputs of A on input x, and $A(X)$ denotes the probability distribution of the outputs of A on random variable X. Thus,

$$\begin{aligned} \Pr(A(X) = s) &= \sum_{x \in S} \big[\Pr(A(X) = s | X = x) \cdot \Pr(X = x)\big] \\ &= \sum_{x \in S} \big[\Pr(A(x) = s) \cdot \Pr(X = x)\big]. \end{aligned}$$

A sequence of indexed random variables $\{X_n\}_{n \in N}$ is called a **distribution ensemble**. Typically, X_n is a random variable over the space $\{0,1\}^{r(n)}$, where $r : N \to N$ is an increasing polynomial function.

Definition 1.3.2 *A function $f(n)$ is called* **negligible** *(in n) if for every polynomial function $p(n)$, there exists an n_0 such that $f(n) < \frac{1}{p(n)}$ for every $n \geq n_0$.*

For example, $2^{-n/2}$ and $n^{-\log n}$ are negligible functions, whereas n^{-2} and $1/2 + 2^{-n}$ are not.

Definition 1.3.3 *Let X and Y be two random variables with a common sample space S. Then the* **statistical distance** *between X and Y is*

$$dist(X,Y) \stackrel{def}{=} \frac{1}{2}\sum_{s\in S} |\Pr(X=s) - \Pr(Y=s)|.$$

Proposition 1.3.4 *Let X, Y and Z be random variables with a common sample space S, and let A be any algorithm. Then*

(i) $dist(X,Y) \leq dist(X,Z) + dist(Z,Y)$,

(ii) $dist(X,Y) \geq dist(A(X),A(Y))$.

Proof.

$$\begin{aligned}
(i)\ dist(X,Y) &= \frac{1}{2}\sum_{s\in S} |\Pr(X=s) - \Pr(Y=s)| \\
&\leq \frac{1}{2}\sum_{s\in S}\big(|\Pr(X=s) - \Pr(Z=s)| + |\Pr(Z=s) - \Pr(Y=s)|\big) \\
&= \frac{1}{2}\sum_{s\in S} |\Pr(X=s) - \Pr(Z=s)| + \frac{1}{2}\sum_{s\in S} |\Pr(Z=s) - \Pr(Y=s)| \\
&= dist(X,Z) + dist(Z,Y). \\
(ii)\ dist(A(X),A(Y)) &= \frac{1}{2}\sum_{s\in S} |\Pr(A(X)=s) - \Pr(A(Y)=s)| \\
&= \frac{1}{2}\sum_{s} |\sum_{z} \Pr(A(z)=s)(\Pr(X=z) - \Pr(Y=z))| \\
&\leq \frac{1}{2}\sum_{s}\sum_{z} |\Pr(A(z)=s)(\Pr(X=z) - \Pr(Y=z))| \\
&= \frac{1}{2}\sum_{z}\Big(\sum_{s} \Pr(A(z)=s)\Big)|\Pr(X=z) - \Pr(Y=z)| \\
&= \frac{1}{2}\sum_{z} |\Pr(X=z) - \Pr(Y=z)| \ \ (\text{since } \sum_{s}\Pr(A(z)=s) = 1) \\
&= dist(X,Y).
\end{aligned}$$

□

We have defined statistical distance, but we have not explained what it means yet. Intuitively, the statistical distance between two random variables X and Y captures how *similar* they are to each other. The next natural question then is whether we can design efficient algorithms to tell whether two random variables are statistically far apart or not. To do so, we may want to construct a **distinguisher** $\mathcal{D}$, an algorithm whose input is either X or Y, and whose boolean output indicates its guess as to which random variable the input comes from. If $\mathcal{D}$ always produces the right guess, then the random variables must be far apart. However, if $\mathcal{D}$ is well-designed but still performs poorly, then the random variables must be statistically very close to each other. In cryptography, the inability to distinguish between two random variables in polynomial time forms the basis upon which cryptographic primitives are built.

Definition 1.3.5 *Let $E_1 = \{X_k\}$ and $E_2 = \{Y_k\}$ be two distribution ensembles. Then E_1 and E_2 are* **polynomially indistinguishable** *if for every probabilistic polynomial-time distinguisher $\mathcal{D}$,*

$$dist(\mathcal{D}(X_k), \mathcal{D}(Y_k)) = |\Pr(\mathcal{D}(X_k) = 1) - \Pr(\mathcal{D}(Y_k) = 1)|$$

is negligible in k.

Suppose we have a distinguisher $\mathcal{D}$ on ensembles E_1 and E_2, and we want to test how good it is. We can toss a coin $b \in_R \{0, 1\}$. If $b = 0$, we set $Z_k \leftarrow X_k$; otherwise we set $Z_k \leftarrow Y_k$. We then feed Z_k as the input to $\mathcal{D}$. Now $\mathcal{D}$'s job is to guess correctly which random variable the input comes from. $\mathcal{D}$'s success probability is

$$\begin{aligned}
\Pr(\mathcal{D}(Z_k) = b) &= \Pr(\mathcal{D}(Z_k) = 0 | b = 0) \cdot \Pr(b = 0) + \Pr(\mathcal{D}(Z_k) = 1 | b = 1) \cdot \Pr(b = 1) \\
&= \Pr(\mathcal{D}(X_k) = 0) \cdot \Pr(b = 0) + \Pr(\mathcal{D}(Y_k) = 1) \cdot \Pr(b = 1) \\
&= \frac{1}{2}\Big(\Pr(\mathcal{D}(X_k) = 0) + \Pr(\mathcal{D}(Y_k) = 1)\Big) \\
&= \frac{1}{2} + \frac{1}{2}\Big(\Pr(\mathcal{D}(Y_k) = 1) - \Pr(\mathcal{D}(X_k) = 1)\Big).
\end{aligned}$$

Of course, we want $\mathcal{D}$'s success probability to be as close to 1 as possible. Since we expect that any naive algorithm can achieve success probability of $\frac{1}{2}$, we are more interested in

the quantity $|\Pr(\mathcal{D}(Z_k) = b) - \frac{1}{2}|$, which measures $\mathcal{D}$'s success probability above $\frac{1}{2}$. We call such quantity the **advantage** of $\mathcal{D}$, denoted $\mathbf{Adv}_{\mathcal{D}}(k)$. It can readily be seen that

$$\mathbf{Adv}_{\mathcal{D}}(k) = \frac{1}{2}\Big|dist(\mathcal{D}(X_k), \mathcal{D}(Y_k))\Big| ;$$

that is, two ensembles are polynomially indistinguishable if and only if the advantage of any distinguisher for the ensembles is negligible.

Example 1.3.6 *Let U_k be the uniform distribution over $\{0,1\}^k$, and let $G : \{0,1\}^k \to \{0,1\}^k$ be a permutation. Then $\{U_k\}$ and $\{G(U_k)\}$ are polynomially indistinguishable.*

Proof. By Proposition 1.3.4(ii), it suffices to show that the statistical distance between $X = U_k$ and $Y = G(U_k)$ is negligible in k. In fact,

$$\begin{aligned} dist(X,Y) &= \frac{1}{2}\sum_{s\in\{0,1\}^k} |\Pr(X = s) - \Pr(Y = s)| \\ &= \frac{1}{2}\sum_{s\in\{0,1\}^k} |\frac{1}{2^k} - \frac{1}{2^k}| = 0, \end{aligned}$$

which is clearly negligible. □

Example 1.3.7 *Let P_k be the uniform distribution of all k-bit primes, and C_k be the uniform distribution of all k-bit composite numbers. Then $\{P_k\}$ and $\{C_k\}$ are polynomially distinguishable.*

Proof. We show that the Soloway-Strassen primality test [39, Chapter 4], denoted $\mathcal{SS}$ and shown in Figure 1.1, can be used as a polynomial-time distinguisher with non-negligible distinguishing probability. It is trivial to see that $\mathcal{SS}$ is a poly-time algorithm in k. Moreover, if $\mathcal{SS}$ outputs composite, then the input must be composite, and if $\mathcal{SS}$ outputs prime, then the input is composite with probability $\leq 1/2^k$. Thus,

$$\begin{aligned} \mathbf{Adv}_{\mathcal{SS}}(k) &= \frac{1}{2}\Big|\Pr(\mathcal{SS}(C_k) = 1) - \Pr(\mathcal{SS}(P_k) = 1)\Big| \\ &\geq \frac{1}{2}(1 - \frac{1}{2^k}) = \frac{1}{2} - \frac{1}{2^{k+1}} \end{aligned}$$

INPUT: $n \in \{0,1\}^k$

OUTPUT: 0 ("n is composite") or 1 ("n is prime")

1: **for** $i = 1$ to k **do**
2: pick $a \leftarrow_R [1, n-1]$
3: compute $d \leftarrow \gcd(a, n)$
4: **if** $d > 1$ **then**
5: output 0
6: **end if**
7: compute $t_1 \leftarrow a^{\frac{n-1}{2}} \mod n$ and $t_2 \leftarrow \left(\frac{a}{n}\right)$
8: **if** $t_1 \not\equiv t_2 \pmod{n}$ **then**
9: output 0
10: **end if**
11: **end for**
12: output 1

Figure 1.1: The Soloway-Strassen primality test $\mathcal{SS} : \{0,1\}^k \rightarrow \{0,1\}$

is clearly non-negligible. Thus, the two ensembles are polynomially distinguishable. □

Sometimes, a distinguisher may need access to the adversary and other external resources during execution. When the formalism is necessary, we write $\mathcal{D}_A^g(\cdot)$ to mean that $\mathcal{D}$ is a distinguisher which uses adversary A as a subroutine, and is given oracle access to function g.

1.3.3 Intractability Assumptions

Let p be a k-bit prime. Let $\mathbb{Z}_p^*$ be the multiplicative group of order $p-1$. Let S_q be a large subgroup of $\mathbb{Z}_p^*$ of prime order $q|p-1$, such that performing the Number Field Sieve [26] in $\mathbb{Z}_p^*$ and the Pollard's Rho method [39, Chapter 3] in S_q take roughly equal amount of time[1]. Let g be a randomly chosen generator of S_q. Let p, q and g be public information.

The **Computational Diffie-Hellman Problem** (CDHP) is as follows. Given g^a, g^b, compute g^{ab}. The **Decisional Diffie-Hellman Problem** (DDHP) is as follows. Given g^a, g^b, and g^c, decide whether $c \equiv ab \pmod{q}$ or not. Both of these problems are believed to be computationally intractable. Of course, they can be easily solved by exhaustive searches if p or the order of the subgroup S_q is small. Thus, these parameters, called **security parameters**, must meet certain minimum values to be of practical use. For a large security parameter k, we have the following.

Definition 1.3.8 (The CDH Assumption) *The Computational Diffie-Hellman Assumption is that, for all probabilistic poly-time algorithms A,*

$$\Pr[A(g^a, g^b) = g^{ab} : a, b \leftarrow_R \mathbb{Z}_q]$$

is negligible in k.

[1]The Number Field Sieve is the best method known for solving the discrete logarithm problem in $\mathbb{Z}_p^*$, whereas Pollard's rho method is the best method known for solving the discrete logarithm problem in a prime order subgroup S_q of $\mathbb{Z}_p^*$. The expected running times of the two methods are $L_p[\frac{1}{3}, 1.923]$ and $O(\sqrt{q})$ respectively. Thus, one typical choice consists of picking p as a 1024-bit number and q as a 160-bit number.

Definition 1.3.9 (The DDH Assumption) *The Decisional Diffie-Hellman Assumption is that, for all probabilistic poly-time distinguishers A,*

$$|\Pr[A(g^a, g^b, g^{ab}) = 1 : a, b \leftarrow_R \mathbb{Z}_q] - \Pr[A(g^a, g^b, g^c) = 1 : a, b, c \leftarrow_R \mathbb{Z}_q]|$$

is negligible in k.

Both the CDH and the DDH assumptions are frequently used in cryptography. Although there are no proofs that these problems are indeed computationally intractable, all existing techniques for solving them take super-polynomial time (see for example Chapter 3 of [39]). Thus, most people are confident that these assumptions are indeed valid.

Nevertheless, we can examine the CDHP and DDHP more closely to gain a better understanding of the problems. First, we can compare their relative computational difficulties. The DDHP is definitely an easier problem than the CDHP. To see this, suppose we are confronted with deciding whether the triple (g^a, g^b, g^c) is a DDHP instance, but with access to a CDHP oracle $\mathcal{O}$. We instantiate $\mathcal{O}$ on instance (g^a, g^b) and receive an answer $h = g^{ab}$. Next we check whether $h \stackrel{?}{=} g^c$, and return yes if and only if they are the same. In other words, we have just shown that DDHP $\leq_P$ CDHP.

Second, we can test whether these problems possess the useful property of random self-reducibility. A computational problem is **random self-reducible** if solving any particular instance of the problem can be efficiently reduced to solving a random instance of the same problem. Self-reducibility is important because it gives assurances that almost all instances of the problem have equivalent computational complexity. Thus, if one has enough evidence that some instances of the CDHP are hard, then this implies that almost all instances of the problem are also hard. Both the CDHP and the DDHP are self-reducible.

Lemma 1.3.10 *The CDHP is random self-reducible.*

Proof. Given an instance (X, Y) of CDHP, where $X = g^x$ and $Y = g^y$, we randomly select $r, s \in_R [0, q-1]$, and create another instance (X', Y'), where $X' \leftarrow Xg^r$ and $Y' \leftarrow Yg^s$. Since g^r and g^s range over S_q uniformly, the instance (X', Y') is random and is uniformly

distributed over the space $S_q \times S_q$. Suppose there is an oracle to compute the solution Z' for the instance (X', Y'). Since $Z' = g^{(x+r)(y+s)} = g^{xy+xs+ry+rs}$, one can simply compute

$$Z' X^{-s} Y^{-r} g^{-rs} = g^{xy}$$

to obtain the solution Z for the original instance (X, Y). □

Lemma 1.3.11 *The DDHP is random self-reducible.*

Proof. Given an instance (X, Y, Z) of DDHP, where $X = g^x, Y = g^y$, and $Z = g^z$, we randomly select $r \in_R [1, q-1]$ and $s, t \in_R [0, q-1]$, and create another instance (X', Y', Z'), where $X' \leftarrow X^r g^s, Y' \leftarrow Y g^t$, and $Z' \leftarrow Z^r Y^s X^{rt} g^{st}$. We can rewrite

$$X' = g^{xr+s}, Y' = g^{y+t}, Z' = g^{zr+ys+xrt+st}.$$

It is easy to see that X' and Y' are independent, and both are random.

Suppose there is an oracle that can distinguish whether the instance (X', Y', Z') is a valid Diffie-Hellman triple. The answer to the original instance would be to return the same decision as the oracle, since if (X', Y', Z') is a yes-instance, then

$$\begin{aligned} g^{(xr+s)(y+t)} &= g^{zr+ys+xrt+st} \\ \implies (xr+s)(y+t) &\equiv zr + ys + xrt + st \pmod{q} \\ \implies xyr &\equiv zr \pmod{q} \\ \implies xy &\equiv z \pmod{q}, \end{aligned}$$

so (X, Y, Z) must also be a yes-instance. Similarly, if (X', Y', Z') is a no-instance, then $xy \not\equiv z \pmod{q}$, so (X, Y, Z) must also be a no-instance. □

1.4 The Provable Security Approach

Recall that a key establishment protocol is a procedure to allow two or more parties to establish a shared secret key. Suppose we have constructed a key establishment protocol.

The next obvious task is to determine whether the protocol is secure or not. There are two ways in which this can be accomplished.

The first approach is generically called the **heuristic analysis** approach. In this approach, a protocol is believed to be secure if it can be shown to withstand a number of existing popular attack scenarios and satisfy a number of desirable properties. This approach has the benefit that insecure protocols can be easily detected. Also, the analysis method is easy to understand and accessible to practitioners.

The second approach is called the **provable security** approach. This approach was first undertaken by Goldwasser and Micali [24] in establishing the security of asymmetric encryption schemes, but has since been generalized and used in all areas of cryptography. In the provable security paradigm, the most central problem in protocol design is to construct secure cryptographic schemes from strong cryptographic primitives. This is so because a cryptographic protocol is clearly insecure if it uses insecure or inappropriate cryptographic primitives. However it may still be insecure even though it uses the strong and correct primitives, if it is poorly designed. As such, the security of the underlying primitives and the way they are used both play a vital role in the security of the cryptographic scheme. Thus, if a well-designed protocol is indeed secure, then one should be able to write down a proof of security, which says that the *only* way that the protocol is broken is if one can break its underlying cryptographic assumptions. In other words, the proof establishes that the protocol is secure as long as the underlying primitive cannot be broken.

Before we give further details about the provable security technique, let us first explain why security guarantees established by the heuristic method are limited and often insufficient.

THE NEED FOR RIGOROUS TREATMENT. The main problem with the heuristic analysis approach is that it lacks formal foundations, and thus can only be used to establish informal arguments about security. In particular, the approach suffers from at least two major problems.

The first and the most obvious problem is that the heuristic approach makes no security

guarantees against the infinite number of possible future attack scenarios. Instead, it advocates the belief that a protocol which has not been broken by existing attacks is likely secure. Since cryptography is concerned with the design and construction of schemes to perform its prescribed functionality in the presence of malicious users, it is unreasonable to believe that an attacker will only act within its current knowledge of attacks and not come up with new attack strategies. In fact, key establishment protocols have historically been particularly vulnerable to unforeseen attacks, and many of them have taken years before they were demonstrated to have subtle security faults.

The second problem is that the approach provides no clear framework to formally describe what it means for a protocol to be "secure", and what constitutes an "attack". It is understandable that most practitioners are not concerned about formal definitions, since having the correct definitions does not necessarily help build better protocols. To them, formal security definitions may even seem to obstruct their creative pursuits in discovering new attacks. However, if one does not clearly specify the computational limits of the attacker, for example, then all public key encryption schemes are considered insecure, since the attacker with unlimited computational power can always obtain the decryption key by simply mounting an exhaustive search. Thus, if one wants to study cryptography rigorously, then models, definitions, and proofs are definitely of foremost importance, and a new way of thinking about "security" is necessary.

Several methods have been proposed to improve upon the heuristic analysis approach. One method is to employ symbolic verification logic from the computer science community. Another approach is to develop automatic model checking tools against protocol descriptions to speed up the analysis process. Others have proposed design guidelines from their hard-earned experience.

The symbolic verification method was first proposed by Burrows, Abadi and Needham [14]. Known as the BAN logic, this method attempts to establish security by first postulating a set of axioms, which are logical, syntactic rules to capture possible protocol behaviours and protocol participants' actions. Next, protocol descriptions are rewritten as logical formulae. Finally, axioms are applied systematically to the logical formulae to

determine whether the protocol satisfies certain desirable properties. Although this technique is useful as an ad-hoc analysis tool, it cannot be used to uncover possible attacks on a protocol. Moreover, the protocol translation and analysis stages are error-prone, since the logical system is often unsound and relies too much on intuition and implicit assumptions.

The second method, called the model checking approach [28, 38], can be seen as an improvement over the first approach. It first expresses a protocol in terms of a finite state system. Next, the entire state space is explored to determine whether properties of the system are satisfied or not. The difference between this method and the former method is that these properties can either be desirable or undesirable. As a result, the model checking approach is often able to uncover errors in a protocol. However, the approach lacks mathematical foundations and can become inefficient as the state space grows rapidly.

The third method of following design guidelines has been particularly useful in reducing errors during the protocol design stage, but it clearly cannot be used as an analysis technique. In summary, all three methods fail to address the main problems associated with the heuristic analysis approach.

THE PROVABLE SECURITY APPROACH. Provable security addresses security in a more convincing manner. A cryptographic protocol is said to achieve provable security if there is a formal security definition for that scheme under an appropriate adversarial model, and there is a proof that the protocol satisfies the definition provided that some well-studied assumptions hold. Thus, a complete proof of security consists of five steps. The first two steps specify the security model and definition. The third step states all computational and cryptographic assumptions. The fourth step gives a security proof under the stated assumptions. The proof is finally given further analysis and interpretations. We now give details on each step.

1. **Specify security model** — The security model should accurately capture the characteristics of the network, the behaviors of honest entities, and the adversary's attack capabilities and goals. Of course, these characteristics, capabilities and goals vary under different settings and environments. For example, there may be situations where

an adversary's goal is to obtain the decryption key of an encryption system, rather than simply being able to decrypt a message. A strong security notion, however, should anticipate an adversary whose attack capabilities are as powerful as possible, but whose adversarial goals are as weak as possible. A major task for cryptographers is in identifying the appropriate security models for specific situations.

2. **Formulate security definition** — The security definition states that a protocol is considered secure if it can withstand the adversary's goals under its attack capabilities.

3. **State assumptions** — If the construction of a protocol relies on any computational or cryptographic primitives, one should precisely state all respective intractability and security assumptions. Of course, the fewer the number and the weaker the assumptions are, the more secure the resulting cryptographic protocol is.

4. **Provide security proof** — The security proof takes the form of a reductionist argument between two computational problems P_1 and P_2. The first is a problem concerning the stated assumptions. The second is a problem to "break" the stated security definition of a particular protocol. The reductionist argument then establishes that $P_1 \leq_P P_2$; that is, if there is an efficient algorithm for solving P_2, then it can be used as a subroutine to solve P_1 in polynomial time. Since P_1 is assumed to be intractable, it follows that P_2 cannot be broken; that is, the protocol is secure under the security definition.

5. **Check and interpret proof** — Although provable security has been in widespread use, proof writing remains a difficult task, comprehensible only by academics and experts in the area. Thus, an important step in provable security is in checking the security proofs for correctness. Once a proof has been verified, another crucial aspect is in understanding and interpreting the results. What real security assurances does the definition give? Are the adversarial goals and attack models appropriate? Are the underlying assumptions too strong? What should the minimum security parameters

be as a result of the analysis in the proof?

Provable security is not without problems and limitations. One problem is that the security model can only capture properties within the realms of computational complexity. As a result, side-channel attacks and implementation problems, which account for a fair number of real attacks, are not covered by the model. Another problem is that security proofs are often too long and difficult to understand and interpret. Finally, practitioners often misinterpret reduction proofs as absolute proofs, without understanding their underlying assumptions and subtleties.

A large number of key establishment protocols have been proposed in the past two decades, but the security of most was only supported heuristically. This thesis examines security models and proofs for key establishment protocols from a provable security point of view.

1.5 Notations

We use the following notations throughout the thesis.

$A, B, \cdots$	honest parties A, B, $\cdots$
$\mathcal{A}$	an adversary in the authenticated-links model
$D_A(m)$	public-key decryption of m using A's private key
E	possibly dishonest party E
$E_A(m)$	public-key encryption of m using A's public key
$E_K(m)$	symmetric encryption of m using K as the symmetric key
$\mathrm{kdf}_K(m)$ or $\mathrm{kdf}(K; m)$	key derivation function on m using key K
$\mathrm{MAC}_K(m)$ or $\mathrm{MAC}(K; m)$	message authentication code on m using key K
N_A	nonce generated by A
$\mathrm{prf}_K(m)$	pseudorandom function on m using key K
$\mathrm{sig}_A(m)$	A's signature on message m
$\mathcal{U}$	an adversary in the unauthenticated-links model

If S is a probability distribution, then $x \leftarrow S$ denotes an assignment to x from the probability distribution S. If S is a finite set, then $x \leftarrow_R S$ denotes an assignment to x from a randomly chosen element in S.

1.6 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 gives a survey of key establishment techniques and provides examples of existing protocols. In Chapter 3, we provide a formal security model for key establishment protocols. Chapter 4 critiques the model and compares it with previous models. Chapter 5 examines security proofs for various key establishment protocols under these models. Finally, Chapter 6 concludes the thesis.

Chapter 2

Key Establishment Protocols

Cryptography is concerned with the study of secure protocol design. In modern cryptography, protocols are often created by combining known cryptographic primitives, such as encryption for data secrecy, MAC functions for message authentication, hash functions for integrity checks, and so on. However, these cryptographic primitives are not operational unless special randomly-selected values, called **secret keys**, are securely distributed to the parties who want to engage in the protocols.

Secret keys can be pre-distributed in some out-of-band (non-cryptographic) manner by, for example, courier delivery or closed room meetings involving the participants. However, these methods are clearly inefficient and non-scalable, leading cryptographers to seek better solutions. **Key establishment** (KE) is a cryptographic mechanism whereby a secret key becomes available to two or more parties. It is a fundamental building block in cryptography. This chapter shall give a broad overview of the field.

Key establishment is an old and diverse subject with an immense published literature dating from Diffie and Hellman's seminal paper [21] in 1976. However, no comprehensive treatment of the subject existed until Boyd and Mathuria's book [13] was published in 2003. Prior to that, the most extensive survey of works in the field could be found in the *Handbook of Applied Cryptography* by Menezes, van Oorschot and Vanstone [39]. Both books contain extensive references to the literature.

We begin in Section 2.1 by introducing some basic concepts in key establishment protocols. Examples of key establishment techniques are then presented in Section 2.2. In Section 2.3, we discuss their various desirable properties.

2.1 Basic Concepts

TAXONOMY. As we have already seen, the main objective of a key establishment protocol is to make available a secret key to two or more parties. When the secret key is to be established among two parties, we call the protocol a **two-party** key establishment protocol; when it is to be established among $n > 2$ parties, we call it a **multi-party** key establishment protocol.

There are various ways in which key establishment can be achieved. First, symmetric or asymmetric techniques can be employed. By **symmetric** we mean that the parties make use of some *a priori* long-term secret information shared among the protocol parties. By **asymmetric** we mean that the parties make use of some *a priori* authenticated (but non-secret) information, usually in the form of public keys for a public-key cryptographic mechanism.

Next, key establishment can be accomplished using key transport or key agreement. In a **key transport** protocol, one party first produces the secret key, then securely transfers it to the other parties. This technique roughly corresponds to that of using a courier service in our earlier example — one party first produces a secret key, then couriers it to another party. Intuitively, this technique may seem "unfair" to the second party because it has no influence over how the key is produced. In a **key agreement** protocol, the secret key is derived as a function of the information associated with each participating party, such that neither party can predetermine the value before engaging in the protocol. This technique roughly corresponds to the closed room meeting example mentioned earlier. A multi-party key agreement protocol is sometimes called a **group key agreement** protocol.

Finally, key establishment protocols can be classified as key pre-distribution or session key establishment schemes. In a **key pre-distribution** scheme, the secret key produced

by a fixed group of users is deterministic. In a **session key establishment** scheme, the key varies for each execution of the protocol.

Different varieties of key establishment schemes exist because higher-level applications have different underlying requirements. For example, deterministic key pre-distribution schemes may be desirable in a wireless ad-hoc network environment [37]. Key transport schemes are desirable in situations where a trusted server demands extensive controls over its users [42], for example if the users (such as smart cards) are deemed incapable of generating good random numbers. Group key agreement schemes are essential for video streaming and collaborative applications [16]. Asymmetric two-party session key agreement protocols, however, are by far the most commonly used schemes.

For many key establishment protocols, a trusted party is often required to perform such tasks as authentication, key distribution and management, or issuing of certificates. While a relevant part of the key establishment process, we focus our attention in this thesis to the protocols themselves.

USES OF SESSION KEYS. The secret key K that becomes available at the end of a key establishment process is known as the **session key**. Ideally, this key should only be used during the period for which the parties are actively communicating, after which it should be entirely erased from memory. In practice this limits the amount of ciphertext available for cryptanalysis and reduces the damage done in the case of session key exposure.

A variety of cryptographic applications may need to use the session key to provide different services. Since it is considered good practice to use independent keys to perform independent tasks, each application i normally applies a **key derivation function** [39, Chapter 12] $\mathrm{kdf}_K(i)$ to produce a modified, independent key. Even if only one application is present, K may be an element of an algebraic structure, and must be represented as a bit string in practical implementations. The key derivation function, which behaves like a secure pseudorandom function, can destroy the mathematical structure and produce the desired bit strings. Examples of key derivation functions include MGF1 [31] and HMAC [35].

2.2 Examples of Protocols

In this section, we briefly present a few different types of key establishment protocols, so as to give a flavor of what they look like. No security analysis will be given.

2.2.1 Key transport using symmetric cryptography

Suppose A wants to securely transport a session key K to B. A *key distribution center* (KDC) is often used in key transport protocols to facilitate long-term key pre-distribution and session key creation. In the Needham-Schröder protocol, the KDC T distributes *a priori* symmetric keys K_{AT} and K_{BT} to A and B respectively. The protocol messages are as follows. Here, $E_K(m)$ denotes the symmetric encryption of a message m using key K, and N_A denotes a nonce (a non-repeating value) generated by A.

1. $A \rightarrow T : A, B, N_A$
2. $A \leftarrow T : E_{K_{AT}}(N_A, B, K, E_{K_{BT}}(K, A))$
   (Note: K is the session key freshly created by T)
3. $A \rightarrow B : E_{K_{BT}}(K, A)$
4. $A \leftarrow B : E_K(N_B)$
5. $A \rightarrow B : E_K(N_B - 1)$

We note that the Needham-Schröder protocol is insecure as it is susceptible to the Denning-Sacco attack [20].

2.2.2 Key transport using public key cryptography

Suppose A wants to securely transport a session key K to B. A first prepares a timestamp t_A and then uses one of the following three protocols. Here, $E_B(m)$ denotes the encryption of a message m with B's public key, and $\text{sig}_A(m)$ denotes A's public-key signature on message m.

1. (Sign-then-encrypt) $A \rightarrow B : E_B(K, t_A, \text{sig}_A(B, K, t_A))$
2. (Sign and encrypt) $A \rightarrow B : E_B(K, t_A), \text{sig}_A(B, K, t_A))$
3. (Encrypt-then-sign) $A \rightarrow B : t_A, E_B(A, K), \text{sig}_A(B, t_A, E_B(A, K))$

2.2.3 Key agreement using symmetric cryptography

Suppose A and B want to engage in a secure key agreement process. Let K_{AB} be a long-term secret key shared between A and B. In practice, K_{AB} is obtained by a key pre-distribution scheme, such as Blom's key pre-distribution scheme. Boyd's two-pass protocol [11] is an example of a symmetric key agreement scheme. Here, $\text{MAC}(K; m)$ denotes the tag on message m produced by a message authentication algorithm using key K.

1. $A \rightarrow B : N_A$
2. $A \leftarrow B : N_B$

The resulting session key is $K = \text{MAC}(K_{AB}; N_A || N_B)$.

2.2.4 Key agreement using public key cryptography

The Diffie-Hellman protocol [21] was the first key agreement protocol using public key cryptography. Here, $kdf_K(m)$ denotes a pseudorandom function, and g is a generator of a cyclic group of order n.

1. $A \rightarrow B : g^x$
2. $A \leftarrow B : g^y$

The resulting session key is $K = \text{kdf}(g^{xy}, 0)$. The value g^{xy} is sometimes called the Diffie-Hellman value.

The Diffie-Hellman protocol is secure against passive adversaries, provided that the Computational Diffie-Hellman problem (CDHP) is intractable. However, it succumbs to active attacks, for example by an adversary who impersonates B.

In an attempt to resist active attacks, one may wish to add basic authentication information to the protocol messages. We call the following protocol the basic authenticated Diffie-Hellman protocol.

1. $A \rightarrow B : A, g^x$
2. $A \leftarrow B : B, g^y, \mathrm{sig}_B(g^x, g^y)$
3. $A \rightarrow B : \mathrm{sig}_A(g^y, g^x)$

The resulting session key is $K = \mathrm{kdf}(g^{xy}, 0)$. Assuming that signatures are unforgeable, this protocol seems to resist the obvious impersonation attacks by an active adversary. However, as we shall see in Section 2.3.4, it succumbs to other types of active attacks.

The station-to-station (STS) protocol is one variant of the Diffie-Hellman protocol that resists some active attacks. The following is the STS-ENC variant of STS, so named because it uses a symmetric-key encryption scheme E to achieve key confirmation.

1. $A \rightarrow B : g^x$
2. $A \leftarrow B : B, g^y, E_{g^{xy}}(\mathrm{sig}_B(g^y, g^x))$
3. $A \rightarrow B : A, E_{g^{xy}}(\mathrm{sig}_A(g^x, g^y))$

The resulting session key is $K = \mathrm{kdf}(g^{xy}, 0)$.

The MQV protocol [36] is another example of a Diffie-Hellman key agreement scheme that appears to resist active attacks. We present a simplified version of the MQV scheme. Here, A's long-term public key is g^a, and B's long-term public key is g^b.

1. $A \rightarrow B : A, B, g^x$
2. $A \leftarrow B : B, A, g^y, \mathrm{MAC}(K_1;\ 2||\ B||\ A||\ g^y||\ g^x)$,
   where $K_1 = \mathrm{kdf}((g^x \cdot (g^a)^{g^x})^{s_B}, 0)$ and $s_B = y + bg^y \pmod{n}$.
3. $A \rightarrow B : A, B, \mathrm{MAC}(K_2;\ 3||\ A||\ B||\ g^x||\ g^y)$,
   where $K_2 = \mathrm{kdf}((g^y \cdot (g^b)^{g^y})^{s_A}, 0)$ and $s_A = x + ag^x \pmod{n}$.

Note that $K_1 = K_2 = \mathrm{kdf}(g^{s_A s_B}, 0)$. The resulting session key is $K = \mathrm{kdf}(g^{s_A s_B}, 1)$.

2.3 Desirable Attributes of Key Establishment Protocols

Due to their inherit nature, different types of key establishment protocols demand different sets of security goals and attributes. For example, it is important for both parties to contribute information in the derivation of the session key in a key agreement protocol, but not so in a key transport protocol. In this section, we look specifically at the sets of security goals and attributes demanded by two-party key establishment protocols.

All key establishment protocols must meet a number of fundamental security goals to be considered secure. It is non-trivial to define these requirements precisely, and we will devote the entire next chapter towards this objective. In this section, we will only give some informal (but accurate) descriptions.

In addition to security goals, however, there are a number of security attributes that key establishment protocols are desired to possess, regardless of the protocol details. We call these the **desirable attributes** of key establishment protocols. An **attack** occurs in the presence of an adversary when a protocol's security goals or desired security attributes are not met. Key establishment protocols have a large number of desirable properties, and this often explains why they are difficult to design and are prone to attacks. This section identifies some of these desirable attributes.

2.3.1 Fundamental attributes

Key freshness — A session key is fresh to a party A if A is assured that the key is a new value, rather than a reused value from a previous session. This attribute is sometimes called key independence.

Key control and dual contributions — Key control is provided to the participating parties A and B if the session key is derived as a function of information provided by both A and B, and neither party can control or influence the value of the session key. Key control is an important attribute for a key agreement protocol, as it ensures

fairness in the derivation of the key. However, key control cannot be achieved in a key transport protocol.

Implicit key authentication — Let A and B be honest parties, and suppose party A believes it's engaging in a key establishment protocol with party B. We say that implicit key authentication is provided to A if A is assured that no party other than B and itself (and possibly other trusted third parties) can possibly compute the session key when the protocol is completed. When implicit key authentication is provided to both participating parties in a key establishment protocol, the protocol is said to provide *mutual key authentication*. The phrase *unilateral key authentication* is used when one specifically speaks of authentication in only one direction. Note that the implicit key authentication property does not give assurances to A that B actually possesses the session key or that B has actually participated in the protocol.

Entity authentication — While implicit key authentication provides assurances that an identified party may gain access to the key, entity authentication gives assurances that an identified party is involved in a protocol (peer awareness), and is actually participating (peer aliveness) at a given instant. In the past, both key authentication and entity authentication were generically referred to as "authentication", although they clearly serve different purposes. Some researchers have argued that entity authentication is not relevant in the context of key establishment [7, 9], but we list it here since it may be desirable for certain specialized applications.

Key confirmation — While implicit key authentication provides assurances that an identified party *may* gain access to the key, key confirmation gives assurances that a second (possibly unidentified) party *actually* has possession of a particular session key. A protocol that provides both implicit key authentication and key confirmation is said to possess **explicit key authentication**. Key confirmation is meant to ensure that the parties are ready to participate in secure communications, but Shoup [45] argues that this may not be a useful property as possession of the session key does not imply *acceptance* of the key for subsequent uses.

Key integrity — Key integrity is provided if a key is assured to have not been modified or disturbed by an adversary. Note that a protocol that provides key control and key freshness may still fail to provide key integrity.

Resistance to known attacks — Since previously unknown attacks can be discovered in the future, it is impossible to list all possible attack strategies. However, it is desirable for a key establishment protocol to withstand the most typical attack strategies. These include man-in-the-middle attacks, reflection attacks and interleaving attacks.

Unknown key share resilience — First introduced by Diffie, van Oorschot and Wiener [22] and also called a key-identity misbinding attack [34], an unknown key share (UKS) attack occurs when A believes it is sharing a key with B, and although this is true, B mistakenly believes it is sharing the key with another (possibly dishonest) party $E \neq A$. The following example shows that such an attack can potentially be devastating. Suppose B is a bank and A is an account holder in the bank, and suppose that the protocol for electronic deposits to an account holder is simply the encryption of the amount using the session key obtained by a secure key establishment between A and B. If the KE protocol does not provide UKS resilience, then an active adversary E can induce B to deposit the funds to E's account instead of A's account.

Composability and secure channels — Arguably, one of the most important and central application for key establishment protocols is the establishment of *secure channels* through their composition with standard encryption and authentication primitives. Although the security objectives of secure channels differ from that of key establishment protocols, it is desirable that the key establishment process and the resulting session key are sufficient for guaranteeing the security of the secure channel.

2.3.2 Attributes concerning compromised keys

Three types of keys may be compromised in a key establishment protocol — ephemeral keys, session keys or long-term keys. **Ephemeral keys** are temporary secret values that

are needed during the execution of a key establishment protocol. Session keys are keys that are computed at the end of a key establishment protocol's execution. **Long-term keys** are values such as the private keys of the parties or the master key in a server. Compromises of long-term keys are the most detrimental, and that of ephemeral keys are the least detrimental. Here we list some security assurances that may be desired in the unfortunate event that these keys are compromised.

Ephemeral key security — Ephemeral key security is provided if compromise of ephemeral secrets does not reveal long-term secrets or session keys. In protocols where the long-term secrets are not only used for authentication, but also incorporated into the computation of the session key (such as the MQV protocol), it may be desirable that leakages of ephemeral secrets do not reveal anything about the long-term key. This property was first introduced by Law et. al. in [36]. While the significance of this property is not formally justified in [36], we believe that this property may be desirable in certain situations, for example when the set of possible ephemeral keys is small (say when computations are performed in a constrained environment), in which case the attacker is able to mount a brute-force dictionary search.

Known-key security — Known-key security is provided if compromise of past session keys does not help an adversary learn about any other session keys or impersonate a party in the future. Known-key security should be considered a fundamental security attribute because there is no guarantee that a session key will not be revealed through its subsequent use in a weak cryptographic scheme.

Forward secrecy — Forward secrecy is provided if compromise of the long-term keys of a set of parties does not compromise past session keys involving those parties. Forward secrecy is typically achieved for Diffie-Hellman-type protocols by the use of ephemeral keys. This property will be further studied in Section 4.2.1.

Key compromise impersonation resilience — Key compromise impersonation (KCI) resilience is provided if compromise of a party A's long-term keys does not allow an

adversary to impersonate other parties to A. This property was first introduced by Just and Vaudenay [30]. It will be studied further in Section 4.2.2.

2.3.3 Other desirable attributes

Identity protection — In an *authenticated* key establishment protocol, the identities of the parties in the protocol must be transmitted. If the protocol messages are not properly encrypted, it is possible for a passive attacker to learn the identities of the parties. Even if the messages are encrypted, it may still be possible for an active attacker to discover the identity of A by impersonating as B in the first few message flows. Of course, with this "identity probing" attack [18] the adversary is unable to complete the protocol since it is unable to forge B's identity. However, the adversary is successful in obtaining identity information about the party. As noted by Perlman and Kaufman [43], it is impossible to protect both parties against active attacks, as one party eventually needs to be authenticated. We say that a protocol offers *identity protection* if it does not reveal the identities of either party to a passive attacker, and does not reveal the identity of one party to an active attacker until the other party has been authenticated.

Non-repudiation — A key establishment protocol is said to provide non-repudiation if it prevents a party from denying having previously established a session with another party. For example, non-repudiation is provided in protocols where each party signs the peer's identity, since such a signature can be used as a proof of engagement to a third party. While many consider non-repudiation as a desirable property, others view that it may be undesirable in situations where privacy protection is a particular concern.

Message independence — A protocol is message independent if protocol messages can be sent and received in any order.

Computational efficiency — It may be desirable to have low computational complexity,

including the ability to perform precomputation to reduce online execution time. Such a property may not be important to both communicating parties: in a key establishment involving a low-powered wireless device and a server, computational efficiency is clearly of more concern to the wireless device.

Communications efficiency — It is desirable for a protocol to have a small number of message exchanges (called *passes*) between parties and a small total number of bits transmitted (bandwidth).

Resistance to side-channel attacks — Side-channel attacks "target sources of information that are inherit to a physical implementation" [41], such as sound, time, computational faults or power consumption. Although mathematical models in cryptography often do not account for side-channel attacks, it is highly desirable for protocol implementations to defend against these attacks.

Availability — It is highly desirable for protocol implementations to be robust against denial of service attacks. **Denial of service** (DoS) attacks occur when an adversary prevents legitimate parties from interacting with the protocol, often by exhausting all available computational resources or network connections of a party. Many key agreement protocols are vulnerable to DoS attacks because both participants are usually required to perform computations and store state information before they complete. Although in principle it is impossible to prevent DoS attacks against KE protocols, several measures (such as client puzzles [29]) can be taken to reduce the successes of such attacks.

2.3.4 The importance of listing desirable properties

It is nearly impossible for a single protocol to possess all desirable properties, and protocols with different sets of desirable properties have been proposed to accommodate particular scenarios. However, it is important for protocol designers to identify at an early stage the desirable properties that a protocol offers. Without doing so, attacks can be discovered

long after a protocol is proposed, only to have the original protocol designer claim that they do not constitute valid attacks, as the protocol did not intend to provide assurances against the desirable properties being exploited.

To illustrate, let us heuristically consider whether some of the desirable properties are offered by the basic authenticated Diffie-Hellman protocol (see Section 2.2.4).

The protocol provides key freshness, since the x and y in the Diffie-Hellman value g^{xy} should be randomly chosen each time by A and B respectively.

The protocol does not provide key control, since B does not have to select the value y until after it has received g^x from A. Indeed B can force some bits of g^{xy} to have a pre-assigned value by performing a dictionary search on the different choices of y. However, the lack of key control does not seem to produce significant damages to the protocol.

The protocol seems to provide mutual implicit key authentication. More precisely, we argue that the protocol provides secrecy for two honest parties and protection from impersonation. First suppose A sent the value g^x to B in the first message flow, and A received the value g^y, from the second message flow, that was indeed sent by B. Of course, an adversary can obtain g^x and g^y easily. However, assuming the intractability of the Computational Diffie-Hellman problem, the Diffie-Hellman value g^{xy} can only be computed by A (by computing $(g^y)^x$) and B (by computing $(g^x)^y$). Suppose now that an adversary E attempts to impersonate B. E's task is pick a value y and another value s so that s is a B's signature on (g^x, g^y). Assuming that the signature scheme is existentially unforgeable against chosen message attacks, E's task is infeasible.

Unfortunately, the protocol does not provide unknown key share resilience. This is demonstrated by the following attack.

1. $A \rightarrow B : A, g^x$; E intercepts this flow
2. $E \rightarrow B : E, g^x$
3. $E \leftarrow B : B, g^y, \mathrm{sig}_B(g^x, g^y)$
4. $A \leftarrow E : B, g^y, \mathrm{sig}_B(g^x, g^y)$

5. $A \rightarrow B : \text{sig}_A(g^y, g^x)$; E intercepts this flow

6. $E \rightarrow B : \text{sig}_E(g^y, g^x)$

At the end of the protocol run, A and B share g^{xy} but A believes that its peer is B, while B believes that its peer is E. As have been discussed in Section 2.3.1, the consequences of a successful unknown key share attack can potentially be devastating, so the protocol clearly cannot be called "secure".

Before unknown key share attacks were discovered in 1992 [22], the basic authenticated Diffie-Hellman protocol could easily have been judged as being "secure", when in fact it clearly is not. One lesson that can be learned here is that it is important to consider all desirable properties and determine how significant they are when evaluating a protocol under its proposed operating environment. Another lesson is that the term "security" carries no well-defined meaning when we evaluate protocols using the heuristic approach, since this approach can never account for all possible attack scenarios. Starting from the next chapter, we will examine the provable security approach for analyzing key establishment protocols.

Chapter 3

The Canetti-Krawczyk Security Model for Key Establishment Protocols

In the previous chapter, we saw that key establishment (KE) protocols are used to provide shared secret keys between two or more parties, which can then be used by symmetric-key mechanisms to provide higher-level cryptographic services. Key agreement protocols, for example, are a kind of key establishment mechanism whereby the shared secret is derived by two or more parties using information associated with each of them. Certainly, one of the security requirements of key agreement protocols is that no party other than the intended parties can derive the key. However, as we have already seen in Chapter 2, this requirement alone does not guarantee the protocols to resist all attacks, and that a formal treatment of security is desirable.

In this chapter, we present a clear and formal security model for key establishment protocols proposed by Canetti and Krawczyk in [17]. The rest of the chapter is organized as follows. In Section 3.1, we briefly look at the history of modeling key establishment protocols. In Section 3.2 and 3.3, we state and discuss the communication model underlying the model, and list security goals and desirable properties that key establishment protocols

should possess. We then model the attacker in terms of its adversarial goals and attack capabilities in Section 3.4, and present an appropriate definition of security in Section 3.5. Finally, we discuss some approaches to proving the security of KE protocols in Section 3.6.

3.1 Related Work

Although key establishment protocols have existed for a long time, a complexity theoretic treatment for analyzing their security did not exist until 1993 when Bellare and Rogaway [6] formalized them in the context of symmetric key transport protocols. Prior to this work, attempts to analyze KE protocol security had mostly taken the heuristic analysis approach, either by identifying specific attacks on specific protocols (for example, see [8, 22]), or by using logic-based formal methods from the computer science community (for example, see [14]).

Bellare and Rogaway [6] defined security using the *indistinguishability* approach first introduced in the context of public key encryption by Goldwasser and Micali [24]. This approach has proven to be useful in practice, and numerous extensions have spawned from the original work. Namely, Bellare and Rogaway [7] considered the three party case, Shoup and Rubin [46] considered the smart card setting, and Blake-Wilson, Johnson and Menezes [9] considered the public key setting.

Yet another way of defining security evolves around the *emulatability* approach used in the study of multi-party computation [40]. Formalized using the notion of universal composability, Bellare, Canetti, and Krawczyk [4] first adopted the idea in the context of key establishment protocols, and subsequent works [17, 18] followed. While an important theoretical framework, it is difficult to apply the emulatability approach to the analysis of real-world protocols.

In this thesis, we will investigate the security model developed by Canetti and Krawczyk [17] in 2001, which we call the **CK01 model**. To date it is the most comprehensive and generic complexity theoretic-based security model for analyzing key establishment protocols. Although it is based on the indistinguishability approach, it also incorporates

ideas from the emulatability approach and includes a fix to a subtle flaw in the Bellare-Rogaway definition identified by Bellare, Petrank, Rackoff and Rogaway [5] in 1995. The remainder of this chapter will be devoted to discussing the details underlying this model. We will highlight some differences and establish some relationships between the Canetti-Krawczyk, Bellare-Rogaway, and the universal composability models in Chapter 4.

3.2 Communication Model: CK01 Model

We begin to establish our general framework by formalizing the model of communication. That is, we define and specify the semantics of key establishment protocols in the context of a communication network. Though elaborate and technical in nature, these formalisms are necessary as they will be used subsequently for defining security and analyzing protocol security.

KE Protocols and sessions. A **protocol** is a multi-party algorithm, defined by a sequence of steps specifying the actions required to achieve a specified objective. Here we investigate protocols in the context of modern communication networks, where all participants are interconnected and messages can be exchanged freely and concurrently. In this setting, protocols are message driven and asynchronous. By message driven, we mean that each party waits for the arrival of incoming messages from the network or action requests from other programs within the party, and generates outgoing messages to the network and action requests to other programs within the party. By asynchronous we mean that we do not place any restrictions on the timings at which messages and action requests arrive and are generated. A two-party authenticated **key establishment protocol** is a two-party protocol in a multi-party setting, in which the main objective is to establish a shared **session key** between them.

Like most authentication protocols, every key establishment protocol is required to specify an initialization function I whereby initial bootstrapping information is communicated in an authenticated and out-of-band manner. I takes as one of its inputs a **security**

parameter k and produces (as a function of k) quantities such as the parties' long-term public and private keys, shared master key, and so on, and/or interacts with a certificate authority. Some of I's outputs may be marked secret, which represent sensitive data that cannot be retrieved publicly. I must complete execution in some assumed secure manner before the KE protocol can start. Abstracting this phase out from the actual key establishment phase allows us to focus our analysis on the main aspects of the key establishment protocol, regardless of whether symmetric or asymmetric authentication is used.

A **session** is an invocation of a protocol by a higher level program executed inside a party. Through the use of unique session identifiers, it is possible to have more than one session running simultaneously inside a party. For a key establishment protocol, a session is invoked by that party on some inputs, maintains a local state within the session, and produces local outputs.

- The **inputs** to a KE session are of the form $(P_i, s, P_j, \mathsf{initiator})$ or $(P_i, s, P_j, \mathsf{responder})$, and are given as an action request by a program within party P_i. Here s is a unique session identifier, P_j is the identity of the intended peer of session s for which the key establishment is to be taken place, and $\{\mathsf{initiator}, \mathsf{responder}\}$ indicates the role of P_i. We assume that the calling program of P_i ensures that all session identifiers among the party participants are unique.

- The **local state** is comprised of a party's local working space within the session.

- At the end of its execution, the calling party outputs (P_i, s, P_j, κ), where P_i, P_j and s are as before, and κ is either null or represents the value of the session key. If κ is not null, it is also marked secret. The party then updates the **local output** by appending the current session output to its existing outputs. Local outputs can be thought of as quantities returned to the calling program. Finally, the party *erases* the information in the local state.

Generation of session id's. In practice, the determination of a unique session identifier between two parties typically involves exchanging further messages before or during the

key establishment execution. One common technique involves having party A (and party B) generate a unique value s_A (s_B respectively) among all the s_A's (s_B's respectively) that it has previously chosen. The values s_A and s_B are then exchanged between the parties, and the session identifier is defined as $s = (s_A, s_B)$.

THE PRE-SPECIFIED PEER (CK01) MODEL. In the specification of the input action request, we implicitly assume that the identity of the intended peer P_j is known at the beginning of the protocol interaction. We may then **identify sessions** within party P_i by (P_i, s, P_j), where P_j is the intended partner and s is the session identifier. This model, which we call the **pre-specified peer** model, is often an oversimplification in practice, since the identity of the intended peer may not be known in advance (as in the case of the IKE protocol [18]) or the parties may want to remain anonymous. Other models, such as the post-specified peer (CK02) model [18] or the anonymous [45] model, have been developed to model those cases. We will examine the post-specified peer model in Chapter 4. However, we will only focus on the pre-specified peer model in this chapter.

Once a KE session has started, it can be in exactly one of four states at a given time: incomplete, completed, aborted, or expired.

- A session is called **incomplete** if it has not yet produced any outputs.
- It is called **completed** once it returns an output with a non-null κ value; otherwise it is called **aborted**.
- A completed session (P_i, s, P_j) is called **expired** if a session expiration action request has been issued to P_i. Session expiration will be discussed in the next section.

Lastly, recall that in the CK01 model, session activations are *locally instantiated* runs of the protocol, and so it is useful to identify when two sessions are engaging in the generation of a shared session key. We call two sessions matching if they bind to the same session identifer.

Definition 3.2.1 (matching sessions in the CK01 model) *Two sessions* (P_i, s, P_j) *and* (P_j, s', P_i) *are called* **matching sessions** *if* $s = s'$*, regardless of the sessions' current states.*

3.3 Attack Capabilities

So far, we have described the network setting and the behaviour of the protocol participants. To summarize, we are considering an open communication network where parties attempt to establish session keys with each other using a message-based, asynchronous KE protocol. The parties hold both public and secret information.

If we have a closed network and every participant is honest, then there is no need to consider security. However, since we are working in an open network, parties in the network are untrusted and can act maliciously. We call a party whose goals deviate from that of the intended protocol goals an **adversary** or an attacker. Consequently, one way to define the security of a system is to ensure that an active adversary, even given a set of capabilities, cannot accomplish its intended malicious goals. This section gives a model of the adversarial capabilities.

Historically, the adversaries have been modeled in terms of an exhaustive list of attacks that they can mount. This approach is undesirable because new attacks may come up in the future and existing attacks cannot be described generically. What we need is an abstraction of the attack capabilities available in our setting of open network and participants. Specific attacks, such as meet-in-the-middle, interleaving, and replay attacks can then be modeled as sequences of actions that the adversary is able to perform. Since we want to capture all possible attacks in this framework, we aim to model as powerful an adversary as possible without being overly unrealistic (in which case a protocol that withstands such adversaries may not exist).

We now present the attack capabilities under two adversarial models: the unauthenticated-links model and the authenticated-links model.

THE UNAUTHENTICATED-LINKS MODEL. In the **unauthenticated-links model** (UM),

the communication links between all parties are controlled by an active adversary $\mathcal{U}$ with the following capabilities:

- listen to all message traffic,
- inject, drop, and modify messages,
- control the scheduling of message delivery.

Without loss of generality, we assume that each participant is only connected to the adversary, rather than having all participants being interconnected. Thus, for example, a message sent from P_i to P_j must travel through $\mathcal{U}$. Protocol executions in the UM can now be modeled as a sequence of **activations** within any party P_i, controlled and scheduled by $\mathcal{U}$ in any order. An activation can take two forms: an action request within a party P_i, or an incoming message within P_i with a specified sender P_j. A party that has been activated runs its required procedure (according to the protocol message) and returns to $\mathcal{U}$ the resulting action request and/or outgoing message and local outputs, except for those marked secret. Note that these capabilities are indeed realistic — for example, a router or a node in a wireless network can easily perform those actions.

The adversary can also obtain secret information about specific parties and their sessions within the network. We list the actions in order of increasing severity.

- **session-state reveal** — this query can be applied to any *incomplete* sessions. The result is that the adversary learns the local state of the session. However, revealing the session state does not necessarily imply learning the long-term key of the party to that session. This is consistent with the practice of using a separate security module to perform long-term key calculations .
- **session key query** — this query can only be applied to any *completed* sessions. The result is that the adversary learns the session key corresponding to that session.
- **party corruption** — this query can be applied at any time. Upon corrupting a party A, the adversary learns all state information associated with A and all secret

information about A, including its long-term secrets. In addition, from that point on, A is no longer activated (i.e. it no longer participates in the protocol and generates local outputs) and its actions are totally controlled by the adversary, who can now send any messages with A as the specified sender. We call a party **honest** if it is uncorrupted, and **dishonest** once it has been corrupted.

Note that the above actions are realistic, intended to capture information leakages via side-channel attacks [41], social engineering, unsafe storage of secrets, physical compromises, and other techniques of cryptanalysis.

Finally, the adversary is allowed to issue a **session expiration** action request to any completed session (P_i, s, P_j) within P_i, upon which the session's state becomes expired. The session expiration query has the effect of erasing the session key within P_i, and is designed to model the common practice of limiting the lifetime of session keys within a party. Thus, an adversary will not be able to issue session-key queries to any expired sessions. Also, even after a party P has been corrupted, the adversary cannot learn anything about the local outputs of P's expired sessions.

A session (P_i, s, P_j) is called **locally exposed within** P_i if either a session-state reveal or a session-key query has been performed to (P_i, s, P_j), or a party corruption has been issued to P_i before (P_i, s, P_j) is expired. A session is called **exposed** if either itself or its matching session is locally exposed. A session is called **unexposed** if both itself and its matching session is locally unexposed.

We summarize all allowable adversarial actions under different protocol states in Figure 3.1.

THE AUTHENTICATED-LINKS MODEL. In the **authenticated-links model** (AM), the adversary $\mathcal{A}$ must faithfully deliver messages belonging to unexposed sessions or originating from uncorrupted parties. Formally speaking, assume that all outgoing messages include the identity of the sender and its intended peer. Whenever an uncorrupted party P_i activates an outgoing message m to its intended peer P_j within an unexposed session, the

[1]This action is allowed, but the adversary will not the learn any session keys that have been expired.

Actions↓/States→	Incomplete	Aborted	Completed	Expired
session-state reveal	Allowed	Disallowed	Disallowed	Disallowed
session-key query	Disallowed	Disallowed	Allowed	Disallowed
party corruption	Allowed	Allowed	Allowed	Allowed[1]
session expiration	Disallowed	Disallowed	Allowed	N/A

Figure 3.1: Adversarial actions under different protocol states

message is added to a list M of globally undelivered messages. Whenever $\mathcal{A}$ activates a party P_j with an incoming message m, it must be the case that $m \in M$ and P_j is specified as the intended peer in m. The message m is then removed from M so that it will not be sent more than once. In effect, all messages originating from uncorrupted parties are *authenticated* since $\mathcal{A}$ cannot inject new messages, resend or modify message bodies, or change the origin or destination of messages. However, we stress that $\mathcal{A}$ is neither required to deliver all messages, nor is it required to maintain the order of message deliveries.

In addition, $\mathcal{A}$ may issue session-state reveal, session key, corruption, or session-expiry queries in the same manner as in the UM. Once a party P_i is corrupted, however, $\mathcal{A}$ can add to M any outgoing messages, as long as P_i as specified as the sender[2].

One may question how realistic the authenticated-links model is in capturing interactions in real-life networks. Indeed, the AM is only meant to be a theoretical device for protocol analysis, and its usefulness will be demonstrated in Section 3.6 and Chapter 5. We shall not discuss the AM further, except to stress again that it is not intended to model practical situations.

[2]The original description of the AM model [4] did not specify the consequences of session-state reveal, session key, or session-expiry queries, since those queries were not yet incorporated into the model at the time. Later [17] stated that the adversary may inject or modify messages belonging to exposed sessions, but with no clear justifications. We reason that such abilities would make the adversary too powerful in the AM (as in breaking most "secure" KE protocols in the AM), and therefore should not be incorporated into our description of the model.

3.4 Adversarial Goals

Having described the communication and attack models, we can begin to formulate an appropriate definition of security. In the CK01 model, *security* is defined as a measure of an adversary's ability to meet its adversarial goals under a certain attack model. A protocol that meets a "good" definition of security will ensure that an adversary equipped with strong attack capabilities cannot achieve even a weak adversarial goal. To that end, one must remember that "there is no ultimate security model" [34] — "weak" and "strong" are only relative terms, subject to critiques and evaluations.

In the case of symmetric encryption schemes, for example, one obvious possible adversarial goal is to obtain the secret key, after which the adversary is able to decrypt all ciphertext. A weaker goal may be to break the *semantic security* of the scheme [24]; that is, to extract useful information about the plaintext from the ciphertext. In terms of attack models, the attacker may only be given the capability to capture ciphertext, or it may be given some previously stolen plaintext/ciphertext pairs encrypted using the secret key, or indeed it may have selected the plaintext and then obtained the corresponding ciphertext. Thus, a good notion of security is that of semantic security against chosen plaintext attacks.

Formulating a reasonable notion of security for key establishment protocols is difficult because such protocols must achieve many goals in a single run, making them vulnerable to many possible points of attacks. We identify three primary requirements which are believed to be necessary for a key establishment to be considered secure.

1. **Secrecy of session key** — If two honest parties establish a shared session key, then the adversary should not be able to learn any information about the key, whether it is the entire session key, some partial bits of it, or any useful information about the key.

2. **(Implicit) key authentication** — Suppose A and B are honest parties. If A wants to establish a key with B, then A must be assured that no parties other than B and itself may possibly compute the session key; and vice versa.

3. **Peer consistency** — If an honest party A establishes a session key K and believes that its peer is another honest party B, then if B also establishes the key K, then it must believe that its peer is A and not a possibly dishonest party $C \neq A$.

An adversary is considered successful in achieving its adversarial goals if it breaks any one of the above protocol goals. For example, an adversary capable of extracting some non-trivial information about the key is considered successful in breaking the secrecy goal.

3.5 Definition of Security: SK1-Security

We now present the definition of security for the UM. Like most indistinguishability-based formulations, security (under the security parameter k) is defined in part by considering the outcome of an *experiment* (or a *game*) played by the **KE-adversary** $\mathcal{U}$ against a **simulator**. Here, a KE-adversary is no different from a normal UM adversary, except that it is allowed to issue a special query during the experiment, which we will describe next. The simulator encapsulates the protocol participants by communicating with the adversary on their behalf, and gives responses to $\mathcal{U}$'s special query.

To capture the secrecy requirement of key establishment protocols, $\mathcal{U}$ is allowed to issue a special query, called a **test-session** query, to any completed, unexpired and unexposed session during the experiment. This query is unusual in the sense that it does not correspond to any adversarial actions in the UM, but rather is an artifact of the experiment. To respond to the query, the simulator first picks a random bit $b \leftarrow_R \{0,1\}$. Let κ_0 be a key sampled randomly from $\mathcal{D}$, where $\mathcal{D}$ is the probability distribution of session keys generated by completed sessions, and let κ_1 be the session key corresponding to the session being tested. The simulator returns κ_b.

The experiment, $\mathbf{Exp}_{\mathcal{U}}(k)$, can now be presented in Figure 3.2. Note from Step 4 of Figure 3.2 that after receiving κ_b from the simulator, the adversary does not need to immediately respond with a guess b', but may perform more work to increase the chance of making the right guess. We are now ready to present the definition of security, called SK1-

Security[3], in the CK01 model. The definition of security for the AM is analogous, except that the experiment is played between a KE-adversary $\mathcal{A}$ in the AM and the simulator.

Definition 3.5.1 (SK1-Security) *A key establishment protocol* Π *is called* **SK1-secure** *in the CK01 model if the following conditions hold:*

P1. If two uncorrupted parties complete matching sessions, then they both output the same session key.

P2. For any KE-adversary $\mathcal{U}$*, the advantage that* $\mathcal{U}$ *wins the distinguishing game at the end of the experiment (see Figure 3.2) is negligible in the security parameter; i.e.,*

$$\Pr(\mathbf{Exp}_{\mathcal{U}}(k) = b) \leq \frac{1}{2} + n(k),$$

where $n(k)$ *is a negligible function in* k*, the security parameter.*

3.6 Proving Security

As will be discussed in Chapter 4, key establishment protocols proven SK1-secure in the UM guarantee certain important security attributes relevant to the real world. However, due to the strong attack capabilities given in the UM, applying Definition 3.5.1 directly in the UM requires careful considerations of the interactions between the honest parties and the adversary. Consequently, direct proofs of security in the UM tend to be technical and cumbersome.

On the other hand, key establishment protocols proven SK1-secure in the AM are not very useful in practice, for the model does not capture attackers realistically. We will illustrate, however, that it is possible to transform any SK1-secure protocol Π in the AM to another protocol Π' in the UM with the same functionality. This modular approach is extremely useful in the design and analysis of key establishment protocols, as

[3]In [17], the security definition is termed SK-Security. We use SK1-Security here in order to distinguish it from other security definitions to be introduced in Chapter 4.

Key establishment experiment performed by $\mathcal{U}$ in the UM.

1. Setting: Security parameter k. Parties $P_1, \cdots, P_n$ have been initialized.
2. While $\mathcal{U}$ is running, it may
   - activate an action request message within some party P_i,
   - activate an incoming message from a party P_i to another party P_j,
   - corrupt a party P_i,
   - issue a session-state reveal query to an incomplete session within some party P_i,
   - issue a session-key query to a completed but unexpired session within some party P_i,
   - issue a session-expiration query to a completed session within some party P_i.
3. At any time during its run, $\mathcal{U}$ may issue a **test-session** query to a completed, unexpired, and unexposed session. A value κ_b is returned.
4. $\mathcal{U}$ may continue to use the capabilities described in Step 2, except it is not allowed to expose the test-session or its matching session. It is, however, allowed to corrupt a partner to a test-session as soon as the session expires at that partner.
5. $\mathcal{U}$ returns a bit b' at the end of its run as its guess for b. The output of the experiment is b'.

Figure 3.2: The distinguishing game $\mathbf{Exp}_{\mathcal{U}}(k)$.

it allows cryptographers to focus their designs and proofs of security on the less complicated authenticated-links model. We now give a brief overview of the approach, which is based on the notion of emulatability.

Let $\text{UNAUTH}_{\Pi,\mathcal{U}}(k, X, R)$ denote the global output of running protocol Π in the UM under security parameter k, input X and random input R, in the presence of adversary $\mathcal{U}$. The global output consists of all cumulative local outputs (excluding secret outputs) of all parties, together with $\mathcal{U}$'s output.

Let $\text{UNAUTH}_{\Pi,\mathcal{U}}$ denote the distribution ensemble $\{\text{UNAUTH}_{\Pi,\mathcal{U}}(k, X, R)\}_{k\in\mathbb{N},X\in_R\{0,1\}^*,R\in_R\{0,1\}^*}$. Let $\text{AUTH}_{\Pi,\mathcal{A}}$ be analogously defined in the AM.

Definition 3.6.1 *Let* Π *and* Π' *be key establishment protocols. We say that* Π' **emulates** Π **in the UM** *if for every UM adversary* $\mathcal{U}$ *there exists an AM adversary* $\mathcal{A}$ *so that* $\text{UNAUTH}_{\Pi',\mathcal{U}}$ *and* $\text{AUTH}_{\Pi,\mathcal{A}}$ *are polynomially indistinguishable.*

Let us consider what steps are necessary to construct a protocol Π' which emulates another protocol Π in the UM. Intuitively, the construction should involve adding features that are not present in the AM. For instance, key authentication, message integrity and replay detection are some features that should be incorporated into Π'. On the other hand, Π' should not remove or modify any functionality that Π already has. Do there exist transformations that can be applied systematically in a provably secure way? And if so, how should the transformation be done?

The answer to the first question is an affirmative yes. Let a **compiler** be an algorithm which takes a protocol description as input, and produces another protocol description. An **authenticator** is a compiler C which takes as input a protocol Π, and produces another protocol $C(\Pi)$ such that $C(\Pi)$ emulates Π in the UM.

Certainly, if we apply an authenticator C to a protocol Π that is SK1-secure in the AM, then the global output of running Π against an AM adversary is polynomially indistinguishable from the global output of running $C(\Pi)$ against a UM adversary. However, the more relevant question, which is not directly implied by the definition of authenticators, is whether the global output of running Π against a KE-adversary in the AM is polynomially

indistinguishable from the global output of running $C(\Pi)$ against a KE-adversary in the UM. Canetti and Krawczyk [17] proved that this is indeed the case.

Theorem 3.6.2 *[17] Let Π be a key establishment protocol that is SK1-secure in the AM, and let C be any authenticator. Then $\Pi' = C(\Pi)$ is SK1-secure in the UM.*

Proof. We show that Π' satisfies both properties of Definition 3.5.1 in the UM.

Property 1: Notice that when considering Property 1, a KE-adversary is the same as a regular AM or UM adversary since it cannot play the distinguishing game. Thus, if Π' does not satisfy Property 1, then the global output of running Π' against a UM adversary is easily distinguishable from the global output of running Π against an AM adversary, contradicting the fact that C is an authenticator.

Property 2: Suppose to the contrary that Π' does not satisfy Property 2. That is, suppose that there exists a KE-adversary $\mathcal{U}$ in the UM who answers the test-session challenge in $\mathbf{Exp}_{\mathcal{U}}^{\Pi'}(k)$ with non-negligible advantage ϵ. We show how to use $\mathcal{U}$ to construct a KE-adversary $\mathcal{A}$ who answers the test-session challenge in $\mathbf{Exp}_{\mathcal{A}}^{\Pi}(k)$ with non-negligible advantage, contradicting the fact that Π is SK1-secure in the AM.

We construct $\mathcal{A}$ from $\mathcal{U}$ according to the following steps:

1. ($\mathcal{U} \rightsquigarrow \bar{\mathcal{U}}$) Given the KE-adversary $\mathcal{U}$ in the UM, we first construct a UM adversary $\bar{\mathcal{U}}$ in the UM. $\bar{\mathcal{U}}$ acts as $\mathcal{U}$'s simulator and responds to $\mathcal{U}$'s operations as follows:

   (a) Whenever $\mathcal{U}$ performs an instruction (except a test-session query), $\bar{\mathcal{U}}$ performs the same instruction.

   (b) Whenever $\mathcal{U}$ issues a test-session query on a completed session s, $\bar{\mathcal{U}}$ responds to $\mathcal{U}$'s request by first issuing a session-key query on s to get the key k. It then picks $b \leftarrow_R \{0, 1\}$, and returns k_b to $\mathcal{U}$, where $k_0 = k$ and k_1 is a random key from the probability distribution of session keys.

(c) When $\mathcal{U}$ halts (normally along with the test-challenge response b'), $\bar{\mathcal{U}}$ outputs its transcript of interactions with $\mathcal{U}$ and then also halts.

2. ($\bar{\mathcal{U}} \rightsquigarrow \bar{\mathcal{A}}$) Given a UM adversary $\bar{\mathcal{U}}$, it follows directly from the definition of authenticators that there exists an AM adversary $\bar{\mathcal{A}}$ against Π such that the transcripts outputted by $\bar{\mathcal{U}}$ and $\bar{\mathcal{A}}$ are indistinguishable (i.e. $\bar{\mathcal{A}}$ perfectly simulates $\bar{\mathcal{U}}$).

3. ($\bar{\mathcal{A}} \rightsquigarrow \mathcal{A}$) Given the AM adversary $\bar{\mathcal{A}}$, we construct a KE-adversary $\mathcal{A}$ in the AM. $\mathcal{A}$ acts as $\bar{\mathcal{A}}$'s simulator and responds to $\bar{\mathcal{A}}$'s operations as follows:

   (a) $\mathcal{A}$ first randomly picks a session $s \in \{s_1, s_2, \ldots, s_l\}$, where the s_i's are the sessions initiated by $\bar{\mathcal{A}}$ and l is the upper bound on the total number of sessions.

   (b) Whenever $\bar{\mathcal{A}}$ performs an instruction (except a session-key query on s), $\mathcal{A}$ performs the same instruction.

   (c) When session s is established, $\mathcal{A}$ issues a test-session query on s and stores the resulting test-session challenge k_b.

   (d) Whenever $\bar{\mathcal{A}}$ issues a session-key query on session s while it is still unexposed, $\mathcal{A}$ answers $\bar{\mathcal{A}}$'s query by returning k_b.

   (e) When $\bar{\mathcal{A}}$ halts, $\mathcal{A}$ examines whether the transcript of $\bar{\mathcal{A}}$ (which describes $\bar{\mathcal{U}}$'s interactions, which in turn describes $\mathcal{U}$'s interactions) describes a test-session challenge by $\mathcal{U}$ on session s. If so, $\mathcal{A}$ outputs the test-challenge response b' that $\mathcal{U}$ outputs. Otherwise, $\mathcal{A}$ outputs a random bit value.

Analysis: By assumption $\mathcal{U}$'s success probability in the UM is $\frac{1}{2} + \epsilon$. So we have

$$\frac{1}{2} + \frac{1}{2}\Big(\Pr[\mathcal{U} \text{ outputs } 1 \mid b = 1] - \Pr[\mathcal{U} \text{ outputs } 1 \mid b = 0]\Big) = \frac{1}{2} + \epsilon.$$

If we let

$$p_1 = \Pr[\mathcal{U} \text{ outputs } 1 \mid b = 1] \text{ and } p_0 = \Pr[\mathcal{U} \text{ outputs } 1 \mid b = 0],$$

then we can rewrite the previous equation as

$$\frac{p_1 - p_0}{2} = \epsilon. \tag{3.1}$$

Now let's consider $\mathcal{A}$'s success probability in the AM.

Case 1 — If $\bar{\mathcal{A}}$ didn't pick s as its test-session (which occurs with probability $(1 - \frac{1}{l})$), then $\mathcal{A}$ returns a random bit, according to 3(e). Therefore

$$\Pr[\mathcal{A} \text{ outputs } 1 \mid \text{ test-session} \neq s] = \frac{1}{2}. \tag{3.2}$$

Case 2 — If $\bar{\mathcal{A}}$ picked s as its test-session (which occurs with probability $\frac{1}{l}$), then $\mathcal{A}$'s response to the test-challenge challenge is the same as $\mathcal{U}$'s, according to 3(e).

1. If $\mathcal{A}$ is given the real key as the test-challenge, then $\bar{\mathcal{A}}$ must have received the real key in its session-key query. Since the outputs of $\bar{\mathcal{U}}$ and $\bar{\mathcal{A}}$ are indistinguishable, $\bar{\mathcal{U}}$ must also have received the real key in its session-key query. Therefore, $\mathcal{U}$ must have prepared $k_0 =$ real key and $k_1 =$ random key, and then returned k_b, where $b \leftarrow_R \{0, 1\}$. Hence

$$\begin{aligned}
& \Pr[\mathcal{A} \text{ outputs } 1 \mid \text{ test-session} = s, \text{ test-challenge is real}] \\
= \; & \Pr\left[\mathcal{U} \text{ outputs } 1 \,\middle|\, \begin{array}{c} \text{test-session} = s, \text{ test-challenge} = k_b, b \leftarrow_R \{0,1\}, \\ k_0 = \text{real}, k_1 = \text{random} \end{array}\right] \\
= \; & \Pr[\mathcal{U} \text{ outputs } 1] \\
= \; & \frac{1}{2}\Big(\Pr[\mathcal{U} \text{ outputs } 1 \mid b = 0] + \Pr[\mathcal{U} \text{ outputs } 1 \mid b = 1]\Big) \\
= \; & \frac{p_0 + p_1}{2}.
\end{aligned} \tag{3.3}$$

2. If $\mathcal{A}$ is given the real random key as the test-challenge, then $\bar{\mathcal{A}}$ must have received the random key in its session-key query. Since the outputs of $\bar{\mathcal{U}}$ and $\bar{\mathcal{A}}$ are indistinguishable, $\bar{\mathcal{U}}$ must also have received the random key in its session-key query. Therefore, $\mathcal{U}$ must have prepared $k_0 =$ random key and $k_1 =$ random

key. Thus

$$\begin{aligned}
& \Pr[\mathcal{A} \text{ outputs } 1 \mid \text{ test-session} = s, \text{ test-challenge is random}] \\
= \; & \Pr\left[\mathcal{U} \text{ outputs } 1 \,\middle|\, \begin{array}{c}\text{test-session} = s,\ \text{test-challenge} = k_b, b \leftarrow_R \{0,1\}, \\ k_0 = \text{random}, k_1 = \text{random}\end{array}\right] \\
= \; & \Pr[\mathcal{U} \text{ outputs } 1 \mid k_b \text{ is random}] \\
= \; & \Pr[\mathcal{U} \text{ outputs } 1 \mid b = 1] \\
= \; & p_1. \qquad (3.4)
\end{aligned}$$

Combining Cases 1 and 2, we get

$$\begin{aligned}
& \Pr[\mathcal{A} \text{ succeeds}] \\
= \; & \left(1 - \frac{1}{l}\right) \Pr[\mathcal{A} \text{ succeeds} \mid \text{ test session} \neq s] + \\
& \frac{1}{l} \Pr[\mathcal{A} \text{ succeeds} \mid \text{ test session} = s] \\
\overset{(3.2)}{=} \; & \left(1 - \frac{1}{l}\right) \frac{1}{2} + \frac{1}{l} \Pr[\mathcal{A} \text{ succeeds} \mid \text{ test session is } s] \\
\overset{(3.3),(3.4)}{=} \; & \left(1 - \frac{1}{l}\right) \frac{1}{2} + \frac{1}{l} \left(\frac{1}{2} + \frac{1}{2}(p_1 - \frac{p_0 + p_1}{2})\right) \\
= \; & \frac{1}{2} + \frac{1}{2l} \left(\frac{p_1 - p_0}{2}\right) \\
\overset{(3.1)}{=} \; & \frac{1}{2} + \frac{\epsilon}{2l},
\end{aligned}$$

which is also non-negligible in k. The result follows. □

To answer the second question, it suffices to show the existence of authenticators. We describe a signature-based authenticator $C_{\lambda_{sig}}$ proposed by Bellare, Canetti, and Krawczyk [4]. We assume that we are using a public key signature scheme that is existentially unforgeable against chosen-message attacks, and that all parties have already set up keys.

The authenticator $C_{\lambda_{sig}}$, on input Π, generates a protocol $\Pi' = C_{\lambda_{sig}}(\Pi)$ as follows: replace each message m that A sends to B in Π by the following message flows in Π':

1. $A \longrightarrow B : m$
2. $A \longleftarrow B : m, N_B$
3. $A \longrightarrow B : m, \mathrm{sig}_A(m, N_B, B)$

Here, N_B is a nonce generated by B, and $\mathrm{sig}_A(m)$ denotes A's signature on m. Upon B receiving the third flow, the protocol continues only if the signature can be successfully verified by B using A's verification algorithm (determined by A's public key).

Bellare, Canetti and Krawczyk proved that $C_{\lambda_{sig}}$ is an authenticator [4]. In addition, they constructed an encryption-based authenticator, $C_{\lambda_{ENC}}$, which uses a public key encryption scheme semantically secure against (non-adaptive) chosen ciphertext attacks and a message authentication scheme strongly unforgeable against adaptive chosen-message attacks.

Chapter 4

Critique of the Canetti-Krawczyk Model

In the previous chapter, we presented a formal security model, called the CK01 model, and a rigorous mathematical definition of security, called the SK1-Security definition. It is important to examine the security definition critically, as it is a central component in the design and analysis of key establishment protocols under the provable security paradigm. However, it is also imperative that one critically examines the model itself, as good definitions can only be constructed on top of a solid security model. This chapter provides a detailed critique of the SK1-Security definition and the underlying CK01 model.

In Section 4.1, we analyze whether the SK1-Security definition is sufficient for guaranteeing the protocol goals and desirable properties listed in Chapter 2. In Section 4.2, we consider possible modifications to the model to accommodate additional desirable properties. Section 4.3 compares the CK01 model with the post-specified peer (CK02) model, the Bellare-Rogaway model, and the Universal Composability (UC) model.

4.1 Sufficiency of the Model

4.1.1 Properties satisfied

The SK1-Security definition guarantees all security requirements listed in Section 3.4.

Proposition 4.1.1 *If* Π *is SK1-secure, then* Π *provides mutual implicit key authentication.*

PROOF. Let Π be SK1-secure, and suppose A has completed (A, s, B) with session key K. If (B, s, A) is also completed, then B can compute K by Property 1 of Definition 3.5.1. Suppose another honest party $C \neq A, B$ can also compute K. Then an adversary can issue a test-session query to (A, s, B), corrupt C to learn K, and win the test-session experiment. This contradicts Property 2 of Definition 3.5.1. Thus, no parties other than A and B can compute K. □

Proposition 4.1.2 *If* Π *is SK1-secure, then an adversary cannot obtain any useful information about session keys produced by completed and unexposed sessions, except with negligible probability.*

PROOF. This is precisely Property 2 of Definition 3.5.1. □

Proposition 4.1.3 *If* Π *is SK1-secure, then* Π *satisfies the peer consistency requirement.*

PROOF. Suppose the contrary. That is, suppose party A completes (A, s, B) with session key output K, while party B completes (B, s, C) with another party $C \neq A$ with the same session key output. Now, an adversary $\mathcal{U}$ can first issue a test-session query to (A, s, B). Since (A, s, B) and (B, s, C) are non-matching, $\mathcal{U}$ can perform a session-key query on (B, s, C) to learn K and use it to respond to the test challenge, thus defeating Property 2 of Definition 3.5.1. □

In addition, the definition guarantees a number of desirable properties.

Proposition 4.1.4 *If* Π *is SK1-secure, then* Π *guarantees key freshness*

PROOF. If Π doesn't produce fresh session keys, then an adversary can win the test-session experiment with non-negligible advantage after it exposes a previous session key, thus contradicting the Property 2 of Definition 3.5.1. □

Proposition 4.1.5 *If* Π *is SK1-secure, then* Π *is resistant to known key attacks.*

PROOF. By Property 2 of Definition 3.5.1, an adversary should not succeed in the distinguishability test for a completed, unexposed, and unexpired test-session, even though it may have exposed and learned the session keys of previous non-matching sessions. Thus, known key security is provided. □

Proposition 4.1.6 *If* Π *is SK1-secure, then* Π *possesses forward secrecy.*

PROOF. Suppose Π doesn't possess forward secrecy. That is, after an adversary $\mathcal{U}$ corrupts a party A, $\mathcal{U}$ is able to learn, with some non-negligible success probability, information about the session key of at least one of A's previously unexposed sessions (say (A, s, B)) that was *expired* before the corruption query is issued. $\mathcal{U}$ can then perform the following:

1. Issue a test-session query to a completed, unexposed and not yet expired session (A, s, B).
2. Issue an expire-session query to (A, s, B).
3. Once expired, corrupt A (this action is allowed in our adversarial model; cf. Section 3.3), and thereby obtain information about (A, s, B)'s session key.

$\mathcal{U}$ thus wins the test-session experiment, contradicting the definition of SK1-Security. □

Proposition 4.1.7 *If* Π *is SK1-secure, then* Π *is resistant to all possible attack scenarios in the UM.*

PROOF. This is simply a consequence of SK1-Security — by definition security is guaranteed in the face of a UM adversary, who is allowed to perform all possible attack scenarios. □

The next two subsections discuss two other desirable properties in greater detail.

4.1.2 Unknown key share resilience

Recall that an unknown key share (UKS) attack occurs when an honest party A believes it is sharing a key with another honest party B, while B mistakenly believes that it is sharing the key with another possibly dishonest party $E \neq A$. One may be tempted to conclude that UKS attacks cannot occur in a protocol that provides mutual key authentication. However, the provision of implicit key authentication is only considered in the case when an honest party engages the protocol with another honest party, in which case unknown key share resilience indeed holds. To see that, suppose to the contrary that A believes it's sharing the session key with B, whereas B believes it's sharing the key with C. By the property of mutual key authentication, A is assured that no parties other than B and itself can possibly compute the key. Now, if B is assured that only C and itself can possibly compute the same key, then this contradicts the fact that A can also compute B's key.

In the case when E is a dishonest party, the previous argument can no longer be used, and we must look at UKS resilience more carefully. As it turns out, the notion of peer consistency, which we listed as a core requirement for KE protocols, is equivalent to that of UKS resilience. This observation leads to two consequences. First, we have the following result:

Proposition 4.1.8 *If* Π *is SK1-secure, then* Π *is resilient to unknown key share (UKS) attacks.*

PROOF. By Proposition 4.1.3 it suffices to show that peer consistency implies unknown key share resilience. This holds trivially since if two honest parties A and B are peer consistent, then it is not possible for B to believe that it is sharing a key with another party $E \neq A$. □

Second, and perhaps more importantly, we now have a better understanding about the circumstances under which UKS attacks occur. UKS attacks are possible precisely when a KE protocol fails to provide an authenticated binding between the session key and the honest identities that generated the key, which leads Krawczyk [34] to use the term *key-identity misbinding attacks* to describe these attacks. Therefore, one situation under which UKS attacks are possible is when an attacker E can convince one of the honest parties A or B that it has knowledge of the session key, when it in fact cannot compute the key.

To illustrate, let us consider again the basic authenticated Diffie-Hellman protocol in the CK01 (pre-specified peer) model:

1. $A \longrightarrow B : A, g^x$
2. $A \longleftarrow B : B, g^y, sig_B(g^x, g^y)$
3. $A \longrightarrow B : sig_A(g^y, g^x)$

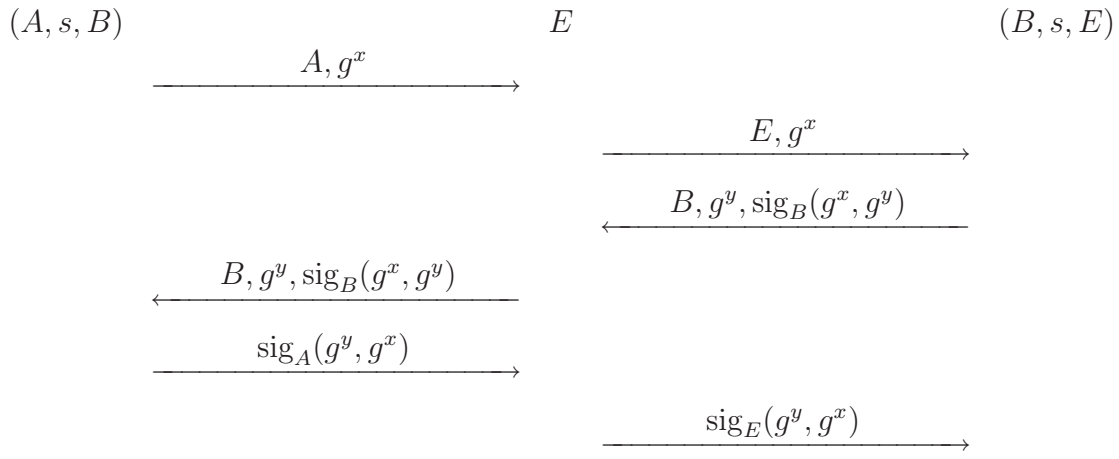

Figure 4.1: An unknown key share attack against the basic authenticated Diffie-Hellman protocol in the CK01 model

Suppose party A activates session (A, s, B) and party B activates session (B, s, E). In this protocol, the session key $K = g^{xy}$ is derived as a function of A and B's ephemeral

secrets (x and y respectively). Therefore, although the adversary E, who does not possess x and y, cannot possibly compute the session key, it is unwise to conclude that E cannot convince someone else that it has possession of the key. In fact, B is led to complete the key establishment with E once it received the last message flow

$$E \longrightarrow B : sig_E(g^y, g^x)$$

from E, which E can easily generate (without knowing x or y). The attack is depicted in Figure 4.1. From this example, we learn that a signature generated by the initiator on the Diffie-Hellman exponents is an insufficient key-identity binding.

As a further example, we show an unknown key share attack on STS-MAC that was first noticed by Blake-Wilson and Menezes [10].

The protocol messages for the STS-MAC protocol are as follows:

1. $A \rightarrow B : g^x$
2. $A \leftarrow B : B, g^y, \mathrm{sig}_B(g^y, g^x), \mathrm{MAC}_{g^{xy}}(\mathrm{sig}_B(g^y, g^x))$
3. $A \rightarrow B : A, \mathrm{sig}_A(g^x, g^y), \mathrm{MAC}_{g^{xy}}(\mathrm{sig}_A(g^x, g^y))$

Notice that the protocol messages in STS-MAC are very similar to those in the basic authenticated Diffie-Hellman protocol, except that each of the second and the last messages also contains a MAC on the signature computed under the Diffie-Hellman key g^{xy}. Thus, for example, after receiving the last protocol message and checking that $\mathrm{MAC}_{g^{xy}}(\mathrm{sig}_A(g^x, g^y))$ is valid against $\mathrm{sig}_A(g^x, g^y)$, B is assured that *someone* has knowledge of g^{xy}. After further verifying that the signature on A is valid, B is convinced that the identity of that someone is A.

Obviously, if an adversary E can compute g^{xy}, then the two-step verifications must succeed. However, if E cannot compute g^{xy}, it can still convince B that it has possession of the key and subsequently mount an unknown key share attack. All E has to do is to first intercept the last message flow in a protocol run between A and B, then select a public/private key pair satisfying

$$\mathrm{sig}_E(g^y, g^x) = \mathrm{sig}_A(g^y, g^x)$$

(the plausibility of selecting such key pairs efficiently for commonly used signature schemes is justified in [10]), register this key pair online, and finally send

$$E \rightarrow B : E, \mathrm{sig}_E(g^x, g^y), \mathrm{MAC}_{g^{xy}}(\mathrm{sig}_E(g^x, g^y))$$

to B. Since both the signature and the MAC verifications will succeed, B is convinced that it has completed a protocol run with E. Here, the key-identity misbinding occurs because the MAC only provides assurances that some party knows the key, but not the identity of that party.

It has taken many attempts to construct the "correct" key-identity bindings, as previous attempts to provide such bindings with the STS-ENC, STS-MAC and the Photuris protocols have all failed for different reasons (see [34]). Currently, we know of two techniques that seem to protect against UKS attacks. The first technique includes the intended recipient within each signature (i.e. computing $\mathrm{sig}_A(m, B)$ instead of $\mathrm{sig}_A(m)$) [47], and the second uses the SIGMA design approach [18]. These techniques will be further studied in Chapter 5, when we also illustrate protocols that can be proven SK1-secure.

4.1.3 Secure channels guarantee

A session-based network channels protocol is a communications protocol for data transmissions between two end-to-end parties, and such a protocol is called *secure* if it guarantees both the authenticity and secrecy of the data. A formal definition of this concept is developed by Canetti and Krawczyk [17]. As discussed in Chapter 1, one motivation for studying key establishment protocols is to determine the security features needed so that it can be appropriately composed with symmetric encryption, message authentication and other cryptographic functions to construct secure channels. Fortunately, as we will formally state in this section, SK1-Security indeed suffices to realize the construction of secure channels.

To appreciate the significance of this result, it is worth noting that key establishment protocols proven secure under the Bellare-Rogaway model do not guarantee secure channels in general. The reason for this will be given in Section 4.3.2.

Let Π be a key establishment protocol, MAC a message authentication function, E a symmetric encryption function and $\{\mathrm{kdf}_i\}$ a family of pseudorandom functions. Now define the functionalities of a network channels protocol $\mathsf{NetSec}(\Pi, \mathrm{MAC}, E, f)$ as follows:

1. $\mathsf{establish\text{-}sesion}(A, B, s, \mathsf{role})$: A initiates a KE-session Π with B under session-id s. When session (A, B, s) completes, A stores the session key output κ, and also computes and stores $\kappa_e = \mathrm{kdf}_\kappa(0)$ and $\kappa_a = \mathrm{kdf}_\kappa(1)$.

2. $\mathsf{expire\text{-}session}$: If the session (A, B, s) is completed, A expires the session and erases the corresponding session key.

3. $\mathsf{send}(A, B, s, m)$: If the session (A, B, s) is completed and unexpired, A checks that m is of the form $m = (mid, \bar{m})$, where mid is a unique message identifier and $\bar{m}$ is the actual message. If so, A produces $m' = (mid, c, t)$, where $c \leftarrow E_{\kappa_e}(\bar{m})$ and $t \leftarrow \mathrm{MAC}_{\kappa_a}(mid, c)$, and sends (A, s, m') to B.

4. Upon receiving a message (B, s, m'), A checks that (A, s, B) is a completed and unexpired sessions. If so, A checks that m' is of the form $m' = (mid', c', t')$, where mid' is unique. If so, A checks that $t' = \mathrm{MAC}_{\kappa_a}(mid', c')$. If so, A computes $\bar{m} \leftarrow E^{-1}_{\kappa_e}(c')$ and returns $(mid, \bar{m})$.

Theorem 4.1.9 *[17] If* Π *is SK1-secure,* MAC *is a message authentication code secure against chosen-message attacks, E is a symmetric encryption function secure against chosen-plaintext attacks, and* $\{\mathrm{kdf}_i\}$ *is a family of secure pseudorandom functions, then* $\mathsf{NetSec}(\Pi, \mathrm{MAC}, E, f)$ *is a secure channels protocol in the UM.*

4.1.4 Properties not guaranteed

The following desirable properties are not guaranteed by the SK1-Security definition.

Key control and dual contributions — The definition only stipulates the secrecy of the session key, and not how this key is to be derived. In fact, many key transport

protocols can be proven SK1-secure, but these protocols clearly do not possess dual contributions.

Entity authentication — Implicit key authentication, which the definition guarantees, is a strictly weaker condition than entity authentication. Entity authentication is not implied by the definition since it gives no assurances that a specified party is participating in the protocol at a given instant.

Key confirmation — Since key confirmation is not possible without simultaneous mutual entity authentication, the definition does not guarantee key confirmation.

Key integrity — An adversary may modify the protocol messages in such a way that both honest parties still complete with the same (modified) session key output, and for which the key still remains indistinguishable. Thus, key integrity is not provided.

Ephemeral key security — Once an adversary has issued a session-state reveal query to a session, it can no longer perform a test-session experiment on that session or its matching session. In other words, the SK1-Security definition does not capture the security of a session once it has been exposed via a session-state reveal query.

Key compromise impersonation resilience — In the CK01 model, a party can no longer interact with any other protocol participants once it has been corrupted. As such, one cannot make any security guarantees about future sessions associated with a corrupted party. Therefore, KCI resilience is not guaranteed (cf. Section 4.2.2).

Identity protection — Session activations in the CK01 model are pre-specified with the identities of the peers, and each message carries with it an associated session number. Therefore, an adversary can easily obtain the identities of the parties in a session.

Message independence — In general, the participating parties during a key establishment process need to maintain state information in order to verify information, complete sessions, derive session keys and so on. These state information cannot be maintained properly if messages are sent and received in an arbitrary order.

In addition, it is trivial to see that non-repudiation, computational and communications efficiency, and resistance to side channel and denial-of-service attacks are not guaranteed by the SK1-Security definition.

4.2 Modifications to the Model

4.2.1 Forward secrecy

While forward secrecy (FS) is regarded as an important desirable property that SK1-secure protocols enjoy, there are application settings in which the feature is not necessary. A typical example arises when one uses a one-pass key transport protocol to achieve short-term secrecy or authentication only, as in the case of a server-side secure socket layer (SSL) execution.

For instance, consider the ENC key transport protocol described in Figure 4.2. It is clear that knowledge of B's private key (which can be obtained via a party corruption) can be used to decrypt $E_B(r)$, and thus recover all session keys sent to B.

Due to the practical importance of these protocols, it is more reasonable to consider a modification of the model that does not account for forward secrecy, than to blindly consider all of them as "insecure". Disallowing the session expiration query, which is instrumental for proving forward secrecy, turns out to be all we need to solve the problem.

Definition 4.2.1 (SK1-Security without FS) *A key establishment protocol is called* **SK1-secure without FS** *if it is SK1-secure in the CK01 model, except that the key expiration query is removed from the list of adversarial actions.*

The ENC protocol can be proven SK1-secure without FS in the AM [17]. If one applies a public-key encryption-based authenticator to ENC, the resulting protocol is SK1-secure without FS in the UM and resembles the non-FS mode of the SKEME protocol [33].

We comment that Definition 4.2.1 is strictly weaker than Definition 3.5.1, since the adversary cannot corrupt a partner to the test-session during the test-session experiment.

That is, a key establishment protocol that is proven secure under Definition 3.5.1 is also secure under Definition 4.2.1, but the converse is not necessarily true.

- Setting: (G, E, D) is a public-key CCA-secure encryption scheme with security parameter k. $\{f_r\}$ is a family of pseudorandom functions.
- Protocol messages:
  1. On input (A, B, s), A picks $r \leftarrow_R \{0,1\}^k$ and sends $(A, s, E_B(r))$ to B. A completes and outputs the session key $K = f_r(A, B, s)$ under session s.
  2. Upon receiving (A, s, c), B computes $r' \leftarrow D_B(c)$. If the decryption process is successful, then B completes and outputs the session key $K' = f_{r'}(A, B, s)$ under session s.

Figure 4.2: The ENC key transport protocol in the AM

4.2.2 Key compromise impersonation resilience

Traditionally, the security of a cryptographic protocol is considered under the assumption that all participating parties protect their long-term secrets from exposures and leakages. That is, once an adversary has learned a party's long-term secret of a cryptographic protocol, that protocol is considered broken, and all security analysis becomes meaningless. Thus, it is important to protect these keys from unintentional leakages, side-channel attacks, or other techniques of cryptanalysis. However, in the unfortunate case that the long-term private key is lost, it may still be desirable for the protocol to carry some, albeit much weaker, security assurances. These assurances are desired because one ideally wants to minimize the amount of damage done to the protocol parties after the key compromise has happened and before it is detected. We consider the key compromise impersonation property, which limits the adversary's ability to impersonate once a long-term private key

is compromised.

Let us consider a two-party cryptographic protocol. Suppose an adversary has compromised a party A's long-term private key, but not B's. Obviously, this allows the adversary to impersonate A to B. If, in addition, the adversary is able to impersonate *to* A as B, then such a protocol is said to be vulnerable to key compromise impersonation (KCI) attacks.

KCI attacks do not make sense in the symmetric key model since an adversary who has compromised a long-term symmetric key between A and B can obviously impersonate A to B or B to A. However, in the public-key model, in which each party possesses its own public/private key pair, KCI attacks may be mounted on any of the parties of a cryptographic protocol.

Although KCI resilience can be considered for many different types of cryptographic protocols, it has only been studied elaborately with respect to key establishment protocols. Moreover, when the property is indeed considered, analysis has mostly been performed heuristically. These observations indicate that there had been a lack of significant evidence showing that KCI resilience is an important desirable property for real-world applications. Also, it may have been difficult to apply existing security models and formal analysis methods to the study of KCI attacks.

We first give supporting evidence demonstrating the importance of KCI resilience by presenting several real-world attack scenarios and their consequences. In the following scenarios we assume that a public key infrastructure is in place for all users.

Attack scenario — Secure channel password login. Let A be a terminal server accepting secure terminal session connections. When a user B wants to establish a secure terminal session with A, a key establishment protocol is first executed between A and B to establish a secure channel. Next, B is prompted for a username/password pair as a second layer authentication above the secure channel level. Upon checking the validity of the username/password pair, the user is granted access to the terminal session.

Suppose B's long-term private key has been compromised. One possible objective of an adversary $\mathcal{U}$ is to obtain B's username/password pair. If the KE protocol is vulnerable

to KCI attacks, then $\mathcal{U}$ can perform the following: it first impersonates A, and waits for B to initiate a connection. Once B has established a connection with $\mathcal{U}$, $\mathcal{U}$ prompts B for a username/password pair (encrypted in the secure channels level). Assuming that B is unaware of the attack and enters the username/password correctly the first time, $\mathcal{U}$ can immediately close the connection, then decrypt the username/password pair, thus achieving its goal. On the other hand, if the KE protocol is resistant to KCI attacks, then knowing B's long-term private key does not help $\mathcal{U}$ obtain any valid username/password pairs.

Attack scenario — Online shopping. Consider a small online bookstore, which operates an SSL-enabled web server A for secure online transactions and user authentication. The site owner understands the importance of online security, but also wants to maintain customer loyalty by keeping online profiles of existing customers. Each customer profile contains information such as email address, mailing address, credit card numbers and books bought.

Suppose A's long-term private key has been compromised. The adversary $\mathcal{U}$ may now replicate every aspects of the original site closely and request sensitive information from users who have entered the site (say by announcing to them that the site's database has been corrupted). Another goal for $\mathcal{U}$ is simply to obtain some of the existing customers' profiles already stored in A. It is not difficult to find out who are some of A's existing customers, since it is common for online bookstores to have an interactive online book review forum for existing customers, where the reviewers' usernames (and often also email addresses) are publicized.

Now suppose $\mathcal{U}$ wants to obtain a particular customer B's online profile already stored in A. If the KE protocol underlying SSL is vulnerable to KCI attacks, then $\mathcal{U}$ can first impersonate B and initiate an SSL session with A. Once the authentication is successful, $\mathcal{U}$ requests the "Retrieve profile information" web page. Assuming that A is unaware of the attack, it will render a page containing B's profile information and send it to $\mathcal{U}$, meaning that $\mathcal{U}$ will have achieved its goal. On the other hand, if the underlying KE protocol is

resistant to KCI attacks, then learning A's long-term private key does not imply that A will reveal user profiles stored in its database to $\mathcal{U}$.

Attack scenario — Peer-to-peer file systems. A peer-to-peer (P2P) network is a self organizing overlay network above the Internet, in which all participating machines (called *nodes*) have the same set of functionalities and responsibilities. In particular, each node acts both as a server and a client. P2P file systems are a popular application of P2P networks whereby the goal is to provide persistent storage of information in a highly available and secure manner. To protect the secrecy of data during a file transfer between a sender A and a receiver B, for example, a key establishment protocol first takes place to mutually authenticate A and B. A then checks whether B is granted read access to the file, and if so, sends the encrypted file to B using the key derived from the key establishment.

P2P nodes are often personal computers, which are more vulnerable to long-term private key compromises than corporate servers. Suppose A's long-term private key has been compromised. A contains a large repository of highly sensitive data, which it is only willing to give read access to another specific user B. An adversary $\mathcal{U}$'s goal is to obtain one or more of A's files. Obviously, if B's long-term private key, instead of A's, has been compromised, then the adversary can just impersonate B and achieve its goal. In our case, however, knowing only A's long-term private key does not help a passive adversary download the files. If the underlying KE protocol is vulnerable to KCI, however, then $\mathcal{U}$ can impersonate B and issue a command to request for A's files after completing a key establishment. Since A now (incorrectly) believes it is communicating with B, the read request is granted, and the file is sent to $\mathcal{U}$ in encrypted form. $\mathcal{U}$ decrypts the file using the key derived from the key establishment, thus achieving its goal.

Significance of KCI attacks. The above scenarios show that the KCI attacks are relevant in situations where the adversary wants to obtain additional secret information that A is normally only willing to share with B, and when it is easier to compromise A's long-term private key than B's. For example, in the first scenario, the username/password

pair represents information that the adversary wants to obtain. The KCI attack is relevant in this situation since it allows the adversary to succeed by impersonating B, when it is normally required to impersonate A.

Although one normally thinks of private key leakages as a result of cryptanalysis, they can also occur if the certificate authority (CA) or the device manufacturer (which pre-installs private keys to devices) is indeed dishonest. If both A and B are clients of a single fraudulent CA, then KCI attacks are not relevant because it is as easy to obtain A's long-term private key as it is to obtain B's. However, if A belongs to a dishonest CA and B belongs to an honest one, then the ability to mount KCI attacks allows the dishonest CA to obtain information that A is normally only willing to share with B.

KCI AND THE CK01 MODEL. The SIG-DH protocol, illustrated in Figure 5.2, seems to be resistant against KCI attacks. This is heuristically so because knowledge of A's private key does not seem to allow an adversary to sign B's messages, which are required in order to successfully impersonate B to A. The MQV protocol described in Section 2.2.4 also seems to be resistant to KCI attacks.

On the other hand, Boyd, Mao and Paterson [12] devised a Diffie-Hellman type key establishment protocol which is vulnerable to KCI attacks. The protocol messages, which look remarkably similar to the Unified Model [2], are as follows. Here, A's long-term public key is g^a, B's long-term public key is g^b, and H is a public hash function.

1. $A \rightarrow B : A, g^x$
2. $A \leftarrow B : B, g^y, H(g^{ab} || A || B || g^x || g^y)$
3. $A \rightarrow B : H(g^{ab} || A || B || g^y || g^x)$

The shared secret key is g^{xy}. Suppose an adversary has learned a, A's long-term private key. Then the adversary can impersonate B to A since a can be used to generate the response message (second message).

Unfortunately, protocols proven SK1-secure in the CK01 model do not necessarily imply resistance to KCI attacks. This is because private key compromises correspond to

corruption queries in the CK01 model, but the effect of corrupting a party A is that A may no longer participate in any further protocol interactions (see page 39). As a result, the CK01 model cannot capture scenarios where an adversary is trying to impersonate to a party whose private key is compromised. In fact, the protocol devised by Boyd, Mao and Paterson is proven SK1-secure, but is vulnerable to KCI attacks. We offer two ideas that may be used to formally consider KCI attacks in the CK01 model. First, suppose that a private-key query private-key(A) is added to the list of possible adversarial actions. Upon issuing such an operation, the adversary learns A's long-term private key (and nothing else), but A is allowed to continue to operate as before. If an adversary cannot win the test-session experiment on a session (A, B, s), where A has been issued a private-key query, then the protocol is resistant to KCI attacks. However, one is not sure what are the relationships between the SK1-Security definition and this new formulation. Another idea is to observe that KCI attacks are possible when a protocol is **deniable** [12]; that is, upon learning A's private key, an honest party B can simulate a protocol interaction with A without the actual presence of A. It may be possible to characterize KCI resilience using the concept of deniability. We consider this as part of our future work.

4.3 Comparisons with Other Models

4.3.1 The post-specified peer (CK02) model

This section introduces a modification of the original CK01 model called the **CK02 model** [18], also called the post-specified peer model. As described in Section 3.4, the CK01 formalism assumes that the identities (public key certificates) of the peers are known and pre-specified at the start of a key establishment session. Moreover, since action requests and incoming messages are activated and controlled by an adversary, the adversary is assumed to be able to obtain the identities of the intended recipients by examining the session numbers. We argue that these assumptions are not always realistic in practice.

As one example, suppose the identity of the peer is unavailable at the onset of the

protocol run. That is, suppose the initiator of a KE protocol is only given the address (for example, IP address) of the party to which it intends to establish a session, rather than the actual identity of the peer. This situation may occur in an ad-hoc or a wireless mobile network setting, where an address may be associated with many identities, since each party can reside at any particular address at any time. Clearly, since the peer identity needed to activate and identify sessions is missing, the CK01 model cannot be used.

As another example, suppose one or both parties in a protocol session do not wish to publicly reveal their identities on the network, but may instead wish to hide their identities against passive and active adversaries in order to protect their privacy. To make the example concrete, suppose the initiator of a KE protocol is a smart card, and the responder is a smart card reader. The owner of the smart card, worried about disclosing the identity of the card to an illegitimate card reader, may not wish to disclose the identity of the card to the reader before it authenticates and authorizes the reader. Recall that identity protection is not provided in the CK01 model, since session activations are pre-specified with the identities of the peers and each message carries with it an associated session number. An active attacker can simply examine the session number associated with each message and obtain the identity information.

Motivated by the inadequacy of the CK01 model to address security under the above scenarios, the CK02 model is developed to provide a more generic security model.

THE POST-SPECIFIED PEER (CK02) MODEL. In the pre-specified peer model, a party is only concerned with verifying the identity of the intended peer, and not discovering the identity during the key establishment execution. The post-specified peer model is a modification of the pre-specified peer model which also models protocols that must discover peer identity information during the protocol run. Luckily, only two changes are required: the semantics of session activation and an adaptation of the notion of matching sessions under the new semantics.

In the post-specified peer model, the activator does not know the identity of the peer with which it is communicating at the onset of the protocol execution. However, it should

know the "address" to which either the peer is associated or to which the message is to be delivered. In the case of the Internet, this address can take the form of an IP address. Thus, protocol activation is specified by the triplet (P, s, d), where P is the identity of the initiator, s is the session identifier, and d is the address of the intended peer. Note that the activation formalism proposed in this model generalizes that of the CK01 model, since d can be uniquely associated with the peer Q in the latter model.

Consequently, we **identify sessions** in the CK02 model by the pair (P, s), reflecting the fact that the identity of the peer may not be known yet. However, this information must be available by the time the session (P, s) completes, for otherwise key authentication cannot be achieved. Thus, the local session outputs consist of public output (P, s, Q), where Q is the identity of the peer to the session, and session key k, labeled secret. We write $peer(P, s) = Q$ and $sk(P, s) = k$. As in the case of the CK01 model, (P, s)'s local session state is erased when the session completes.

Next, we adapt the notion of matching sessions under the new session semantics. We use the following definition:

Definition 4.3.1 (matching sessions in the CK02 model) *Let (P, s) be a completed session in the CK02 model with public output (P, s, Q). We call (Q, s') the* **matching session** *of (P, s) if*

1. *$s = s'$, and either*
2. *(Q, s') is not completed; or (Q, s') is completed and produces public output (Q, s', P).*

Now we are ready to formulate a definition of security in the CK02 model, which we call SK2-Security. The definition of SK2-Security is identical to that of SK1-Security (Definition 3.5.1), except that the notion of matching sessions used is that of Definition 4.3.1.

Definition 4.3.2 (SK2-Security [18]) *A key establishment protocol Π is called* **SK2-secure** *in the CK02 model if the following conditions hold:*

P1. If two uncorrupted parties complete matching sessions, then they both output the same session key.

P2. For any KE-adversary $\mathcal{U}$, the advantage that $\mathcal{U}$ wins the distinguishing game at the end of the experiment is negligible in the security parameter; i.e.,

$$\Pr(\mathbf{Exp}_{\mathcal{U}}(k) = b) \leq \frac{1}{2} + n(k),$$

where $n(k)$ is a negligible function in k, the security parameter.

Krawczyk [34] proposed another formulation of KE security, in which Property 1 of Definition 4.3.2 is slightly strengthened.

Definition 4.3.3 (SK3-Security [34]) *A key establishment protocol Π is called* **SK3-secure** *in the CK02 model if the following holds. Suppose an uncorrupted party P completes session (P, s) with $peer(P, s) = Q$. Then:*

P1. If Q completes session (Q, s) while P and Q are uncorrupted, then

(a) $peer(Q, s) = P$; and

(b) $sk(Q, s) = sk(P, s)$.

P2. At the end of the experiment, the advantage that the KE-adversary $\mathcal{U}$ wins the distinguishing game when (P, s) is chosen as the test session is negligible in the security parameter; i.e.,

$$\Pr(\mathbf{Exp}_{\mathcal{U}}(k) = b) \leq \frac{1}{2} + n(k),$$

where $n(k)$ is a negligible function in k, the security parameter.

Critiques and comparisons. We first examine whether peer unavailability and identity protection are captured in the CK02 model. We then compare the SK2-Security and SK3-Security definitions in the CK02 model with the SK1-Security definition in the CK01 model.

In the CK02 model, session activations are specified by an address d instead of the peer's identity. Since knowledge of this address at the onset of the protocol run is insufficient to determine the identity residing at the address, the model is able to capture the unavailability of the peer identity.

Moreover, the identity of the peer cannot be determined from the session number until the key establishment process is completed. Thus, identity protection can be provided in the CK02 model, given that additional cryptographic mechanisms are used to protect the protocol messages from revealing identity information.

At first, it may seem trivial to include identity protection to messages in a key establishment protocol. However, practical experiences have shown that it is indeed very tricky to do so without affecting the security of the protocol. One evidence pointing to the difficulty is the conflicting nature of key authentication and identity protection. Here, we briefly describe a few possible solutions to the problem.

When identity protection against passive attackers is desired, one may simply apply symmetric encryption on the identities.

1. $A \longrightarrow B : g^x$
2. $A \longleftarrow B : B, g^y, E_{g^{xy}}(\mathrm{sig}_B(g^x, g^y))$
3. $A \longrightarrow B : A, E_{g^{xy}}(\mathrm{sig}_A(g^y, g^x))$

Figure 4.3: The STS-ENC key establishment protocol.

Example 4.3.4 *The STS-ENC protocol, shown in Figure 4.3, offers no identity protection against eavesdroppers, since the identities of the communicating parties A and B are transmitted in the clear. However, one may consider the following identity protecting variant of the STS-ENC protocol:*

1. $A \longrightarrow B : g^x$

2. $A \longleftarrow B : g^y, E_{g^{xy}}(B, \mathrm{sig}_B(g^x, g^y))$

3. $A \longrightarrow B : E_{g^{xy}}(A, \mathrm{sig}_B(g^y, g^x))$

Unfortunately, it is not possible for a KE protocol which protects both identities against an eavesdropper to also protect these identities against an active adversary. This is because one of the two participating parties must first prove its identity to its peer in a serially executing protocol. An active attacker can thus easily obtain this party's identity information by pretending to be the peer of the party. Thus, a KE protocol may at best protect only one party against an active adversary.

Example 4.3.5 *Consider the SIGMA-I protocol:*

1. $A \longrightarrow B : g^x$

2. $A \longleftarrow B : g^y, E_{k_2}(B, \mathrm{sig}_B(g^x, g^y), \mathrm{MAC}_{k_1}(B))$

3. $A \longrightarrow B : E_{k_2}(A, \mathrm{sig}_A(g^y, g^x), \mathrm{MAC}_{k_1}(A))$

Here, $k_1 = \mathrm{kdf}_{g^{xy}}(1)$ *and* $k_2 = \mathrm{kdf}_{g^{xy}}(2)$. *The shared session key is* $k = \mathrm{kdf}_{g^{xy}}(0)$. *This protocol protects the identity of both parties against eavesdroppers and identity of the initiator* A *against active attackers. The identity of the responder* B *can be easily obtained if an active attacker initiates the protocol and waits for the response from* B.

Next, we establish some relationships between the SK1, SK2, and SK3 security definitions. Propositions 4.3.6 and 4.3.7 show that KE protocols that are SK3-secure are also SK2-secure, but the converse is not necessarily true.

Proposition 4.3.6 (SK3 $\Rightarrow$ SK2) *If a KE protocol* Π *is SK3-secure, then it is SK2-secure.*

Proof. We need to show that Π satisfies the two conditions in the SK2-Security definition. *Property 1* (If two uncorrupted parties complete matching sessions then they both output the same key): Let P and Q be two uncorrupted parties that complete matching sessions.

These matching sessions must be of the form (P, s) and (Q, s), with $peer(P, s) = Q$ and $peer(Q, s) = P$. Thus, by the first condition for SK3-Security, we know that $sk(P, s) = sk(Q, s)$; that is, both parties output the same key.

Property 2 (The advantage of $\mathcal{U}$ in $\mathbf{Exp}_{\mathcal{U}}(k)$ is negligible): Suppose that $\mathcal{U}$ issues a test-session query to (P, s), where (P, s) is a completed, unexpired and unexposed session with $peer(P, s) = Q$. Since (P, s) is unexposed, both (P, s) and (Q, s) are locally unexposed. Thus, by the second condition of SK3-Security, $\mathcal{U}$ cannot distinguish $sk(P, s)$ from a random value. Thus the advantage of $\mathcal{U}$ is negligible. □

Proposition 4.3.7 (SK2 $\not\Rightarrow$ SK3) *A KE protocol* Π *that is SK2-secure is not necessarily SK3-secure.*

Proof. Consider the ENC key transport protocol described in Figure 4.2. The protocol can be shown to be SK2-secure in the AM [18]. However it is not SK3-secure in the AM, as the following attack scenario illustrates:

1. A sends $(A, s, E_B(r))$ to B. A completes with $peer(A, s) = B$ and $sk(A, s) = f_r(A, B, s)$ under session (A, s).

2. The adversary corrupts party C, and adds $(C, s, E_B(r))$ to the list of outgoing messages. Note that this is allowed in the AM since C is specified as the sender of this message.

3. Upon receiving (C, s, c), B computes $r \leftarrow D_B(c)$, checks that it is a valid plaintext, then completes with $peer(B, s) = C$ and $sk(B, s) = f_r(B, C, s)$ under session (Q, s).

Thus A and B are both uncorrupted, (A, s) completes with $peer(A, s) = B$, but (B, s) completes with $peer(B, s) \neq A$ and $sk(A, s) \neq sk(B, s)$, contradicting the first condition for SK3-Security. □

Note that the attack given in the previous proposition does not contradict the SK1 or SK2 security definition. For example, the first condition is not violated because the two

completed sessions (A, s) (with public output (A, s, B)) and (B, s) (with public output (B, s, C)) are non-matching. Since $\{f_r\}$ is assumed to be a family of secure pseudorandom functions, knowledge of $f_r(m_1)$ does not give any information about $f_r(m_2)$ for $m_1 \neq m_2$. Since $sk(A, s) \neq sk(B, s)$ in the above attack, the second condition is also not violated.

Next we investigate relationships between the SK1 and SK2 security definitions. Recall that the two definitions are stated in exactly the same way, except that their underlying security models (CK01 and CK02) are different.

Proposition 4.3.8 (SK2 $\not\Rightarrow$ SK1) *A KE protocol* Π *that is SK2-secure in the CK02 model is not necessarily SK1-secure in the CK01 model.*

Proof. Consider the basic SIGMA protocol (Σ_0) described in Figure 5.3 of Section 5.2. Σ_0 can be shown to be SK2-secure in the CK02 model (a sketch proof is given in Section 5.2.1). However, it is not SK1-secure in the CK01 model, as the following attack scenario in the UM illustrates:

1. The adversary $\mathcal{U}$ activates a new session (A, s, B) where A is the initiator. A sends the first flow

   $$A \rightarrow B : m_1 = (s, g^x)$$

   to B.

2. $\mathcal{U}$ corrupts a party E, so that E can no longer participate in the protocol.

3. Upon B receiving m_1, $\mathcal{U}$ activates a new session (B, s, E) where B is the responder. B responds by sending the second flow

   $$E \leftarrow B : m_2 = (s, g^y, \mathrm{sig}_B(g^y, g^x), \mathrm{MAC}_{k_1}(B))$$

   to E. Note that (B, s, E)'s session state now contains the value g^{xy}, since it is needed to calculate $k_1 = \mathrm{prf}_{g^{xy}}(1)$.

4. $\mathcal{U}$ activates session (A, s, B) to receive m_2 as the response to m_1. Upon receiving m_2, A completes (A, s, B) with a session key derived from g^{xy}.

$\mathcal{U}$ now issues a test-session query to (A, s, B). Since (B, s, E) is non-matching to (A, s, B), $\mathcal{U}$ can issue a state-reveal query to (B, s, E) to learn the value g^{xy} and successfully win the test-session experiment. □

Note that the above attack does not work in the CK02 model. This is because the session (A, s) is matching to (B, s) while it is still incomplete (see Definition 4.3.1), so the adversary is not allowed to issue a session-state reveal query to (B, s) to obtain information about the session key. On the other hand, the attack is possible in the CK01 model because (B, s, E)'s session state already contains the key when it is still incomplete, and the value of this key does not depend on the identities of the initiator or the responder.

We do not know whether all KE protocols that are SK1-secure are also SK2-secure or not. However, we have the following trivial and interesting result.

Proposition 4.3.9 ($\exists$ SK1 $\Rightarrow$ $\exists$ SK2) *Let Π be a SK1-secure KE protocol in the CK01 model. Then there exists a KE protocol Π' that is SK2-secure in the CK02 model.*

Proof. The construction of Π' is as follows.

1. When a session is activated with (A, s, d), Π' checks that d must specify an identity, and aborts otherwise.

2. Π' behaves exactly the same as Π.

3. Upon completing session s, Π' checks that the identity to be output is the same as the identity specified in d, and aborts otherwise.

Steps 1 and 3 of the construction essentially restrict Π' to only use features available in the CK01 model, even though it is running in the CK02 model. Thus, it follows trivially that the protocol is SK2-secure in the CK02 model. □

We believe that there exist SK1-secure KE protocols that are also SK2- and SK3-secure. In particular, we believe the SIG-DH protocol, to be discussed in Chapter 5, is one such protocol.

Conjecture 4.3.10 *The SIG-DH protocol is SK2- and SK3- secure in the CK02 model.*

However, there are two obstacles if one has to work out a full security proof for a KE protocol in the CK02 model. First, direct proofs in the UM are very difficult to construct and verify. Moreover, without an AM version of the CK02 model and a set of authenticators proven secure in this model, the authenticator proof technique may not be applied as easily as in the CK01 model. As was stated in [18], "While [the AM notion] could have been used in our analysis too, they would have required their re-formulation to adapt to the post-specified peer setting treated here."

One set of possible relationships between the SK1, SK2, and SK3 security definitions is captured in Figure 4.4. We finish this subsection with two open problems.

Open Problem 1 *Does there exist a KE protocol that is SK1-secure in the CK01 model, but not SK2-secure (therefore not SK3-secure) in the CK02 model?*

Open Problem 2 *Does there exist a KE protocol that is SK2-secure in the CK02 model, but not SK1-secure in the CK01 model and not SK3-secure in the CK02 model?*

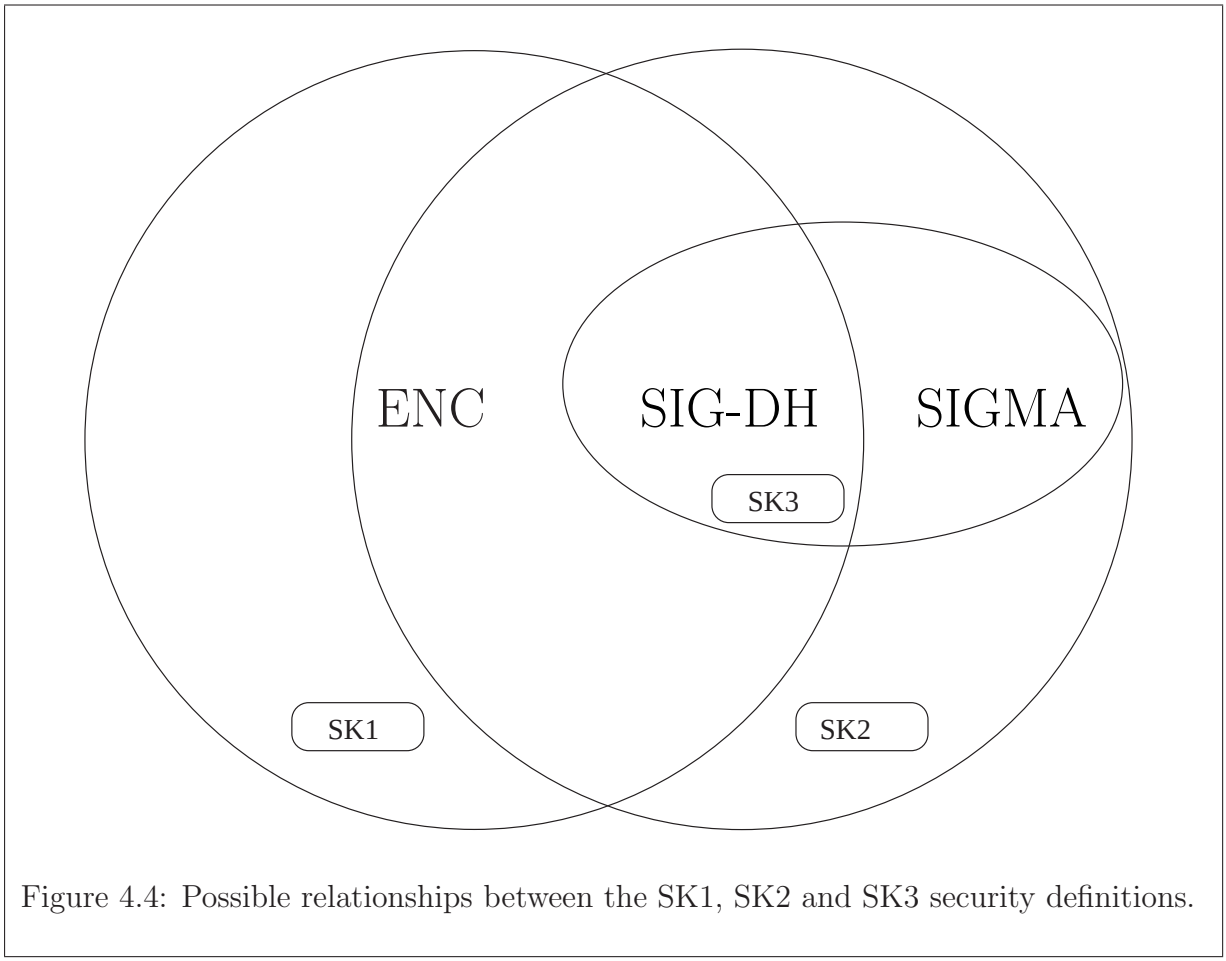

Figure 4.4: Possible relationships between the SK1, SK2 and SK3 security definitions.

4.3.2 The Bellare-Rogaway model

Bellare and Rogaway [6] developed the first security model and definition for KE protocols in 1993. Based on the indistinguishability approach, the original Bellare-Rogaway (BR) model only addressed security for two party KE protocols in the symmetric-key setting, but was subsequently extended by the same authors to model the three-party case [7], and by Blake-Wilson, Johnson and Menezes to model the public-key setting [9]. The BR model is very similar to the Canetti-Krawczyk model with respect to the communications model, adversarial goals, attack models, and definition of security. Indeed, the BR model differs from the CK01 model mostly in areas where the former model is deemed deficit or inadequate. In this subsection, we highlight the major differences between the two models, then critique and compare these differences.

The Bellare-Rogaway (BR) model. In terms of the communications model, recall that honest parties are modeled as separate entities in the CK01 model, each of which is responsible for maintaining sessions that it is involved with. The BR model, on the other hand, postulates an adversary in charge of a series of "oracles" $\Pi^s_{A,B}$, representing the s^{th} session invocation by a party A to establish a shared secret key with party B. Each oracle maintains its own state and can send messages to other oracles. Session identifiers are not presented in the BR model. Thus, it is important to note that the value s is not a session number, but a local counter necessary for distinguishing multiple KE executions between the same parties.

The attack model in the BR model is similar to that of the UM in the CK01 model. Adversarial actions, including message deliveries, session key reveals, and party corruptions, are modeled as oracle queries. However, session state reveal queries are not available in this model, and oracles never expire.

Armed with these formalisms, one can develop an analogous definition of security in the BR model, with two exceptions. First, an equivalent notion of matching sessions cannot be formulated, since the notion of session identifiers, which is essential for naming sessions, is missing in the model. Because of this problem, another partnering notion, called matching

conversations, is used instead. Two oracles are said to have **matching conversations** if in the presence of an adversary, the order and contents of the message flows between them are consistent with the message flows in the presence of a benign adversary. Second, a KE-adversary (cf. Figure 3.2) in the BR model is faced with a *non-adaptive* game; that is, once the adversary receives a challenge after issuing a test-session query to an oracle, it must respond to the challenge immediately (i.e. the game does not contain Step 4 of Figure 3.2).

Critiques and comparisons. As a first security model for KE protocols, the Bellare-Rogaway model has been successful in capturing a large number of desirable properties, as evidenced by the number of subsequent works that are based on it. However, as we will discuss, the model is not without areas of improvements, some of which are incorporated into the CK01 and CK02 models.

The use of non-adaptive experiments is considered one of the most significant deficiencies of the BR model. In an unpublished manuscript [5], Bellare, Petrank, Rackoff and Rogaway demonstrated a scenario showing that the model is insufficient to guarantee security when particular KE protocols are composed with certain higher level applications. They showed that the fix consists of merely allowing the KE-adversary to continue interacting after receiving the test challenge.

The adaptive experiment brings two immediate benefits. First, it ensures security when composed with secure channel protocols. In fact, Theorem 4.1.9 does not hold in the BR model. Also, together with session expiration queries, it allows forward secrecy to be properly captured.

One may wonder how such a simple modification can bring so many benefits. After all, it is not obvious at all what advantages an adversary can gain from an adaptive experiment. To appreciate the modification, it may be worthwhile to look at the difference between indistinguishability against non-adaptive chosen ciphertext attacks (IND-CCA1) and its adaptive variant (IND-CCA2) as security definitions for public-key encryption schemes.

Suppose that (E, D) is a public-key encryption scheme that is IND-CCA1-secure. We may wish to use the scheme in some higher-level applications, say an entity authentication

protocol which performs the following when a party A wants to authenticate another party B:

1. $A \rightarrow B : \cdots, E_B(r), \cdots$
2. $A \leftarrow B : \cdots, r, \cdots$.

Here, the intended party B can decrypt $E_B(r)$ and return r as proof of its identity, whereas a malicious party cannot do so. For an adversary who can send messages to B, however, this protocol effectively acts as a "decryption oracle" that can potentially be used to compromise the security of the encryption scheme. Now, since the scheme is CCA1-secure, an adversary who has access to this "decryption oracle" before receiving the target ciphertext, but is forbidden to use it thereafter, will not win the game with non-negligible advantage. However, is there any reason to believe that the "decryption oracle" in this scenario will stop operating after receiving the target ciphertext? Perhaps the adversary can make much better use of the decryption oracle after it knows exactly what the target ciphertext is. Thus, IND-CCA2, despite being potentially a much stronger security notion, is indeed more natural and covers more existing and unanticipated future attack scenarios.

The same reasoning can be applied to security formulations for key establishment protocols. Consider again the STS-ENC protocol in the post-specified peer model, shown in Figure 4.3. The protocol is not SK2-secure, although it is not clear whether it is BR-secure or not. It is vulnerable to different types of **protocol interference attacks** [45]; i.e. attacks that exploit weaknesses when the protocol is composed with certain higher-level applications. For example, upon receiving the second flow, A completes with $peer(A, s) = B$ and secret key K, but B hasn't completed as yet. A normally sends the third flow to B so that B can complete, but suppose that A invokes the secure channel application on top of the KE protocol before sending the third flow, a legitimate action. This application urgently starts sending secure messages to B of the form

$$A \rightarrow B : mid, E_K(m), \mathrm{MAC}_K(mid, E_K(m)).$$

If an attacker can somehow convince A to send the message $m = \mathrm{sig}_E(g^x, g^y)$ in the higher-level application, then it can replace the third flow with

3. $E \longrightarrow B : E, E_K(\text{sig}_E(g^y, g^x))$.

By the end of the second flow, session (A, s) has completed with $\text{peer}(A, s) = B$; and by the end of the third flow, session (B, s) has also completed, but with $\text{peer}(B, s) = E$. That is, an unknown key share attack has occurred.

As another example, upon receiving the second flow, an attacker can register a party E's public key to be the same as A's public key, then replaces the third flow with the message

3. $E \longrightarrow B : E, E_K(\text{sig}_E(g^y, g^x))$.

Now, B's verification of the signature using E's identity must succeed since A and E's public keys are the same. The result, again, is an unknown key share attack.

As expected, STS-ENC is not SK2-secure, as the following adaptive strategy suggests.

1. At the end of the second flow, (A, s) completes with $\text{peer}(A, s) = B$.
2. $\mathcal{U}$ picks (A, s) as the test-session, and is given the test challenge κ_b.
3. Instead of replying to the test-session immediately, $\mathcal{U}$ mounts the secure channels attack or the public key registration attack, and sends the third flow accordingly.
4. B completes session (B, s) with $\text{peer}(B, s) = E$.
5. $\mathcal{U}$ exposes (B, s) and obtains κ (this is possible since the session is non-matching to (A, s) according to Definition 4.3.2).
6. $\mathcal{U}$ can win the test-session experiment with overwhelming success probability.

Is the protocol also insecure in the BR model? In particular, what are the consequences if the adversary is faced with a non-adaptive experiment instead? For one thing, Step 3 can no longer be performed after Step 2. It is true that the adversary can perform Step 3 before Step 2 in order to avoid the adaptive stage, but such a proposition is very dangerous. Without first specifying in the test-session query the specific session (A, s) to attack, the

adversary may at best attack *all* secure channel sessions and mount public-key registration attacks on *all* sessions, which is clearly an arduous task. Thus, an adaptive experiment is absolutely essential in the security formulation.

There are a number of other differences between the CK models (CK01 and CK02) and the BR model:

Matching conversations — The BR model uses matching conversations to identify two oracles that are partners of each other, since session identifiers are not present in the model. This notion is obviously more complex and less intuitive than matching sessions for the purpose of identifying partners, with Rogaway acknowledging that it is "syntactic and fundamentally irrelevant" [44]. In the extreme case of one-pass KE protocols where both parties are activated as initiators, this definition of matching conversations actually fails to identify the parties as partners. However, since the notion of matching conversations takes real time communications into consideration, it enables the BR-Security definition to model key confirmation and explicit key authentication optionally.

Session state reveal queries — Session state reveal queries are not presented in the BR model, which implies that the model does not capture attack scenarios involving session state information.

Session expirations queries — Along with adaptive experiments, session expiration queries are instrumental for providing forward secrecy in the CK models.

Party corruptions — In the BR model, a corrupt query results in a "weak corruption", which reveals only the corrupted party's long term secret key. In the CK models, however, a corrupt query results in a "strong corruption", revealing the long term secret key and all session state information. Moreover, oracles in the BR model continue to operate after they have been corrupted, whereas honest parties in the CK models cease operating once corrupted.

AM adversary and protocol compilations — The lack of a modular compilation technique in the BR model makes security proofs much more difficult to understand and analyze.

While an important contribution, we hope that the reader is convinced that the BR model contains enough shortcomings that it should not be used as a primary security model for key establishment.

4.3.3 The Universal Composability model

In Section 4.1.3, we saw that a SK1-secure KE protocol can be composed with symmetric encryption and message authentication to obtain secure channels. This security guarantee is indeed an important one, since secure channels protocols are regarded as the most common applications which require secure key establishment. However, in general, one would desire to have the same security guarantee when a KE protocol is composed with any (reasonably secure) higher-level cryptographic protocols that require shared secret keys. In this subsection, we give a brief non-technical overview of the Universal Composability (UC) model [19], intended to capture security when general composability is desired.

The Universal Composability (UC) framework. In the Canetti-Krawczyk and Bellare-Rogaway models, security is defined by showing that an adversary's success probability in winning an experiment, which captures the adversary's ability to achieve its goals given a set of attack capabilities, is always negligible. This *reductionist* approach is intuitive and succinct, and has been used to define security for many other cryptographic schemes. There is a more general approach, called the *emulatability* approach, where the security of a cryptographic protocol is guaranteed if it can be shown to be "as secure as" an "ideal functionality". Here, an "ideal functionality" captures the desired behavior (i.e. both correctness and secrecy) of a cryptographic task in an assumed secure environment, and it is usually described as a trusted party (TP), who is responsible for relaying information between parties and computing output on behalf of the parties. In the case of

key establishment protocols, the "ideal functionality" should mimic the same behavior as a face-to-face establishment between two parties. A protocol Π is "as secure as" an ideal functionality $\mathcal{F}$ if for any adversary $\mathcal{A}$ attacking Π, there exists another adversary (or simulator) $\mathcal{S}$ attacking $\mathcal{F}$, such that the global outputs of the two executions are polynomially indistinguishable by a distinguisher $\mathcal{Z}$.

Although the above basic emulation approach is easy to understand, the security guarantees only hold for a single execution of the protocol, without considering its interactions with other protocols. The UC framework is intended to capture security under general composition operations when an arbitrary number of cryptographic protocols are executing concurrently in an environment. This framework also uses the emulation paradigm, but in a considerably more complicated manner. In particular, the distinguisher $\mathcal{Z}$, instead of merely able to inspect global output, is much more powerful in the UC framework. Modeled as an adversarial entity, $\mathcal{Z}$ is called an **environment** in the UC framework, capable of generating inputs to all parties, reading all outputs, and interacting with the adversary in any arbitrary manner during the execution.

Definition 4.3.11 *A protocol* Π *is said to* **securely realize** *an ideal functionality* $\mathcal{F}$ *if for any adversary* $\mathcal{A}$ *attacking* Π*, there exists a simulator* $\mathcal{S}$ *attacking* $\mathcal{F}$*, such that no environment* $\mathcal{Z}$ *can distinguish whether it is interacting with* $\mathcal{A}$ *and the parties running* Π*, or with* $\mathcal{S}$ *and the parties interacting with* $\mathcal{F}$*.*

Figure 4.5 gives an illustration of Definition 4.3.11. To obtain a security definition for secure KE protocols in the UC framework, it suffices to develop an appropriate description of the ideal functionality of secure key establishment protocols. Such an ideal functionality $\mathcal{F}_{\mathrm{KE}}$ is proposed by Canetti and Krawczyk [19] and is shown in Figure 4.6.

Definition 4.3.12 (UC-Security) *A KE protocol* Π *is called* **UC-secure** *if it securely realizes* $\mathcal{F}_{KE}$*.*

UC notions of security have also been developed for encryption schemes, signature schemes, bit commitment, oblivious transfers, electronic voting protocols, message trans-

missions protocols, and others [15].

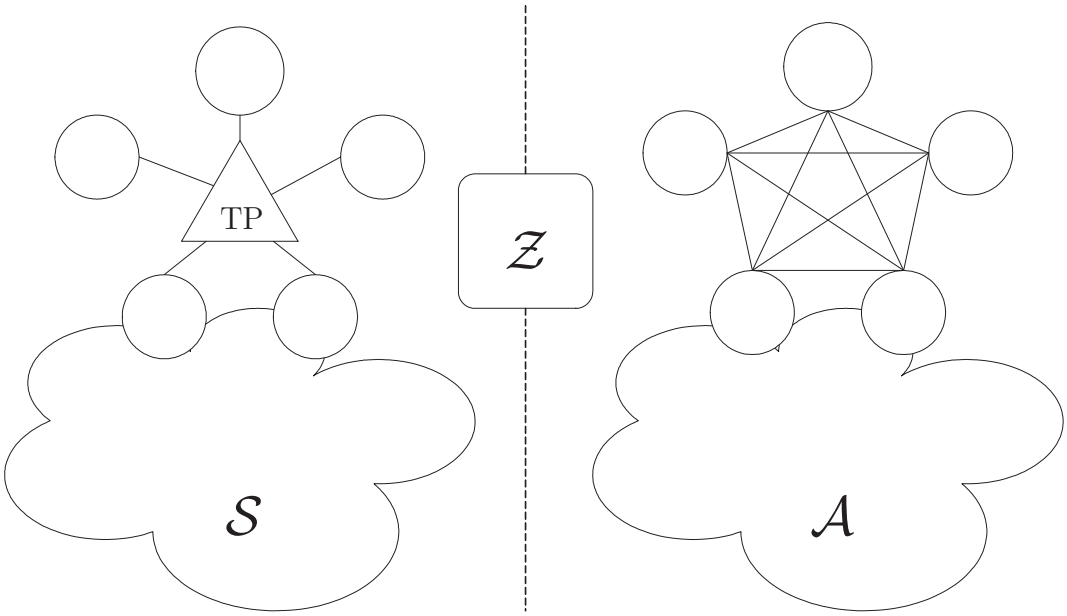

Figure 4.5: An illustration of the UC notion of security.

Critiques and comparisons. Security definitions for the UC framework model provide general composability properties in concurrent execution environments, but often at the expense of ruling out protocols that do not seem problematic in practice. As Theorem 4.3.13 shows, this is also the case for KE protocols.

Theorem 4.3.13 *[19] A key establishment protocol that is UC-secure is also SK1-secure, but the converse is not true.*

Canetti and Krawczyk developed a relaxed notion of the UC-Security definition, and showed that the SK1-Security notion and the relaxed UC-Security notion are equivalent. On one hand, this "fix" is a welcoming sign, since it not only confirms that formulations

Setting: Security parameter k, with parties $P_1, \cdots P_n$ and an adversary $\mathcal{S}$.

1. Upon receiving (establish-session, $sid, P_i, P_j, role$) from party P_i, record the value ($sid, P_i, P_j, role$) and also send it to $\mathcal{S}$. Similarly, upon receiving (establish-session, $sid, P_j, P_i, role'$) from party P_j, record the value ($sid, P_j, P_i, role'$) and also send it to $\mathcal{S}$.

2. When both ($sid, P_i, P_j, role$) and ($sid, P_j, P_i, role'$) are recorded (regardless of whether $role = role'$ or not), then:

   (a) If both P_i and P_j are uncorrupted, then pick $K \in_R \{0,1\}^k$, send K to P_i and P_j, send (sid, P_i, P_j) to $\mathcal{S}$, and halt.

   (b) If either P_i or P_j is corrupted, then request a value $K \in \{0,1\}^k$ from $\mathcal{S}$, then send K to P_i and P_j, and halt.

3. Upon receiving (corrupt-party, P_i) or (corrupt-party, P_j) from $\mathcal{S}$, check whether the value K has been sent yet. If so, send K to $\mathcal{S}$. Otherwise, send nothing to $\mathcal{S}$.

Figure 4.6: An ideal two-party KE functionality $\mathcal{F}_{\text{KE}}$.

based on indistinguishability and emulatability approaches can sometimes be made equivalent, but also implies that the SK1-Security definition carries more composability properties than previously thought. On the other hand, if "fixes" and relaxations are always necessary, critics of the UC framework are concerned about whether the complexity of the framework and the restrictiveness of the resulting security notions outweigh its practical applicabilities to real world protocols. Nevertheless, general composability frameworks, including the UC framework, are an important and exciting area of research in cryptography, and will remain so in the near future.

Chapter 5

Applications of the Canetti-Krawczyk Model

In this chapter we prove that the basic Diffie-Hellman protocol (2DH) is SK1-secure in the CK01 model (cf. Definition 3.5.1), and give a sketch of the proof that the SIGMA protocol is SK2-secure (cf. Definition 4.3.2) in the CK02 model. These proofs are due to Canetti and Krawczyk [17, 18].

5.1 Basic Diffie-Hellman Protocols

The basic two-move Diffie-Hellman protocol (2DH) is presented in Figure 5.1. We first show that 2DH is SK1-secure in the authenticated-links model. We then show the protocol can be transformed via the compiler technique introduced in Section 3.6 to become SK1-secure in the unauthenticated-links model.

5.1.1 Partial information about Diffie-Hellman keys

The 2DH protocol is implemented over a prime order subgroup $S_q = \langle g \rangle$ of the full multiplicative group $\mathbb{Z}_p^*$. Suppose for a moment that it is instead implemented over the full mul-

- Setting: p is a k-bit prime, q is a prime such that $q|p-1$, and $\langle g \rangle = S_q$.
- Protocol messages:
  1. On input (A, B, s), A picks $x \leftarrow_R \mathbb{Z}_q$ and sends (A, s, g^x) to B.
  2. Upon receiving (A, s, g^x), B picks $y \leftarrow_R \mathbb{Z}_q$ and sends (B, s, g^y) to A. B completes and outputs the session key $(g^x)^y$ under session s.
  3. Upon receiving (B, s, g^y), A also completes and outputs the session key $(g^y)^x$ under session s.

Figure 5.1: The two-move Diffie-Hellman (2DH) protocol in the AM

tiplicative group $\mathbb{Z}_p^* = \langle h \rangle$, with $K = h^{xy}$ as the resulting session key. By the intractability of the Diffie-Hellman Problem, a passive adversary shouldn't be able to recover K even though it can obtain h^x and h^y by eavesdropping the network. However, this is not enough to satisfy the definition of SK1-Security, as the adversary may still be able to obtain partial information about K. For example, even though the adversary may not be able to recover all bits of K, it may be able to find the last 56 bits of K. This is certainly not a desirable situation, as the adversary now has a non-negligible advantage in the distinguishability test.

Indeed, it is easy to compute the quadratic character of $K = h^{xy}$ in $\mathbb{Z}_p^*$ given h^x and h^y. The reason is as follows. Recall that the Legrendre symbol

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue} \pmod p, \\ -1 & \text{if } a \text{ is a non-quadratic residue} \pmod p, \\ 0 & \text{if } a \text{ divides } p, \end{cases}$$

and that h^x is a quadratic residue $\pmod p$ if and only if x is even. Thus, we can easily find the parities of x and y from h^x and h^y respectively. The parity of xy can then be recovered, from which the quadratic character of h^{xy} can be easily deduced.

Luckily, no method is known for determining any partial information about g^{xy} in a prime order subgroup $\langle g \rangle$ of $\mathbb{Z}_p^*$. That is, the Decisional Diffie-Hellman Problem (DDHP) for prime order subgroups of $\mathbb{Z}_p^*$ is believed to be intractable. In practice, however, one usually also applies a hash function or a key derivation function to g^{xy} as mentioned in Section 2.1.

5.1.2 SK1-Security in the authenticated-links model

Theorem 5.1.1 *[17] Protocol 2DH is SK1-secure in the AM, assuming that the Decisional Diffie-Hellman Problem (DDHP) for prime-order subgroups is intractable.*

Proof. We show that 2DH satisfies both Properties 1 and 2 of Definition 3.5.1

Property 1: If both party A and B complete matching sessions while uncorrupted, then B must have received g^x from the first flow, and A must have received g^y from the second flow. This is so because an AM adversary is not allowed to modify or inject messages belonging to uncorrupted parties. Thus, A computes $(g^y)^x$ and B computes $(g^x)^y$, so both establish the same session key g^{xy}.

Property 2: Assume to the contrary that there exists a KE-adversary $\mathcal{A}$ in the AM such that

$$\Pr(\mathbf{Exp}_{\mathcal{A}}^{\mathsf{2DH}}(k) = b) = \frac{1}{2} + \epsilon(k),$$

where $\epsilon(k)$ is a non-negligible function. We use $\mathcal{A}$ to construct a DDH distinguisher $\mathcal{D}_{\mathcal{A}}(Z)$, which outputs $b' = 0$ if it guesses Z comes from $Q_0 = \{(g^x, g^y, g^{xy}) : x, y \leftarrow_R \mathbb{Z}_q\}$, and outputs $b' = 1$ if it guesses Z comes from $Q_1 = \{(g^x, g^y, g^z) : x, y, z \leftarrow_R \mathbb{Z}_q\}$. We show that $\mathcal{D}_{\mathcal{A}}$ succeeds with non-negligible advantage in k when its input is chosen from Q_0 and Q_1 each with $\frac{1}{2}$ probability, thus contradicting the DDH assumption.

The distinguisher $\mathcal{D}_{\mathcal{A}}(\alpha, \beta, \gamma)$ works as follows in the AM:

1. Pick $r \leftarrow_R \{1, \ldots, l\}$, where l is the upper bound on the number of sessions invoked by $\mathcal{A}$ in any interaction. Let (P_i, P_j, s) be the rth session.

2. Invoke $\mathcal{A}$ to interact with parties $P_1, \ldots, P_n$ running the 2DH protocol in the AM, responding to $\mathcal{A}$'s requests as follows:

   - *Message exchanges:* If $\mathcal{A}$ activates (P_i, P_j, s), then let P_i send (P_i, s, α) to P_j. If P_j receives (P_i, s, α), let P_j send (P_j, s, β) to P_i. Otherwise, relay the exchanges of messages as in a real 2DH interaction.
   - *Party corruption:* If $\mathcal{A}$ corrupts party P_i, then give $\mathcal{A}$ all information about party P_i. $\mathcal{D}_{\mathcal{A}}$ outputs $b' \leftarrow_R \{0,1\}$ and returns.
   - *Session exposure:* If A exposes session (P_i, P_j, s), then $\mathcal{D}_{\mathcal{A}}$ outputs $b' \leftarrow_R \{0,1\}$ and returns. Otherwise, expose the session as in a real interaction.
   - *Test-session query:* If $\mathcal{A}$ picks (P_i, P_j, s) as the test session, then give γ to $\mathcal{A}$ as the test challenge. Otherwise, $\mathcal{D}_{\mathcal{A}}$ outputs $b' \leftarrow_R \{0,1\}$ and returns.

3. If $\mathcal{A}$ halts without choosing a test-session, then $\mathcal{D}_{\mathcal{A}}$ outputs $b' \leftarrow_R \{0,1\}$ and returns. Otherwise, if $\mathcal{A}$ outputs b as its guess to the test challenge, then $\mathcal{D}_{\mathcal{A}}$ outputs $b' \leftarrow b$ and returns.

Let E be the event that $\mathcal{A}$ picks (P_i, P_j, s) as the test-session, which happens with probability $\Pr(E) = \frac{1}{l}$. If E occurs, then $\mathcal{A}$ is given γ by $\mathcal{D}_{\mathcal{A}}$ as the test challenge. If $Z = (\alpha, \beta, \gamma)$ comes from Q_0, then γ has the same property as a "real" test-challenge key in a real, non-simulated experiment; if Z comes from Q_1, then γ has the same property as a "random" test-challenge key. Now, since $\mathcal{D}_{\mathcal{A}}$ returns the same answer as that of $\mathcal{A}$, the simulated test-challenge is a perfect simulation. Thus,

$$\Pr(\mathcal{D}_{\mathcal{A}}(Z) = b' \mid E) = \Pr(\mathbf{Exp}_{\mathcal{A}}^{\mathsf{2DH}}(k) = b).$$

On the other hand, if E doesn't occur, then $\mathcal{D}_{\mathcal{A}}$ always outputs a uniformly random bit. Since Q_0 and Q_1 are also chosen uniformly randomly, we have

$$\Pr(\mathcal{D}(Z) = b' \mid \bar{E}) = \frac{1}{2}.$$

Therefore,

$$\begin{aligned}
\Pr(\mathcal{D}_{\mathcal{A}}(Z) = b') &= \Pr(\mathcal{D}_{\mathcal{A}}(Z) = b' \mid E)\Pr(E) + \Pr(\mathcal{D}_{\mathcal{A}}(Z) = b' \mid \bar{E})\Pr(\bar{E}) \\
&= \Pr(\mathbf{Exp}_{\mathcal{A}}^{\mathsf{2DH}}(k) = b)\frac{1}{l} + \frac{1}{2}\left(1 - \frac{1}{l}\right) \\
&= \left(\frac{1}{2} + \epsilon\right)\frac{1}{l} + \frac{1}{2}\left(1 - \frac{1}{l}\right) \\
&= \frac{1}{2} + \frac{\epsilon}{l}.
\end{aligned}$$

Thus, $\left|\Pr(\mathcal{D}_{\mathcal{A}}(Z) = b') - \frac{1}{2}\right| = \frac{\epsilon}{l}$ is non-negligible in k, a contradiction. □

5.1.3 SK1-Security in the unauthenticated-links model

As explained in Section 3.6, we can now apply the authenticator $C_{\lambda_{sig}}$ to the 2DH protocol to obtain a protocol that is SK1-secure in the UM. The resulting protocol 2DH′ consists of the following protocol messages:

$$1.\begin{cases} 1. & A \longrightarrow B : A, s, g^x \\ 2. & A \longleftarrow B : A, s, g^x, N_B \\ 3. & A \longrightarrow B : A, s, g^x, \mathrm{sig}_A(A, s, g^x, N_B, B) \end{cases}$$

$$2.\begin{cases} 4. & A \longleftarrow B : B, s, g^y \\ 5. & A \longrightarrow B : B, s, g^y, N_A \\ 6. & A \longleftarrow B : B, s, g^y, \mathrm{sig}_B(B, s, g^y, N_A, A) \end{cases}$$

The 2DH′ protocol has a total of six message flows. The first three flows come from the first message in 2DH, and the last three flows come from the second message. Because six flows are required, the protocol is inefficient in practice and does not resemble any real-world protocols that have been proposed.

Canetti and Krawczyk suggest reducing the number of flows by joining (piggy-backing) common flows and using the Diffie-Hellman exponents (i.e. g^x and g^y) both as ephemeral

keys and as nonces. The resulting protocol, called SIG-DH, is illustrated in Figure 5.2. It is remarkably similar to the ISO protocol specified in [1].

- Setting: p is a k-bit prime, q is a prime such that $q|p-1$, and $\langle g \rangle = S_q$. All parties have private/public key pairs generated from a secure public-key signature scheme $\mathcal{S}$.
- Protocol messages:
  1. On input (A, B, s), A picks $x \leftarrow_R \mathbb{Z}_q$ and sends (A, s, g^x) to B.
  2. Upon receiving (A, s, g^x), B picks $y \leftarrow_R \mathbb{Z}_q$ and sends the message $(B, s, g^y, \mathrm{sig}_B(B, s, g^y, g^x, A))$ to A.
  3. Upon receiving $(B, s, g^y, \mathrm{sig}_B(B, s, g^y, g^x, A))$, A verifies B's signature. If the verification succeeds, then A sends $(A, s, \mathrm{sig}_A(A, s, g^x, g^y, B))$ to B. A completes and outputs the session key g^{xy} under session s.
  4. Upon receiving $(A, s, \mathrm{sig}_A(A, s, g^x, g^y, B))$, B verifies A's signature. If the verification succeeds, then B also completes and outputs the session key g^{xy} under session s.

Figure 5.2: A signature-based Diffie-Hellman protocol (SIG-DH) in the UM

Although the technique used by Canetti and Krawczyk in reducing the number of flows seems intuitive, it is not at all clear whether the changes will circumvent security. For instance, how can one be sure that combining flow 2 and flow 6 in 2DH′ has no adverse effects? Putting this issue aside, we have the following result:

Theorem 5.1.2 *[17] Protocol SIG-DH is SK1-secure in the UM, provided that the DDHP is intractable and the signature scheme $\mathcal{S}$ is existentially unforgeable by chosen message attacks.*

PROOF. The result follows directly from Theorems 3.6.2 and 5.1.1 applied using $C_{\lambda_{sig}}$. □

It is worth noting that Wang [47] considered the piggybacking issue more extensively, and gave an informal proof that the SIG-DH protocol is a secure mutual authentication protocol.

5.2 IKE/SIGMA Protocols

The Internet Key Exchange (IKE) Protocol [27] is a Diffie-Hellman type key establishment protocol specified in the Internet Protocol Security (IPSec) standards [32]. The IKE protocol is an important protocol for two reasons. Firstly, the protocol is the *de facto* standard for enabling secure channels communications at the IP layer in TCP/IP. Secondly, the protocol is designed as a result of a collective engineering effort, incorporating design and implementation lessons learned in the past. In IKE, three modes of operations are supported, where the modes differ in the ways key authentications are performed. In this section, we describe a particular mode of operation, known as the "SIGn-and-MAc" (SIGMA) mechanism. The other two modes of operations, which are public-key encryption and digital signatures, are not discussed in this section.

The SIGMA family of protocols has three distinctive features:

1. SIGn-and-MAc design,
2. SK2-Security in the CK02 model, and
3. optional identity protecting features.

The basic version of IKE, denoted Σ_0, is given in Figure 5.3. The actual IKE protocol in SIGMA mode, however, differs from the one we have presented in two ways. First, the MAC values are sent inside the signatures rather than separately. Next, the session identifier s is included in the MAC values. Thus, message data of the form $\text{sig}_B(g^y, g^x), \text{MAC}_{k_1}(B)$ are replaced with $\text{sig}_B(\text{MAC}_{k_1}(s, g^y, g^x, B))$. We present the simplified version here in order

to highlight the SIGMA design as incorporated into the protocol messages. Krawczyk [34] showed that the modifications do not affect the security results.

We discuss the three main features of the SIGMA protocols in the next three subsections.

- Setting: p and q are prime numbers with $q|p-1$. $g \in \mathbb{Z}_p^*$ is a generator of order q. Suppose A and B randomly select ephemeral keys x and y in $[0, q-1]$ respectively.
- Protocol messages:
  1. Start message $(A \rightarrow B) : s, g^x$
  2. Response message $(A \leftarrow B) : s, g^y, B, \text{sig}_B(g^y, g^x), \text{MAC}_{k_1}(B)$
  3. Finish message $(A \rightarrow B) : s, A, \text{sig}_A(g^x, g^y), \text{MAC}_{k_1}(A)$
- Upon receiving the message in Step 2, A verifies B's signature using B's public key, then verifies the MAC under $k_1 = \text{prf}_{g^{xy}}(1)$. It completes its session by computing the secret key $k_0 = \text{prf}_{g^{xy}}(0)$.
- Upon receiving the message in Step 3, B verifies A's signature using A's public key, then verifies the MAC under k_1. It then completes the session by computing k_0.

Figure 5.3: The basic SIGMA protocol (Σ_0)

5.2.1 SIGn-and-MAc design

The design of the SIGn-and-MAc (SIGMA) family of protocols was strongly influenced by design elements from the STS protocol. It has two design rationales:

SIGning of g^x and g^y — Digital signatures are applied to g^x and g^y by each party in order to protect the ephemeral keys from modifications. Unlike the SIG-DH protocol

design in which the peer's identity is also included in the signature, this particular design allows a party to authenticate itself to another party without knowing the peer's identity. As a consequence, SIGMA protocols are suitable for providing identity protection (cf. Section 5.2.3).

MAC-ing of sender identity under g^{xy} — The MAC applied to the sender identity under the Diffie-Hellman key is sufficient to provide the necessary binding between the session key and the identity of the sender as to avoid possible UKS attacks. If the MAC is omitted, this protocol becomes the (insecure) basic authenticated Diffie-Hellman protocol described in Section 4.1.2.

5.2.2 SK2-Security in the CK02 model

While the proof of security of the IKE protocol is too complicated to present in full detail, we state the main result and present the main ideas in the proof.

Theorem 5.2.1 (Σ_0 is SK2-secure) *[18] Protocol Σ_0 is SK2-secure in the CK02 model, provided that the DDHP is intractable and that all of* $\mathrm{sig}(\cdot), \mathrm{MAC}(\cdot)$ *and* $\mathrm{prf}(\cdot)$ *are secure (under their respective definitions of security).*

Sketch of Proof. We show that Σ_0 satisfies Property 1 of Definition 4.3.2 in detail, and briefly sketch the proof of Property 2.

Property 1: We need to show that if two uncorrupted parties A and B complete matching sessions (A, s, B) and (B, s, A), then both compute the same Diffie-Hellman value g^{xy}, regardless of the UM adversary $\mathcal{U}$'s actions.

Since both sessions have been completed, A must have picked an ephemeral secret x, generated a start message of the form $(s, u_A = g^x)$, and received a response message of the form $(s, v_A, B, s_A, \cdots)$ where v_A is a DH exponential and s_A is a signature (the identity and the MAC are omitted for clarity of presentation). Similarly, B must have received a start message of the form (s, u_B), picked an ephemeral secret y, and sent a response

message of the form $(s, v_B = g^y, B, s_B = \mathrm{sig}_B(v_B, u_B), \cdots)$. Note that in the presence of a UM adversary $\mathcal{U}$, it is possible that $u_A \neq u_B$, $v_A \neq v_B$, and/or $s_A \neq s_B$. The situation is illustrated in Figure 5.4.

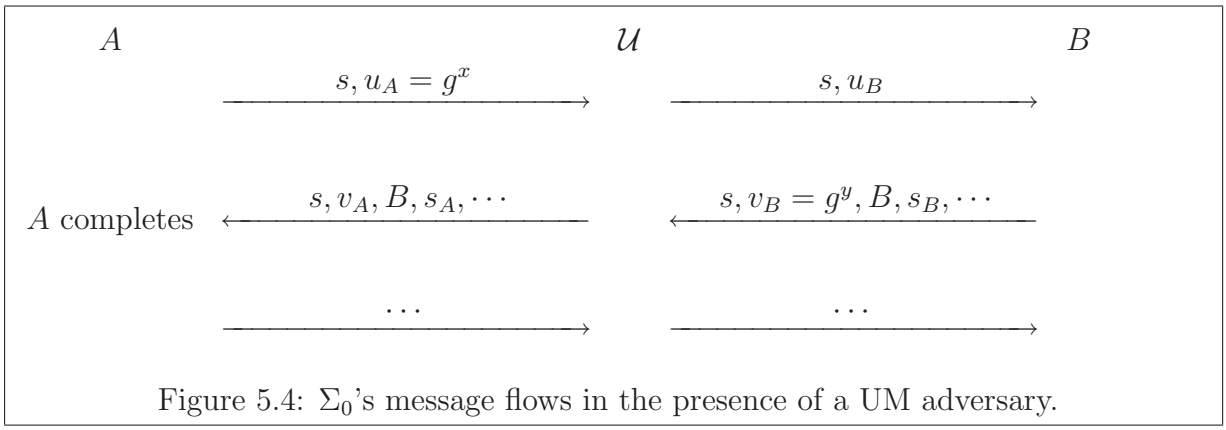

Figure 5.4: Σ_0's message flows in the presence of a UM adversary.

Since A has completed, it must have verified that the signature s_A is valid against (v_A, u_A) using B's public key. Thus if $(u_A, v_A) \neq (u_B, v_B)$, then the adversary has succeeded in forging a new message/signature pair $(u_A, v_A), s_A$ under B's identity, contradicting the security of the signature scheme. Therefore, we must have $u_A = u_B$ and $v_A = v_B$, except with negligible probability.

Therefore, (A, s, B) produces the Diffie-Hellman key $v_A^x = v_B^x = (g^y)^x = g^{xy}$, and (B, s, A) produces $u_B^y = u_A^y = (g^x)^y = g^{xy}$, so they both establish the same key.

Property 2: The main idea of the proof involves showing that if an attacker can win the test-session experiment with non-negligible advantage, then it can in turn be used to construct an attack against one of the cryptographic primitives $(\mathrm{sig}, \mathrm{MAC}, \mathrm{prf})$ or a distinguisher for the DDHP. The full proof presented in [18] contains an overwhelming amount of technical details and will not be replicated here. □

We remark that the protocol also satisfies the stronger SK3 security definition [34].

5.2.3 Identity protecting variants

One of the design goals of the SIGMA protocol is to optionally provide identity protection against active attackers. By separately signing only the ephemeral keys and MAC-ing only the message sender's identity (and not also its peer's identity), it is possible that one of the peers delays communicating its own identity until it learns the other peer's identity. Identity protection can then be easily provided by symmetrically encrypting the identities, the signature and the MAC by a key derived from g^{xy}.

Two variants of SIGMA are possible, SIGMA-I, which protects the identity of the initiator, and SIGMA-R, which protects the identity of the responder. Recall that it is not possible to protect the identities of both parties against an active adversary.

The protocol messages for the SIGMA-I (Σ_I) protocol are as follows:

1. $A \longrightarrow B : s, g^x$
2. $A \longleftarrow B : s, g^y, E_{k_2}(B, \mathrm{sig}_B(g^x, g^y), \mathrm{MAC}_{k_1}(B))$
3. $A \longrightarrow B : s, E_{k_2}(A, \mathrm{sig}_A(g^y, g^x), \mathrm{MAC}_{k_1}(A))$

The protocol messages for the SIGMA-R (Σ_R) protocol are as follows:

1. $A \longrightarrow B : s, g^x$
2. $A \longleftarrow B : s, g^y$
3. $A \longrightarrow B : s, E_{k_2}(A, \mathrm{sig}_A(g^y, g^x), \mathrm{MAC}_{k_1}(A))$
4. $A \longleftarrow B : s, E_{k_2}(B, \mathrm{sig}_B(g^x, g^y), \mathrm{MAC}_{k_1}(B))$

Although the security of the Σ_I and Σ_R protocols seems to follow naturally from the security of the Σ_0 protocol, we cannot affirmatively conclude so until we have provided rigorous proofs of security. To do so, we first need to show that the modified protocols are still SK2 secure. We then need to explicitly prove that the protocols provide identity protection, since the SK2 definition does not guarantee this property. Fortunately, both statements are true and have been proven in [18].

Theorem 5.2.2 (Σ_I and Σ_R are SK2-secure) *[18] Protocols Σ_I and Σ_R are SK2-secure in the CK02 model, provided that the DDHP is intractable and that all of $E(\cdot), \mathrm{sig}(\cdot), \mathrm{MAC}(\cdot)$ and $\mathrm{prf}(\cdot)$ are secure (under their respective definitions of security).*

Theorem 5.2.3 (Σ_I and Σ_R offer identity protection) *[18] Protocol Σ_I provides identity protection for both parties against passive attackers, and identity protection for the initiator against active attackers. Protocol Σ_R provides identity protection for both parties against passive attackers, and identity protection for the responder against active attackers.*

Chapter 6

Conclusion

This thesis presented the security models developed by Canetti and Krawczyk for analyzing the security of key establishment protocols. We have thoroughly examined and critiqued the models, looked at and compared the various definitions of security, and provided several proofs of security under the models. Although we have exclusively focused our attention on key establishment protocols, it should be clear that the provable security concepts and techniques that we have presented can be readily applied to the study of other cryptographic schemes.

There are both practical lessons that we have learned and open questions that we have discovered. In concluding this thesis, Section 6.1 gives practical advices on designing and implementing key establishment protocols, and Section 6.2 discusses open questions and future work.

6.1 Key Establishment Best Practices

Practitioners who are serious about implementing and using key establishment protocols should choose those that are provably secure whenever possible. In addition, there are a number of details that should be carefully considered:

1. Choose public key security parameters carefully (cf. Section 1.3.3).

2. Always apply a key derivation function to the derived session key, especially when using Diffie-Hellman type systems (cf. Section 2.1).

3. Properly erase the session key from memory when it is no longer in use (cf. Section 3.3).

4. Make sure public-key validations are performed by the public-key infrastructure to prevent unknown key share attacks (cf. Section 4.1.2).

5. Always use provably secure cryptographic primitives as building blocks for implementations (cf. Theorem 5.1.2 and Theorem 5.2.1).

6.2 Open Questions and Future Work

We have posted several open questions throughout this thesis. In Section 4.2.2, we ask whether the Canetti-Krawczyk model can be modified so that a security definition ensuring key compromise impersonation resilience can be formulated. In Section 4.3.1, we post two open problems on the relationships between the several definitions of security in the CK01 and CK02 models. In Section 5.1.2, we suggest to formally analyze the security impact of "collapsing" common flows after applying an authenticator.

In addition, there are several avenues for further research. First, it will be worthwhile to work on providing formal proofs of security for the MQV protocol or its variants, if there exists any. This may not be possible until we have a clear understanding of the intractability assumptions underlying the protocols and better techniques to simplify the proof process. Also, there has yet to appear a treatment of key establishment protocols in the context of practice-oriented provable security [3]. Simply put, practice-oriented provable security attempts to give practical recommendations on key lengths, protocol efficiency and other parameters by measuring the reductions quantitatively, rather than asymptotically. We are interested in knowing how much security is lost in the compilation process using authenticators. We are also interested in seeing whether the practice-oriented approach will give new insights into the design of key establishment protocols.

Bibliography

[1] ISO/IEC IS 9798-3. Entity authentication mechanisms — Part 3: Entity authentication using asymmetric techniques, 1993.

[2] Richard Ankney, Don Johnson, and Michael Matyas. The Unified Model – contribution to X9F1, October 1995.

[3] Mihir Bellare. Practice-oriented provable-security. In *Proceedings of First International Workshop on Information Security (ISW 97)*, volume 1396 of *Lecture Notes in Computer Science*, pages 221–231, 1998.

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key-exchange protocols. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 1998. Full version available at `http://eprint.iacr.org/1998/009/`.

[5] Mihir Bellare, Erez Petrank, Charles Rackoff, and Phillip Rogaway. Authenticated key exchange in the public key model, 1995-96. Unpublished manuscript.

[6] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249, 1994.

[7] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution – the three party case. In *Proceedings of the 27th Annual ACM Symposium on Theory of*

Computing (STOC), pages 232–249, 1995. Full version available at `http://www.cs.ucsd.edu/users/mihir/papers/3pkd.pdf`.

[8] Ray Bird, Inder Gopal, Amir Herzberg, Phil Janson, Shay Kutten, Refik Molva, and Moti Yung. Systematic design of two-party authentication protocols. In *Advances in Cryptology – CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 44–61, 1991.

[9] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key exchange protocols and their security analysis. In *Proceedings of the Sixth IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45, 1997. Full version available at `http://www.cacr.math.uwaterloo.ca/~ajmeneze/publications/agreement.ps`.

[10] Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. In *Proceedings of Public Key Cryptography, Second International Workshop on Practice and Theory in Public Key Cryptography, PKC '99*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170, 1999.

[11] Colin Boyd. Towards a classification of key agreement protocols. In *8th IEEE Computer Security Foundations Workshop*, pages 38–43, 1995.

[12] Colin Boyd, Wenbo Mao, and Kenneth G. Paterson. Key agreement using statically keyed authenticators. In *Applied Cryptography and Network Security: Second International Conference, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 248–262, 2004.

[13] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[14] Michael Burrows, Martin Abadi, and Roger M. Needham. A logic for authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[15] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, pages 136–145, 2001. Full version available at `http://eprint.iacr.org/2000/067`.

[16] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM' 99*, volume 2, pages 708–716, March 1999.

[17] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474, 2001. Full version available at `http://eprint.iacr.org/2001/040`.

[18] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161, 2002. Full version available at `http://eprint.iacr.org/2002/120/`.

[19] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351, 2002. Full version available at `http://eprint.iacr.org/2002/059/`.

[20] Dorthy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[21] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[22] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[23] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge Press, 2001.

[24] Shafi Goldwasser and Silvio Micali. Probablistic encryption. *Journal of Computer & System Sciences*, 28/2:270–299, 1984.

[25] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. In *Proceedings of the 17th ACM Symposium on Theory of Computing (STOC)*, pages 291–304, 1985.

[26] Daniel Gordan. Discrete logarithms in $GF(p)$ using the number field sieve,. *SIAM Journal on Discrete Mathematics*, 6:124–138, 1993.

[27] Dan Harkins and Dave Carrel. RFC 2409: The Internet Key agreement (IKE), November 1998.

[28] Tony (C.A.R.) Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[29] Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Network and Distributed System Security Symposium*, pages 151–165, 1999. Available at `http://www.isoc.org/isoc/conferences/ndss/99/proceedings`.

[30] Mike Just and Serge Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology – ASIACRYPT 96*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49, 1996.

[31] Burt Kaliski and Jessica Staddon. RFC 2437: PKCS #1: RSA cryptography specifications version 2, October 1998.

[32] Stephen Kent and Randall Atkinson. RFC 2401: Security Architecture for the Internet Protocol, November 1998.

[33] Hugo Krawczyk. SKEME: A versatile secure key exchange mechanism for Internet. In *Network and Distributed Systems Security Symposium*, pages 114–127, 1996.

[34] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425, 2003. Full version available at `http://www.ee.technion.ac.il/~hugo/sigma.html`.

[35] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. RFC 2104: HMAC: Keyed-hashing for message authentication, February 1997.

[36] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes, and Cryptography*, 28:119–134, 2003.

[37] Jooyoung Lee and Douglas R. Stinson. Deterministic key predistribution schemes for distributed sensor networks. In *Selected Areas of Cryptography 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 294–307, 2004.

[38] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, February 1996.

[39] Alfred Menezes, Paul Van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[40] Silvio Micali and Phillip Rogaway. Secure computation. In *Advances in Cryptology – CRYPTO 91*, volume 576 of *Lecture Notes in Computer Science*, 1992. Preliminary version.

[41] James A. Muir. Techniques of side channel cryptanalysis. Master's thesis, Department of Combinatorics and Optimization, University of Waterloo, 2001. Available at `http://www.math.uwaterloo.ca/~jamuir/sidechannel.htm`.

[42] Roger Needham and Michael D. Schröder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[43] Radia Perlman and Charlie Kaufman. Key exchange in IPSec: Analysis of IKE. *IEEE Internet Computing*, 4(6):50–56, 2000.

[44] Phillip Rogaway. On the role of definitions in and beyond cryptography. In *The Ninth Asian Computing Science Conference (ASIAN'04)*, volume 3321 of *Lecture Notes in Computer Science*, pages 13–32, 2004.

[45] Victor Shoup. On formal models for secure key exchange. Technical Report IBM Research Report RZ 3120, IBM Research, 1999.

[46] Victor Shoup and Avi Rubin. Session key distribution using smart cards. In *Advances in Cryptology – EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 321–331, 1996.

[47] Hao-Hsien Bobby Wang. Desired features and design methodologies of secure authenticated key exchange protocols in the public-key infrastructure setting. Master's thesis, School of Computer Science, University of Waterloo, 2004. Available at `http://etheses.uwaterloo.ca/display.cfm?ethesis_id=476`.