Classical Authenticated Key Exchange and Quantum Cryptography

by

Douglas Stebila

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics & Optimization

Waterloo, Ontario, Canada, 2009

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

Cryptography plays an integral role in secure communication and is usually the strongest link in the chain of security. Yet security problems abound in electronic communication: spyware, phishing, denial of service, and side-channel attacks are still major concerns. The main goal in this thesis is to consider how cryptographic techniques can be extended to offer greater defence against these non-traditional security threats.

In the first part of this thesis, we consider problems in classical cryptography. We introduce *multi-factor password-authenticated key exchange* which allows secure authentication and key agreement based on multiple short secrets, such as a long-term password and a one-time response; it can provide an enhanced level of assurance in higher security scenarios because a multi-factor protocol is designed to remain secure even if all but one of the factors has been compromised due to attacks such as phishing or spyware. Next, we consider the integration of *denial of service* countermeasures with key exchange protocols: by introducing a formal model for denial of service resilience that complements the extended Canetti-Krawczyk model for secure key agreement, we cover a wide range of existing denial of service attacks and prevent them by carefully using client puzzles. Additionally, we look at how *side-channel attacks* affect certain types of formulæ used in elliptic curve cryptography, and demonstrate that information leaked during field operations such as addition, subtraction, and multiplication can be exploited by an attacker.

In the second part of this thesis, we examine cryptography in the quantum setting. We argue that *quantum key distribution* will have an important role to play in future information security infrastructures and will operate best when integrated with the powerful public key infrastructures that are used today. Finally, we present a new look at *quantum money* and describe a quantum coin scheme where the coins are not easily counterfeited, are locally verifiable, and can be transferred to another party.

Acknowledgements

I am grateful to the many people who have helped me throughout my PhD. First, I thank my supervisor, Michele Mosca, for his guidance and support over the years. I am also grateful for the mentorship of my industrial supervisor, Sheueling Chang of Sun Microsystems Laboratories, and the kind advice of Alfred Menezes.

Much of the work that appears in this thesis is the result of collaboration with others: Sheueling Chang, Norbert Lütkenhaus, Matthew McKague, Michele Mosca, Nicholas Thériault, Poornaprajna Udupi, and Berkant Ustaoglu.

I have had many helpful discussions with others during this PhD, including Scott Aaronson, Niel de Beaudrap, Anne Broadbent, Donny Cheung, Isabelle Déchène, Joseph Fitzsimons, Ian Goldberg, Nils Gura, Elham Kashefi, Debbie Leung, Bodo Möller, Kenny Paterson, Barry Sanders, Miklos Santha, and John Watrous, and received helpful reports from various anonymous referees.

My research and studies have been supported by an NSERC Industrial Postgraduate Scholarship in conjunction with Sun Microsystems Laboratories, an NSERC Canada Graduate Scholarship, and University of Waterloo Sinclair, William Tutte, and President's Postgraduate Scholarships, as well as funding for the Institute for Quantum Computing from CIFAR, CFI, CSE, MITACS, and ORDCF.

Finally, I would like to acknowledge those whose friendship was especially important to me during my PhD — Paul Dickinson, Matthew McKague, and Lana Sheridan — and my parents who have encouraged me throughout my studies.

Table of Contents

List of Figures

Chapter 1

Introduction

Contents

1.1 Introduction

Cryptography plays an essential role in modern life as it allows for secure communication over insecure channels, even in the face of powerful adversaries. Cryptography has been used for military and government communications for more than 2000 years, but only in the past 20 years has cryptography come to be used in day-to-day life. The expansion of the Internet from a small-scale academic network to a global network enabling hundreds of billions of dollars of electronic commerce [Sho08] was due in no small part to the availability of cryptography.

The successful design of multitudes of secure cryptographic algorithms — public key agreement, digital signatures, block and stream ciphers, hash functions, message authentication codes — and secure protocols that employ these primitives is a remarkable achievement. Of those that have been widely adopted, the majority have remained fundamentally secure despite years of intensive cryptanalysis and advancing computer technology.

Yet security problems abound on the Internet. The servers of governments and major corporations are subjected to denial of service attacks. Spyware, viruses, and malware are installed on users' computers. Individuals fall victim to phishing attacks and identity theft. Devices are subject to attacks that exploit subtle variations in their power usage.

These security problems do not arise as a result of a break of cryptographic algorithms or protocols. Rather, they arise by working around the cryptography:

why exert billions of hours of computer effort to break an advanced encryption protocol to crack a user's password when you can simply trick the user into telling you their password directly?

The first part of this thesis aims to extend cryptographic techniques to offer greater defence against some of these non-traditional security threats. Using cryptographic protocols, we can defend users' passwords from phishing attacks that aim to trick the user into revealing their password to an attacker. We can protect servers from being flooded by millions of bogus requests that would otherwise lead to a denial of service attack during key establishment protocols. We can guard against side-channel attacks that exploit variations in power consumption in elliptic curve cryptography. By extending the scope of cryptography to these settings, we can hope to offer greater robustness against attacks in a hostile communications network.

While the discussion above deals with present threats, it is important to be aware of threats that are on the horizon. One of the most significant discoveries in theoretical computer science in the last 15 years was the field of quantum computation which grew out of the notion that all computation is physical: computational devices that exploit the properties of quantum physics will have a significant advantage compared to current computers where the computational model is based on classical physics. Most importantly for cryptography, large scale quantum computers would be able to solve the hard problems that underlie the most widely used public key cryptography systems — RSA, finite field Diffie-Hellman, and elliptic curve cryptography — and which are the basis of security for the techniques in the first part of this thesis.

Thus, the second part of this thesis considers the field of cryptography in a quantum setting. Though quantum computers may eventually take away many of the public key techniques that are widely used today, quantum cryptography makes new tools available for use. The most researched of these new techniques is quantum key distribution, where two parties can establish a secure key over an authentic, public channel by using quantum communication. Most interestingly, the key that is established is entirely independent of any inputs to the protocol, a feature which cannot be achieved with classical physics. Quantum techniques can also be applied to other cryptographic tasks, and one such task is digital cash: the no-cloning theorem which prevents quantum states from being copied naturally leads one to ask if one could create quantum money which could not be counterfeited because of the laws of nature.

1.2 Contributions

We now summarize the main contributions of this thesis.

1.2.1 Classical Authenticated Key Exchange

1.2.1.1 Password-Authenticated Key Exchange (Chapter 3)

More Efficient PAK (Section 3.3.2.2) and More Efficient PAK-Z+ (Section 3.3.3.2) In this section we describe more efficient forms of the PAK and PAK-Z+ protocols for password-authenticated key exchange. Our improvement for the non-verifier-based PAK protocol allows the protocol to achieve the same number of operations as Diffie-Hellman — two exponentiations — on both client and server. For the verifier-based PAK-Z+ protocol, we achieve a protocol with just three exponentiations on both client and server.

1.2.1.2 Multi-Factor Password-Authenticated Key Exchange (Chapter 4)

In this chapter, we provide the first formal security treatment of multi-factor password-authenticated key exchange. We define a formal model which is an extension of the Bellare-Pointcheval-Rogaway model [BPR00] for password-authenticated key exchange. We formalize the security notion that a multi-factor protocol should remain secure even if all but one of the factors has been compromised by adapting the definition of freshness of a session.

We present an efficient two-factor protocol, MFPAK, that is secure in this model under standard cryptographic assumptions in the random oracle model.

Our multi-factor authentication protocol offers enhanced authentication protection through two complementary factors, a long-term password and a one-time response, and achieves two-factor security with the same computational efficiency as the one-factor protocol mePAK-Z+ from Section 3.3.3.2. The protocol remains secure even if all but one of the authentication factors is fully known to an adversary. Protocols secure in our model are resistant to man-in-the-middle and impersonation attacks, providing enhanced authentication in the face of more complex threats like spyware and phishing.

Our work differs from previous work in password-authenticated key exchange because it utilizes two independent, complementary factors for authentication. Other work has considered some aspects of multi-factor authentication, but these have either used at least one factor that is a long cryptographic secret (as opposed to our work which allows both factors to be short, human-friendly strings), or have not provided strong server-to-client authentication resistant to man-in-the-middle attacks.

The work on models for multi-factor password authenticated key exchange and the MFPAK protocol in Chapter 4 is joint work with Poornaprajna Udupi and Sheueling Chang and appeared as the following publication:

- Douglas Stebila, Poornaprajna Udupi, and Sheueling Chang. Multi-factor

password-authenticated key exchange. In preparation, 2009. eprint `http://eprint.iacr.org/2008/214`.

1.2.1.3 Denial-of-Service-Resilient Authenticated Key Exchange (Chapter 5)

In this chapter, we propose a formal model for denial of service attacks in the context of the extended Canetti-Krawczyk (eCK) model for secure and authentic shared key establishment that takes into account denial of service (DoS) attacks. Whereas previous work focused mainly on proper DoS countermeasures, we are interested in the *integration* of DoS countermeasures with key establishment: just giving the client a puzzle and checking that it was solved is not enough to guarantee denial of service resistance for key agreement.

We analyze many of the DoS attacks presented in the literature and argue that they do not stem from a weak DoS countermeasure but from incorrect integration into the key agreement protocol. Existing formal methods have already led to the discovery of some novel DoS attacks, but not all attacks can be identified in this way. Our model covers a wider range of denial of service resistance goals and provides a framework which can be used to analyze and compare DoS countermeasures. Previous formal treatments of denial of service deal with the two-party setting: an honest server and an adversary. These previous models do not capture goals related to hijacking of connections, but do allow for a fine analysis of the strength of a DoS countermeasure. Our approach can be used in conjunction with previous analyses: our model deals with the integration of DoS countermeasures and key agreement protocols, and previous work can be used to analyze the strength of the countermeasures.

The work on denial-of-service-resilient key exchange in Chapter 5 is joint work with Berkant Ustaoglu.

1.2.1.4 Unified Point Addition Formulæ in Elliptic Curve Cryptography (Chapter 6)

In this chapter, we give a projective version of the elliptic curve unified point addition formulæ of Brier, Déchène, and Joye [BDJ04]. We extend Walter's side-channel attack [Wal04] to make use of the detection of a conditional addition which appears in many field subtraction implementations. This *conditional modular reduction attack* substantially decreases the amount of work necessary to recover the key: when used with projective coordinates and Montgomery field representation and combined with Walter's original technique, our attack is feasible on prime field elliptic curves up to 384 bits. We suggest some countermeasures which may help achieve constant run-time field operations. We also provide some performance results for the various unified point addition formulæ and discuss the applicability of timing attacks.

The work on side-channel attacks on unified point addition formulæ in elliptic curve cryptography in Section 6.3 is joint work with Nicholas Thériault and appeared as the following publication:

- Douglas Stebila and Nicolas Thériault. Unified point addition formulæ and side-channel attacks. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems (CHES) 2006*, *LNCS*, volume 4249, pp. 354–368. Springer, 2006. DOI:10.1007/11894063_28. EPRINT `http://eprint.iacr.org/2005/419`.

1.2.2 Quantum Cryptography

1.2.2.1 The Case for Quantum Key Distribution (Chapter 7)

In this chapter, we examine the role of quantum key distribution (QKD) in cryptographic infrastructures. QKD promises secure key agreement by using quantum mechanical systems. We argue that QKD will be an important part of future cryptographic infrastructures. It can provide long-term confidentiality for encrypted information without reliance on computational assumptions. Although QKD still requires authentication to prevent man-in-the-middle attacks, it can make use of either information-theoretically secure symmetric key authentication or computationally secure public key authentication: even when using public key authentication, we argue that QKD still offers stronger security than classical key agreement.

The work on the case for quantum key distribution in Chapter 7 is joint work with Michele Mosca and Norbert Lütkenhaus and appeared as the following publication:

- Douglas Stebila, Michele Mosca, and Norbert Lütkenhaus. The case for quantum key distribution. EPRINT arXiv:0902.2839, `http://eprint.iacr.org/2009/082`.

1.2.2.2 Quantum Money (Chapter 8)

In this chapter, we present a new type of quantum money, which we call *quantum coins*: coins are transferable, locally verifiable, and unforgeable, and have some anonymity properties. Each coin generated by the bank is a copy of the same quantum state; in such a scheme, coins should be indistinguishable from one another. Additionally, a circuit is provided to allow the coins to be verified locally and then transferred for later use.

We describe how to achieve quantum coins with black box quantum circuits and with blind quantum computation. The unforgeability of coins in our scheme comes from complexity theoretic assumptions on the adversary's running time.

Our work on quantum coins contrasts with previous quantum money schemes, which we call *quantum bills*: in a quantum bill scheme, the bank generates tokens that are classical/quantum pairs, which in general are distinct. The classical string may serve as a serial number or as some input value to be used in the verification procedure.

The work on quantum money is joint work with Michele Mosca and parts of it appeared as the following poster:

- Michele Mosca and Douglas Stebila. A framework for quantum money. In *Quantum Information Processing (QIP) 2007*.

Part I

Classical Authenticated Key Exchange

Chapter 2

Background

Contents

In this chapter, we provide some background material that builds up the ingredients we will use in cryptographic protocols. In Section 2.1 we describe the algebraic structures that we use in our protocols: modular arithmetic and elliptic curves. In Section 2.2, we describe the various computational problems that underly our cryptographic protocols.

2.1 Algebraic and number theoretic background

2.1.1 Algebra

2.1.1.1 Groups

Definition 2.1 *A* **group** G *is a set along with an operation* $\odot$ *which satisfies the following axioms:*

G1. $\odot$ *is associative.*

G2. There is an element $e \in G$, called the **identity**, *such that $e \odot a = a = a \odot e$ for all $a \in G$.*

G3. Each element in G is invertible: for each $a \in G$, there exists $b \in G$ such that $a \odot b = b \odot a = e$.

A group G is **abelian** *if the operation $\odot$ is also commutative.*

The **order** *of a group G, denoted $|G|$, is the number of elements in G. The order may be infinite. If the order is finite, then G is said to be a* **finite group**.

Sometimes we write groups using additive notation, where the operation $\odot$ is $+$; often we use this notation when the group is abelian. Sometimes we write groups using multiplicative notation, where the operation $\odot$ is $\times$; often we use this notation when the group is not necessarily abelian.

Some examples of groups are $\mathbb{Z}^+$, the integers, with addition; and $\mathbb{R}^\times$, the non-zero real numbers, with multiplication. Both of these are abelian groups.

Definition 2.2 *Let G be a group with operation $\odot$. Let H be a subset of G. Then H is a* **subgroup** *of G if H is closed under $\odot$, H contains the identity of G, and H is closed under inversion.*

Then H is a **cyclic subgroup** *of G if there exists some element $x \in G$ such that*

$$H = \{\dots, x^{-2}, x^{-1}, 1, x, x^2, \dots\}$$

where we have used multiplicative notation for the group. We say that H is the subgroup of G **generated by** *x and write $H = \langle x \rangle$; x is called a* **generator** *of H.*

Definition 2.3 *The* **order** *of an element $x \in G$, denoted $\mathrm{ord}_G(x)$, is the order of $\langle x \rangle$.*

Fact 2.4 *If G is a finite group, then all elements in G have finite order, and $\mathrm{ord}_G(x)$ is the smallest non-negative integer m such that $x^m = 1$ (in multiplicative notation).*

2.1.1.2 Rings

Definition 2.5 *A* **ring** *R is a set along with two operations $+$ and $\times$, called addition and multiplication respectively, which satisfy the following axioms:*

R1. Under $+$, R is an abelian group, with identity denoted by 0. This abelian group is denoted by R^+.

R2. $\times$ is associative and has an identity denoted by 1.

R3. Multiplication is distributive over addition: for all $a, b, c \in R$,

$$(a + b) \times c = (a \times c) + (b \times c) \quad \text{and} \quad a \times (b + c) = (a \times b) + (a \times c) \ .$$

A ring is **commutative** *if its multiplication operation is commutative.*

An element u in a ring R is a **unit** *if u has an inverse in R: namely, if there exists $v \in R$ such that $uv = vu = 1$.*

2.1.1.3 Fields

Definition 2.6 *A* **field** *F is a ring in which every non-zero element is invertible.*

A **finite field** *is a field with finitely many elements.*

The order of a finite field (i.e., the number of elements in the finite field) is always a power of a prime: $|F| = p^r = q$ for some prime p, which is called the **characteristic** *of F.*

The following are well-known results about finite fields.

Theorem 2.7 (Theorem 13.6.4, [Art91]) *Let p be a prime and let $q = p^r$ be a power of p, with $r \geq 1$. Then*

1. *There exists a field of order q.*
2. *Any two fields of order q are isomorphic.*
3. *Let F be a field of order q. The set $F^\times$ of non-zero elements of F, together with the operation of multiplication, is a cyclic group of order $q - 1$ and is called the multiplicative group of $F^\times$.*

A finite field is also called a **Galois field**. Since by the previous theorem all finite fields of the same order are isomorphic, it makes sense to speak of *the* Galois field of order n, denoted $\mathrm{GF}(n)$, where n is a prime power.

2.1.2 Number theory

Definition 2.8 *The* **greatest common denominator** *of two integers a and b, denoted $\gcd(a, b)$, is the largest integer g such that g divides a and g divides b.*

Two integers a and b are said to be **coprime** *if $\gcd(a, b) = 1$.*

Definition 2.9 *The* **Euler phi function**, *also called the Euler totient function, denoted by φ, gives the number of positive integers less than or equal to n that are coprime to n:*

$$\varphi(n) = |\{a : \gcd(a, n) = 1, a \leq n\}| \ .$$

If n is a prime number, then $\varphi(n) = n - 1$. If $n = ab$, where $a, b \in \mathbb{Z}$ are coprime, then $\varphi(n) = \varphi(a)\varphi(b)$.

Definition 2.10 *The set of integers modulo a positive integer n is denoted $\mathbb{Z}/n\mathbb{Z}$ or equivalently $\mathbb{Z}_n$.*

Theorem 2.11 *$\mathbb{Z}_n$ is a commutative ring for all positive integers n. Moreover, if n is a prime p, then $\mathbb{Z}_n = \mathrm{GF}(p)$.*

Definition 2.12 *The set of integers modulo a positive integer n that are units (i.e., have inverses) is denoted $\mathbb{Z}_n^\times$. Equivalently, this is the set of integers that are coprime to n.*

Theorem 2.13 *$\mathbb{Z}_n^\times$ under multiplication is an abelian group of order $\varphi(n)$.*

Thus, we can say that $\mathbb{Z}_n^\times$ is the **multiplicative group of integers modulo** n.

2.1.3 Elliptic curves

Curves over prime fields. In fields F of prime characteristic other than 2 or 3, the Weierstraß form of a non-supersingular **elliptic curve** E is given by the equation

$$y^2 = x^3 + ax + b \ , \tag{2.1}$$

where $a, b \in F$. [HMV04, §3.1]

The set of points in $F \times F$ on the curve, joined with the **point at infinity** $\mathcal{O}$, forms an abelian group, denoted $E(F)$, when combined with the operation of **point addition**. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, $P \neq -Q$, be two points on the curve E. These two points can be added to obtain a third point $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \text{ (addition)} \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \text{ (doubling)} \end{cases} \ . \tag{2.2}$$

Note that when $P = Q$, the above formula describes the **point doubling** operation.

Curves over characteristic 2 fields. In fields F of characteristic 2, the Weierstraß form of a non-supersingular elliptic curve E is given by the equation

$$y^2 + xy = x^3 + ax^2 + b \ , \tag{2.3}$$

where $a, b \in F$. [HMV04, §3.1]

The affine formula for point addition in characteristic 2 is as follows. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, $P \neq \pm Q$, be two points on the curve E. Then $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad \text{and} \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \ , \tag{2.4}$$

where $\lambda = (y_1 + y_2)/(x_1 + x_2)$. If $P \neq -P$, then $2P = (x_4, y_4)$, where

$$x_4 = \lambda^2 + \lambda + a \quad \text{and} \quad y_4 = x_1^2 + \lambda x_3 + x_3 \ , \tag{2.5}$$

where $\lambda = x_1 + y_1/x_1$.

Curves over characteristic 3 fields. In fields F of characteristic 3, the Weierstraß form of an elliptic curve E is given by the equation

$$y^2 = x^3 + ax^2 + c \ , \tag{2.6}$$

where $a, c \in F \setminus \{0\}$. [SW03]

The affine formula for point addition in characteristic 3 is as follows. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, $P \neq \pm Q$, be two points on the curve E. Then $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 - a - x_1 - x_2 \quad \text{and} \quad y_3 = \lambda(x_1 - x_3) - y_1 \ , \tag{2.7}$$

where $\lambda = (y_2 - y_1)/(x_2 - x_1)$. If $P \neq -P$, then $2P = (x_4, y_4)$, where

$$x_4 = \lambda^2 - a + x_1 \quad \text{and} \quad y_4 = \lambda(x_1 - x_4) - y_1 \ , \tag{2.8}$$

where $\lambda = ax_1/y_1$.

Supersingularity. The **trace** of an elliptic curve E over the field $\mathbb{F}_q$ is t, where $\#E(\mathbb{F}_q) = q + 1 - t$ is the number of points on the curve. If p divides t, where p is the characteristic of $\mathbb{F}_q$, then E is said to be **supersingular**, otherwise it is **non-supersingular**.

Additional formulæ. The point addition formulæ in (2.2)-(2.8) use **affine coordinates**; other coordinate systems are possible. The formulæ for λ all require an inversion, which can be computationally expensive in practice. This has motivated the development of formulæ using **projective coordinates**. In the ordinary projective case, a point is represented by three coordinates, $P = (X, Y, Z)$, with $x = X/Z$ and $y = Y/Z$. Denominators are used for all of the point additions and point doublings comprising a point multiplication, and only at the end is the inversion Z^{-1} computed to return the final result to affine coordinates.

Because λ is defined differently depending on whether or not $P = Q$, the formula for point addition differs from the formula for point doubling. This can lead to so-called side channel attacks, which are discussed in Chapter 6.

2.2 Cryptographic assumptions

The security of the protocols we consider in later sections of Part I of this thesis are based on various computational assumptions. Rather than having information-theoretic security, these protocols rely on the hardness of some underlying cryptographic problem for their security. In particular, the arguments for the security of the protocols relate the difficulty of breaking the security of the protocol, which is expressed as some task for an adversary to perform, to the difficulty of solving some underlying computational problem which is believed to be hard.

The notation we use to express the ability of an adversary to solve a particular task is the advantage notation, defined as follows. Examples of the advantage notation are used in the remainder of this section.

Notation 2.14 *Let* problem *be some computational problem, and let* params *be a set of parameters for that computational problem. Let $\mathcal{A}$ be a probabilistic algorithm. The* **advantage** *of $\mathcal{A}$ in solving* problem *with parameter set* params *is the probability that $\mathcal{A}$ can solve a randomly chosen instance of* problem *when the parameters are* params*, and is denoted by* $\mathsf{Adv}^{\text{problem}}_{\text{params}}(\mathcal{A})$.

Typically, we intend for the advantage to be negligibly small. A function $f : \mathbb{Z} \to [0, 1]$ is **negligible** if, for all $k > 0$, there exists a value $n_k > 0$ such that, for all $n > n_k$, $f(n) < n^{-k}$. In other words, if for sufficiently large n, $f(n)$ is smaller than the inverse of any polynomial in n.

We overload the advantage notation as follows to allow for the running time and other powers of the adversary to be more precisely specified.

Notation 2.15 *Let* problem *be some computational problem, and let* params *be a set of parameters for that computational problem. Let $t, q_1, \ldots$ be non-negative integers. Define*

$$\mathsf{Adv}^{\text{problem}}_{\text{params}}(t, q_1, \ldots) = \max_{\mathcal{A}} \left\{ \mathsf{Adv}^{\text{problem}}_{\text{params}}(\mathcal{A}) \right\} \tag{2.9}$$

where the maximum is taken over all probabilistic algorithms $\mathcal{A}$ running in time at most t and performing at most q_i operations of type i.

Typically, we will use the advantage notation of Notation 2.15 where we bound the number q_{ro} of queries to a random oracle (see Section 2.2.3) performed by the adversary, as well as other types of queries as specified by the formal security model.

2.2.1 Discrete logarithm problem

The Discrete Logarithm problem is an important problem that underlies many cryptographic protocols. It was first described by Diffie and Hellman in their seminal paper introducing public key cryptography [DH76].

Definition 2.16 *Let G be a finite cyclic group of order q and let g be a generator of G. Let $x \in \mathbb{Z}_q$. The* **Discrete Logarithm problem** *is as follows: given (g, g^x), find $x = \text{dlog}_g(g^x)$. Let $\mathcal{A}$ be a probabilistic algorithm. The* **DL advantage** *of $\mathcal{A}$ for G is defined to be*

$$\mathsf{Adv}^{\text{DL}}_{G,g}(\mathcal{A}) = \Pr\left(\mathcal{A}(G, g, q, g^x) = x\right) \quad , \tag{2.10}$$

where the probability is taken over uniformly random choices of x. The **Discrete Logarithm assumption** *for a group G is that, for all probabilistic adversaries running in time polynomial in $\log_2(q)$, $\mathsf{Adv}^{\text{DL}}_{G,g}(\mathcal{A})$ is negligible in a security parameter.*

There are two types of groups that are commonly used for cryptography for which the above problem is believed to be hard: the multiplicative group of integers modulo a large prime ($\mathbb{Z}_p^\times$), and additive groups of prime order of points on an elliptic curve.

The Discrete Logarithm problem can be solved a few different ways. The known algorithms for classical (non-quantum) computers cannot solve the Discrete Logarithm problem efficiently, while an algorithm is known for quantum computers that can solve the Discrete Logarithm problem efficiently, assuming a sufficiently large quantum computer.

The first set of techniques for computing discrete logarithms (such as Baby Step-Giant Step and Pollard's rho Method [Pol78]) work generically on the group generated by the generator g; in other words, they perform only group operations, and thus can be applied to groups of integers modulo a prime and to elliptic curve groups. If the cyclic group has order q, then these algorithms can compute the discrete logarithm of an element in this group using $O(\sqrt{q})$ group operations.

A second technique for computing discrete logarithms of integers modulo a large prime is known ([Gor93], based on the General Number Field Sieve). This technique takes advantage of the structure of the field of integers rather than the structure of the group, and thus is only applicable to the group of integers modulo a large prime. If arithmetic is being done modulo p, then the expected running time is $L_p(1/3, (64/9)^{1/3})$, where

$$L_x(v, c) = \exp\left((c + o(1))(\ln x)^v (\ln \ln x)^{1-v}\right) \quad . \tag{2.11}$$

Figure 2.1 describes the runtime for solving the Discrete Logarithm problem in various groups based on the above techniques, using the detailed analysis found in an ECRYPT report [BCC+06] (and based on the tables of [GB08]). In particular, the ECRYPT analysis uses a small modification, giving a runtime of

$$\exp\left((64/9)^{1/3}(\ln x)^{1/3}(\ln \ln x)^{2/3} - 14\right) \quad . \tag{2.12}$$

We have chosen to present the numbers from this ECRYPT report for convenience; however, there are several other analyses, a list of which can be found in [GB08].

Roughly speaking, if we want a security level of 2^{80}, meaning we expect an adversary to have to do at least 2^{80} steps of computation before having a reasonable chance of solving the problem, we need to use integers modulo a prime of at least 1248 bits (and working in a group of order at least 2^{160}), or an elliptic curve of at least 160 bits.

If a large scale quantum computer is available, then the Discrete Logarithm problem can be solved efficiently. In particular, there is a quantum algorithm running in time $O(\lceil \log_2(q) \rceil^2)$ that can compute discrete logarithms in any finite abelian group. This discrete logarithm algorithm for integers modulo a prime was described by Shor [Sho94, Sho97] and was extended to arbitrary finite abelian groups by others [Kit95, CEMM98] and is described well in [NC00, §5.4.2]. The

Runtime	$\mathbb{Z}_p^\times$		Elliptic curve
	$\log_2(p)$	Group order	Group order
2^{72}	1008	144	144
2^{80}	1248	160	160
2^{128}	3248	256	256
2^{256}	15424	512	512

Figure 2.1: Runtimes for solving Discrete Logarithm problem in various groups

special case of Shor's algorithm applied to elliptic curves has been studied in detail [PZ03, KZ04, CMMP08]. However, quantum computers of sufficiently large scale for computing discrete logarithms in groups used for cryptographic purposes today appear infeasible for at least 20 years, and thus these number-theoretic building blocks are still of interest for some immediate and short-term cryptographic problems.

2.2.2 Diffie-Hellman problems

The Computational Diffie-Hellman problem is one of the most widely used computational assumptions in cryptography. It was first described by Diffie and Hellman in 1976 [DH76].

Definition 2.17 *Let G be a finite cyclic group of order q and let g be a generator of G. Let $x, y \in \mathbb{Z}_q$. Define $\mathsf{DH}(g^x, g^y) = g^{xy}$. The* **Computational Diffie-Hellman problem** *is as follows: given (g, g^x, g^y), find $\mathsf{DH}(g^x, g^y)$. Let $\mathcal{A}$ be a probabilistic algorithm. The* **CDH advantage** *of $\mathcal{A}$ for G is defined to be*

$$\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(\mathcal{A}) = \Pr\left(\mathcal{A}(G, g, q, g^x, g^y) = \mathsf{DH}(g^x, g^y)\right) \ , \tag{2.13}$$

where the probability is taken over uniformly random choices of x and y. The **Computational Diffie-Hellman assumption** *for a group G is that, for all probabilistic adversaries running in time polynomial in $\log_2(q)$, $\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(\mathcal{A})$ is negligible in a security parameter.*

Clearly, if there exists an algorithm to solve the Discrete Logarithm problem efficiently, then this algorithm can be used to solve the Computational Diffie-Hellman problem. This means that the following relation holds:

Lemma 2.18 *Let G be a finite cyclic group of order q and let g be a generator of G. Let t_{exp} be the time required to perform an exponentiation in the group G. Then*

$$\mathsf{Adv}^{\mathrm{DL}}_{G,g}(t) \leq \mathsf{Adv}^{\mathrm{CDH}}_{G,g}(t + t_{\mathsf{exp}}) \ . \tag{2.14}$$

A variant of the Computational Diffie-Hellman problem is the List Computational Diffie-Hellman problem, which allows the output of the adversary to be a

list of group elements, only one of which need be the Diffie-Hellman value. The List Computational Diffie-Hellman problem was explicitly stated by [Sho06] but was used elsewhere earlier (for example, [Mac02]).

Definition 2.19 *Let G be a finite cyclic group of order q and let g be a generator of G. Let $x, y \in \mathbb{Z}_q$ and let $\ell \in \{1, 2, \dots\}$. The* **List Computational Diffie-Hellman problem** *is as follows: given (g, g^x, g^y), output a set $\mathcal{S}$ containing at most ℓ elements such that $\mathsf{DH}(g^x, g^y) \in \mathcal{S}$. Let $\mathcal{A}$ be a probabilistic algorithm outputting a set of at most ℓ elements. The* **List-CDH advantage** *of $\mathcal{A}$ for G is defined to be*

$$\mathsf{Adv}^{\mathrm{LCDH}}_{G,g}(\mathcal{A}, \ell) = \Pr\left(\mathsf{DH}(g^x, g^y) \in \mathcal{A}(G, g, q, g^x, g^y)\right) \ , \tag{2.15}$$

where the probability is taken over uniformly random choices of x and y. The **List Computational Diffie-Hellman assumption** *for a group G is that, for all probabilistic adversaries running in time polynomial in $\log_2(q)$ (and thus outputting a set of length ℓ bounded by a polynomial in $\log_2(q)$), $\mathsf{Adv}^{\mathrm{LCDH}}_{G,g}(\mathcal{A}, \ell)$ is negligible in a security parameter.*

Clearly, if there exists an algorithm to solve the Computational Diffie-Hellman problem efficiently, then this algorithm can be used to solve the List Computational Diffie-Hellman problem. Moreover, an algorithm that solves List Computational Diffie-Hellman can be used to probabilistically solve Computational Diffie-Hellman. This means that the following relations hold:

Lemma 2.20 *Let G be a finite cyclic group of order q, let $\ell \in \{1, 2, \dots\}$, and let g be a generator of G. Then*

$$\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(t) = \mathsf{Adv}^{\mathrm{LCDH}}_{G,g}(t, 1) \leq \mathsf{Adv}^{\mathrm{LCDH}}_{G,g}(t, \ell) \tag{2.16}$$

and

$$\mathsf{Adv}^{\mathrm{LCDH}}_{G,g}(t, \ell) \leq \ell \cdot \mathsf{Adv}^{\mathrm{LCDH}}_{G,g}(t, 1) \ . \tag{2.17}$$

Whereas CDH and LCDH require that the adversary find the Diffie-Hellman value, the Decisional Diffie-Hellman problem only requires that an adversary distinguish the Diffie-Hellman value from a random string. It was first stated by Diffie and Hellman [DH76] and a good survey of research related to the problem is given by Boneh [Bon98].

Definition 2.21 *Let G be a finite cyclic group of order q and let g be a generator of G. Let $x, y, z \in \mathbb{Z}_q$. The* **Decisional Diffie-Hellman problem** *is as follows: given (g, g^x, g^y, g^z), determine if $z = xy$ (or equivalently, if $g^z = \mathsf{DH}(g^x, g^y)$). Let $\mathcal{A}$ be a probabilistic algorithm. The* **DDH advantage** *of $\mathcal{A}$ for G is defined to be*

$$\mathsf{Adv}^{\mathrm{DDH}}_{G,g}(\mathcal{A}) = \left|\Pr\left(\mathcal{A}(G, g, q, g^x, g^y, g^{xy}) = 1\right) - \Pr\left(\mathcal{A}(G, g, q, g^x, g^y, g^z) = 1\right)\right| \ , \tag{2.18}$$

where the probability is taken over uniformly random choices of x, y, and z. The **Decisional Diffie-Hellman assumption** *for a group G is that, for all probabilistic adversaries running in time polynomial in $\log_2(q)$, $\mathsf{Adv}^{\mathrm{DDH}}_{G,g}(\mathcal{A})$ is negligible in a security parameter.*

Clearly, if there exists an algorithm to solve the Computational Diffie-Hellman problem efficiently, then this algorithm can be used to solve the Decisional Diffie-Hellman problem. This means that the following relation holds:

Lemma 2.22 *Let G be a finite cyclic group of order q and let g be a generator of G. Then*

$$\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(t) \leq \mathsf{Adv}^{\mathrm{DDH}}_{G,g}(t) \ . \tag{2.19}$$

One other variant is the Gap Diffie-Hellman problem, which requires that the adversary solve a CDH problem given access to a DDH oracle. The existence of groups satisfying this condition was noted by Joux and Nguyen [JN01] and this problem was first stated by Okamoto and Pointcheval [OP01].

Definition 2.23 *Let G be a finite cyclic group of order q and let g be a generator of G. Let $x, y \in \mathbb{Z}_q$. The* **Gap Diffie-Hellman problem** *is as follows: given (g, g^x, g^y), find $\mathsf{DH}(g^x, g^y)$ given an oracle $\mathcal{O}$ that solves the Decisional Diffie-Hellman problem in G. Let $\mathcal{A}$ be a probabilistic algorithm. The* **GDH advantage** *of $\mathcal{A}$ for G is defined to be*

$$\mathsf{Adv}^{\mathrm{GDH}}_{G,g}(\mathcal{A}) = \Pr\left(\mathcal{A}(G, g, q, \mathcal{O}, g^x, g^y) = \mathsf{DH}(g^x, g^y)\right) \ , \tag{2.20}$$

where the probability is taken over uniformly random choices of x and y. The **Gap Diffie-Hellman assumption** *for a group G is that, for all probabilistic adversaries running in time polynomial in $\log_2(q)$, $\mathsf{Adv}^{\mathrm{GDH}}_{G,g}(\mathcal{A})$ is negligible.*

Boneh, Lynn, and Shacham [BLS01, §3.2] propose the use of certain supersingular elliptic curves of the form $y^2 = x^3 + 2x \pm 1$ over the field $\mathbb{F}_{3^\ell}$ to instantiate a Gap Diffie-Hellman group and give parameter size guidance. The DDH oracle is provided by an algorithm that makes use of the Weil pairing or the Tate pairing [FMR99].

Clearly, if there exists an algorithm to solve the Computational Diffie-Hellman problem efficiently, then this algorithm can be used to solve the Gap Diffie-Hellman problem. This means that the following relation holds:

Lemma 2.24 *Let G be a finite cyclic group of order q and let g be a generator of G. Then*

$$\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(t) \leq \mathsf{Adv}^{\mathrm{GDH}}_{G,g}(t) \ . \tag{2.21}$$

By combining Lemmas 2.18, 2.20, 2.22, and 2.24, we can see the relationships between the various computational problems discussed so far, and where being able to solve one implies the ability to solve another.

$$\begin{array}{ccccc} \text{solve DL} & \Longrightarrow & \text{solve CDH} & \Longleftrightarrow & \text{solve List-CDH} \\ & & \Downarrow \quad \Downarrow & & \\ & & \text{solve DDH} \quad \text{solve GDH} & & \end{array}$$

The best known techniques for solving CDH, List-CDH, DDH, and GDH in groups for which the respective problems are believed to be hard involve first computing a discrete logarithm, and thus the conjectured difficulty of solving these problems is often taken to be equivalent to the difficulty of solving the Discrete Logarithm problem. CDH is in fact equivalent to the Discrete Logarithm problem for groups with smooth order (every prime factor of the order is polynomial in the logarithm of the order) [dB90, MW99].

2.2.3 Random oracle model

The random oracle model, described by Bellare and Rogaway [BR04] is a useful tool in cryptographic arguments in which we assume that a hash function as used in a protocol behaves as a random function. This is a powerful assumption about which there is some debate ([KM04, §6], [Gol06]), but remains a widely used and useful tool in cryptographic research. It was recently shown that there is a connection between the random oracle model and the ideal cipher model [CPS08a].

Definition 2.25 *A* **random oracle** R *is a map from* $\{0,1\}^*$ *to* $\{0,1\}^\infty$ *chosen by selecting each bit of* $R(x)$ *uniformly and independently at random for each* x*. A* **random hash function** $H : \{0,1\}^* \to \{0,1\}^k$ *is chosen in the same way.*

A random hash function can be constructed from an oracle "on the fly": when called with input x, if the oracle has not seen x before, it picks the output y uniformly at random, returns y, and stores (x, y) in a table; if x is already in the table, then it returns the corresponding entry y.

Many independent random hash functions $H_0, H_1, H_2, \ldots$ can be constructed from a single random hash function H, for example by setting $H_\ell(x) = H(\ell, x)$. There exist efficient, well-known constructions for returning integers instead of bit strings when necessary (for example, [BK07, §B.5.1]).

2.2.4 Digital signatures

Some protocols also make use of digital signature schemes which are existentially unforgeable under chosen message attacks [GMR88], [Sti02, §7.2].

A **signature scheme** $\mathcal{S}$ is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, where Gen is a key generation algorithm such that $\mathsf{Gen}(1^\kappa)$ returns a private key / public key pair (v, V) for a security parameter κ, Sign is a signature generation algorithm such that $\mathsf{Sign}_v(m)$ returns a signature of the message m under the private key v, and Verify is a verification algorithm such that $\mathsf{Verify}_V(m, s)$ returns true if s is a valid signature of m under public key V and false otherwise.

Definition 2.26 *A* **chosen message attack** *by a forger* $\mathcal{F}$ *on a signature scheme* $\mathcal{S}$ *is an attack in which* $\mathcal{F}$ *is able to supply messages* $m_1, m_2, \ldots$ *of its choice to a signing oracle and obtain valid signatures* $s_1 = \mathsf{Sign}_v(m_1), s_2 = \mathsf{Sign}_v(m_2), \ldots$.

Definition 2.27 *An* **existential forgery** *by a forger $\mathcal{F}$ on a signature scheme $\mathcal{S}$ is an attack in which $\mathcal{F}$ is able to create a valid signature s for at least one message m, so that $\mathsf{Verify}_V(m, s) = \mathsf{true}$ and so that m was not previously signed by any signing oracle for a private key corresponding to V.*

Definition 2.28 *Let $\mathcal{S}$ be a signature scheme with security parameter κ. A forger $\mathcal{F}$ is a probabilistic algorithm that is given a public key V and whose goal is to construct an existential forgery using a chosen message attack. The* **eu-cma advantage** *of $\mathcal{F}$ for $\mathcal{S}$ is defined to be*

$$\mathsf{Adv}^{\text{eu-cma}}_{\mathcal{S},\kappa}(\mathcal{F}) = \Pr(\mathcal{F} \textit{ succeeds}) \ . \tag{2.22}$$

A signature scheme $\mathcal{S}$ is **existentially unforgeable under chosen message attacks** *(***eu-cma***) if, for any probabilistic polynomial time algorithm $\mathcal{F}$, $\mathsf{Adv}^{\text{eu-cma}}_{\mathcal{S},\kappa}(\mathcal{F})$ is negligible.*

The RSA-PSS signature scheme [BR96] has been shown to be existentially unforgeable under chosen message attacks assuming the RSA problem is hard. Other popular signature schemes, such as DSA and ECDSA, are not known to be existentially unforgeable under chosen message attacks. However, a variant of ECDSA has been shown to be secure in the generic group model which suggests some confidence in security of ECDSA [Bro04].

Chapter 3

Password-Authenticated Key Exchange

Contents

The basic problem of password-authenticated key exchange[1] is as follows. Two parties wish to authenticate each other and agree upon a secret key, but they only share a short text string, not a long cryptographic secret. The client cannot simply tell the server her password as proof of identity because a malicious "phishing" server could then steal the password and impersonate the client at the real server. A password-authenticated key exchange protocol allows each party to authenticate the other's identity based solely on their knowledge of a short password, without revealing any useful information about the password to any other party; moreover, the two parties can also agree on a shared key suitable for other cryptographic purposes such as bulk encryption.

[1] Although the term "password-authenticated key agreement" might be more appropriate for the protocols we consider, the term "password-authenticated key exchange" is more often used in the literature, and we make use of the latter for the rest of this thesis.

3.1 Literature review

Password-authenticated key exchange protocols have been extensively studied in the literature. A large list of papers concerning password-based key exchange is maintained by Jablon [Jab08].

Password-authenticated key exchange was first introduced by Bellovin and Merritt in 1992 [BM92] in the form of encrypted key exchange (EKE), a protocol in which the client and server shared the plaintext password and exchanged encrypted information to allow them to derive a shared session key. A later variant by Bellovin and Merritt, Augmented EKE (A-EKE) [BM94], removed the requirement that the server have the plaintext password, instead having a one-way function of the password. The former is called a *symmetric* password-based protocol, because both client and server share the same plaintext password, whereas the latter is called *asymmetric* or *verifier-based*.

A number of formal models for the security of password-authenticated key exchange protocols have been introduced, including one by Boyko, Mackenzie, and Patel [BMP00a] (which extends the simulation model for authenticated key exchange by Shoup [Sho99b]), one by Bellare, Pointcheval, and Rogaway [BPR00], and the three-party setting of Abdalla, Fouque, and Pointcheval [AFP05]. The Bellare-Pointcheval-Rogaway model has been used extensively and forms the basis of our security analyses. A description of the model is given in Section 3.2.

Wu [Wu98] introduced an asymmetric (verifier-based) protocol called the Secure Remote Password (SRP) protocol (called SRP-3) in 1998; after a minor flaw was found [Mac01] in the initial version, a revised version (SRP-6) was described in 2002 [Wu02]; an elliptic curve version also exists [Wan01]. A description of SRP is given in Section 3.3.1. However, there is no argument for the security of SRP in any of the formal models listed above, only a proof that the key exchange portion of SRP-3 is secure against passive adversaries. Nonetheless, SRP is of particular interest because it is efficient and more importantly because its message flow allows it to fit inside the pre-defined message flow of the existing network protocol SSL/TLS. As such, it has been standardized as an informational RFC [TWMP07].

The SOKE protocol [ABC+06] does have formal security arguments in the Bellare-Pointcheval-Rogaway model and was designed explicitly to fit into the SSL/TLS handshake protocol, but it is not a verifier-based protocol; there are no verifier-based protocols with formal security arguments that fit into the existing message flow of the SSL/TLS handshake protocol.

Other recent work has focused on protocols with formal security claims and proofs in the standard model (that is, not assuming random oracles), but our work in the rest of this thesis assumes the random oracle model.

Two recent protocols of interest and which inspire some of our protocol designs are PAK [BMP00a, Mac02] and PAK-Z+ [GMR05], which we describe in Sections 3.3.2 and 3.3.3, respectively. PAK is a symmetric protocol whereas PAK-Z+

is an asymmetric protocol: both have a similar structure but use authentication secrets of different natures. Both have been shown to be secure in the Bellare-Pointcheval-Rogaway model. The technique used to show the security of PAK-Z+ is a specialization of the same authors' later Ω-method [GMR06] for converting a symmetric password-authenticated key exchange protocol into an asymmetric one.

A one-time password-authenticated key exchange protocol called OPKeyX was suggested by Abdalla, Chevassut, and Pointcheval [ACP05a]: their scheme uses a sequence of one-time passwords generated by iterative hashing of a seed.

3.2 Formal model

The formal model we use for analyzing the security of password-authenticated key exchange is that of Bellare, Pointcheval, and Rogaway [BPR00], as modified by Gentry, MacKenzie, and Ramzan [GMR05]; we refer to this as the BPR model for password-authenticated key exchange.

The basic idea of the model is as follows. We envision a number of honest, interacting parties connected by communication links. The communication links are controlled by an adversary. The adversary can observe messages sent by the parties, which models passive eavesdropping. The adversary can also modify, delete, or reorder messages sent by the parties, and can inject her own messages as well, which models active interference in the network. In addition to controlling the communication links, the adversary is also allowed to learn certain information from parties, which models partial compromise of private information.

To analyze security in this model, we define one or more tasks — games — that the adversary needs to accomplish in order to have broken the security of the protocol, provided that the adversary has not compromised particular interacting parties so much so to make the tasks trivial.

We now describe the BPR model more precisely.

3.2.1 Model setup

Participants. Each interacting party is either a **client** or a **server**, is identified by a unique fixed length string, and the identifier is a member of either the set $\mathsf{Clients}$ or $\mathsf{Servers}$, respectively, with $\mathsf{Parties} = \mathsf{Clients} \dot{\cup} \mathsf{Servers}$.

Passwords. Authentication secrets are short strings selected uniformly at random from an appropriate set. Passwords are chosen from the set $\mathsf{Passwords}$. Typically, this set is large, but not so large that a brute-force search is infeasible. For example, the author's keyboard has 95 distinct printable characters on it: if $\mathsf{Passwords}$ consists of all strings of length 8 from this set of characters, then $|\mathsf{Passwords}| = 95^8 \doteq 2^{52.6}$. In other words, such passwords only have about 52 bits

of entropy, which is much smaller than what is commonly employed for cryptographically large secrets, usually having at least 80 or 128 bits of entropy.

For the rest of this thesis, we assume passwords are chosen uniformly at random from the set $\mathsf{Passwords}$. In practice, this assumption is often unmet. When passwords are generated by computers (for example, by a password helper program or by a hardware password token as discussed in Section 4.3.3), it is easier to ensure passwords are uniformly distributed. Verheul [Ver07] examines the different types of attacks against computer-generated passwords and considers a variety of entropy measures for password security.

Human-generated passwords are almost certainly not going to be uniformly distributed. Many studies have reported on poor password distributions; in one analysis of cracked Kerberos passwords by Wu [Wu99, §3.1], 74% of passwords contained no digits, 96% of passwords contained no non-alphanumeric symbols, and 86% of passwords could be typed without using the shift key. Wagner and Goldberg [WG00, §7] introduce *passphrase-based cryptography* where keying material is derived from human-generated passphrases and thus likely to be non-uniform and apply it to the case of Unix password hashing. Non-uniform passphrases have been considered only a few times in the password-authenticated key exchange literature, in the context of universally composable password-authenticated key exchange [CHK+05] and in a brief discussion by Katz *et al.* [KOY, §2.2.1].

For each client-server pair $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$, an authentication secret exists: there is a long-term password $\mathsf{pw}_{\hat{C},\hat{S}} \in \mathsf{Passwords}$, and a corresponding $\mathsf{pw}_{\hat{S}}[\hat{C}]$ which is some transformation of $\mathsf{pw}_{\hat{C},\hat{S}}$; $\mathsf{pw}_{\hat{S}}[\hat{C}]$ is stored on the server $\hat{S}$. If $\mathsf{pw}_{\hat{S}}[\hat{C}] = \mathsf{pw}_{\hat{C},\hat{S}}$ (in other words, if the transformation is the identity), then this is called the **symmetric model** or **non-verifier-based model** of password-authenticated key exchange. Alternatively, it may be that $\mathsf{pw}_{\hat{S}}[\hat{C}] \neq \mathsf{pw}_{\hat{C},\hat{S}}$, which is called the **asymmetric model** or **verifier-based model** of password-authenticated key exchange.

In the rest of this work, we actually relax the distinction between the symmetric model and the asymmetric model. If the server's value $\mathsf{pw}_{\hat{S}}[\hat{C}]$ is the same as or a trivial or reversible transformation of the client's value $\mathsf{pw}_{\hat{C},\hat{S}}$, then we say that we are in the symmetric model. We are trying to capture the following notion: in an asymmetric (verifier-based) protocol, the value $\mathsf{pw}_{\hat{S}}[\hat{C}]$ stored on the server should not be enough to immediately impersonate a client. There may be protocols where the value stored on the server is not identical to what is stored on the client, but is enough to impersonate the client with little extra work, and we wish to include this in the notion of symmetric (non-verifier-based) protocols.

Execution of the protocol. The protocol is, formally, a probabilistic algorithm on strings, and specifies how each party responses to messages. During execution, each party may have multiple instances of the protocol running. Each instance i of a party $\hat{U} \in \mathsf{Parties}$ is treated as an **oracle** denoted by $\Pi_i^{\hat{U}}$, modeling some procedure run by party $\hat{U}$.

In a protocol, there is a sequence of messages, called **flows**, starting with a flow from the client instance, responded to by a server instance, and so on. At any time, parties may **abort** the protocol, or, after some fixed number of flows, may terminate and **accept**, producing some output. For authenticated key exchange protocols, the output of an accepting protocol instance for a party consists of a session key sk, a **partner identifier** pid which identifies the party with which the present party thinks it has just exchanged a key, and a **session identifier** sid, which uniquely identifies the conversation. The session identifier and partner identifier need not be kept secret, and in fact are typically based on information in messages exchanged publicly.

Two instances $\Pi_i^{\hat{C}}$ and $\Pi_j^{\hat{S}}$ are said to be **partnered** if they both accept, hold outputs $(\mathsf{pid}, \mathsf{sid}, \mathsf{sk})$ and $(\mathsf{pid}', \mathsf{sid}', \mathsf{sk}')$ with $\mathsf{pid} = \hat{S}$, $\mathsf{pid}' = \hat{C}$, $\mathsf{sid} = \mathsf{sid}'$, and $\mathsf{sk} = \mathsf{sk}'$, and no other instance accepts with session identifier equal to sid.

Powers of the adversary. While the protocol specification determines how parties respond to inputs from the environment, the environment is considered to be controlled by the **adversary**, which is formally a probabilistic algorithm that issues **queries** to parties' oracle instances and receives responses. There are in general three classes of queries: (1) queries that model the transmission of messages across communication links, (2) queries that model the adversary learning certain information by compromising a party in some way, and (3) queries that have been added to allow the formulation of a game for the adversary to win.

For a protocol P, we define the following queries that the adversary can issue (where clear by the setting, we may omit the subscript P).

The first two queries, Send and $\mathsf{Execute}$, model the transmission of messages across communication links.

$\mathsf{Send}_P(\hat{U}, i, M)$: Sends message M to user instance $\Pi_i^{\hat{U}}$, which faithfully performs the appropriate portion of protocol P based on its current state and the message M, updates its state as appropriate, and returns any resulting messages. If the oracle accepts, then this fact, as well as the session identifier and partner identifier, are returned to the adversary as well. The message $M =$ "start" can be used to cause $\hat{U}$ to send the initial message in the protocol.

$\mathsf{Execute}_P(\hat{C}, i, \hat{S}, j)$: Causes client instance $\Pi_i^{\hat{C}}$ and server instance $\Pi_j^{\hat{S}}$ to faithfully execute protocol P and returns the resulting transcript, assuming that the oracles $\Pi_i^{\hat{C}}$ and $\Pi_j^{\hat{S}}$ have not been used before. The $\mathsf{Execute}$ query models passive eavesdropping by the adversary. This query may seem redundant as it could be implemented solely using a sequence of Send queries, but it is convenient to have a query that explicitly models passive attacks.

The next three queries, $\mathsf{RevealSessionKey}$, $\mathsf{RevealPWC}$, and $\mathsf{RevealPWS}$, model the adversary's ability to compromise certain pieces of information held by parties.

$\mathsf{RevealSessionKey}_P(\hat{U}, i)$: If user instance $\Pi_i^{\hat{U}}$ has accepted, then returns session

key sk held by $\Pi_i^{\hat{U}}$. This query is used to model the notion that the compromise of one session key should not be damaging to other sessions.

$\mathsf{RevealPWC}_P(\hat{C}, \hat{S})$: Returns the password $\mathsf{pw}_{\hat{C},\hat{S}}$ of client $\hat{C}$ with server $\hat{S}$. This query is used to model compromise of the client's password, for example by the use of spyware installed on the client's computer. This corresponds to the weak corruption model of the BPR formal model, in which the adversary may learn, but not alter, the password information.[2]

$\mathsf{RevealPWS}_P(\hat{S}, \hat{C})$: Returns the transformed password $\mathsf{pw}_{\hat{S}}[\hat{C}]$ of client $\hat{C}$ on server $\hat{S}$. This query is used to model the compromise of the server's password database, which, in the verifier-based model, may not have the same effect as the compromise of the client's password.[3]

Additionally, the use of a hash function may be modeled as an oracle query when working in the random oracle model, but we leave that out of this section as it is not needed for all authenticated key exchange protocols.

3.2.2 Session key security

Informally, a session key output by a key exchange protocol is considered secure if it is indistinguishable from a random string chosen from the same distribution as the possible session keys.

In order to model session key security, the BPR model defines an additional query that is used to define the task of the adversary, which is to distinguish a session key from a random string for an uncompromised session. The Test query, which does not correspond to any real-world operation of the adversary, is used to define the game that the adversary must win in order for the protocol to be considered broken.

$\mathsf{Test}_P(\hat{U}, i)$: If user instance $\Pi_i^{\hat{U}}$ has accepted, then this query causes the following to happen: choose $b \in_R \{0, 1\}$; if $b = 1$, then return the session key of $\Pi_i^{\hat{U}}$, otherwise return a random string chosen from the same distribution as the session key (e.g., a random string of the same length). This query may only be asked once.

Freshness. Some of the queries the adversary is allowed to call are very powerful and reveal important session information, including, for the $\mathsf{RevealSessionKey}$ query, the session key. In order to describe which sessions ought to be hard for the adversary to win the Test game, we use the notion of freshness. Informally, if the adversary issues the query $\mathsf{Test}(\hat{U}, i)$, then that oracle should be considered unfresh

[2]The $\mathsf{RevealPWC}$ query is usually called $\mathsf{Corrupt}$ in the BPR model but, since we are only modeling weak corruption, we use the terminology "reveal" instead of "corrupt" to be more suggestive of the actual effect of this query.

[3]The $\mathsf{RevealPWS}$ query did not exist in the original BPR model and was added by Gentry, MacKenzie, and Ramzan [GMR05].

if the adversary has previously queried for some passwords related to that oracle or has revealed the session key for that oracle.

More formally, an instance $\Pi_i^{\hat{U}}$ with partner id $\hat{U}'$ is **fresh** (with forward-secrecy)[4] if and only if none of the following events occur:

1. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}, i)$;
2. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}', j)$, where $\Pi_j^{\hat{U}'}$ is the partner instance of $\Pi_i^{\hat{U}}$;
3. if $\hat{U} \in \mathsf{Clients}$, the adversary has issued either the query $\mathsf{RevealPWC}(\hat{U}, \hat{U}')$ or the query $\mathsf{RevealPWS}(\hat{U}', \hat{U})$ before the Test query, and has issued the query $\mathsf{Send}(\hat{U}, i, M)$ for some string M;
4. if $\hat{U} \in \mathsf{Servers}$, the adversary has issued the query $\mathsf{RevealPWC}(\hat{U}, \hat{U}')$ before the Test query, and has issued the query $\mathsf{Send}(\hat{U}, i, M)$ for some string M.

Adversary's goals. The goal of an adversary is to guess the bit b used in the Test query of a fresh session. This corresponds to the ability of an adversary to distinguish the session key from among the distribution of possible session keys.

Let $\mathcal{A}$ be a probabilistic algorithm. Let $\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh instance $\Pi_i^{\hat{U}}$ that has accepted and $\mathcal{A}$ eventually outputs a bit b', where $b' = b$ and b is the randomly selected bit in the Test query. The **ake advantage** of $\mathcal{A}$ attacking P is defined to be

$$\mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) = \left| 2 \Pr\left(\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})\right) - 1 \right| \ . \tag{3.1}$$

The following lemma establishes a basic fact about the ake advantage.

Lemma 3.1 *For any protocols P and P', any probabilistic algorithm $\mathcal{A}$, and any $\epsilon \geq 0$,*

$$\Pr\left(\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})\right) = \Pr\left(\mathsf{Succ}_{P'}^{\mathrm{ake}}(\mathcal{A})\right) + \epsilon \quad \Longleftrightarrow \quad \mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) = \mathsf{Adv}_{P'}^{\mathrm{ake}}(\mathcal{A}) + 2\epsilon \ . \tag{3.2}$$

Proof. By (3.1), we have that

$$\Pr\left(\mathsf{Succ}_P^{\mathrm{ake}}(\mathcal{A})\right) = \frac{1}{2}\mathsf{Adv}_P^{\mathrm{ake}}(\mathcal{A}) + \frac{1}{2} \ . \tag{3.3}$$

[4]The original BPR model [BPR00, §3] included two notions of freshness: without and with forward secrecy. **Forward secrecy** means that the session key should remain secure even if the long-term secret (the password) is later disclosed. Most provably secure protocols in the literature achieve forward secrecy, so we only consider that notion in the rest of this thesis.

Thus,

$$\Pr\left(\mathsf{Succ}^{\mathrm{ake}}_{P}(\mathcal{A})\right) = \Pr\left(\mathsf{Succ}^{\mathrm{ake}}_{P'}(\mathcal{A})\right) + \epsilon \tag{3.4}$$

$$\iff \quad \frac{1}{2}\mathsf{Adv}^{\mathrm{ake}}_{P}(\mathcal{A}) + \frac{1}{2} = \frac{1}{2}\mathsf{Adv}^{\mathrm{ake}}_{P'}(\mathcal{A}) + \frac{1}{2} + \epsilon \tag{3.5}$$

$$\iff \quad \mathsf{Adv}^{\mathrm{ake}}_{P}(\mathcal{A}) = \mathsf{Adv}^{\mathrm{ake}}_{P'}(\mathcal{A}) + 2\epsilon \ . \tag{3.6}$$

□

We see from Lemma 3.1 that the situation in which the session key is perfectly secure — that is, when the probability of distinguishing it from a random string is 1/2 — corresponds precisely to the adversary's ake advantage being 0.

As in Notation 2.15, we overload the Adv notation to allow us to more precisely describe the running time of the adversary. Let t, q_{se}, q_{ex}, and q_{ro} be non-negative integers. Then we define

$$\mathsf{Adv}^{\mathrm{ake}}_{P}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) = \max_{\mathcal{A}}\{\mathsf{Adv}^{\mathrm{ake}}_{P}(\mathcal{A})\} \tag{3.7}$$

where the maximum is taken over all probabilistic algorithms $\mathcal{A}$ running in time at most t, making at most q_{se} queries of type Send, q_{ex} queries of type $\mathsf{Execute}$, and q_{ro} queries to any random oracles.

Security. We say that a protocol P is a **secure password-authenticated key exchange protocol** if, for all probabilistic algorithms $\mathcal{A}$ running in polynomial time and making at most q_{se} queries of type Send, there exists a constant δ and a negligible (in the security parameter λ) ϵ such that

$$\mathsf{Adv}^{\mathrm{ake}}_{P}(\mathcal{A}) \leq \frac{\delta q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \ . \tag{3.8}$$

Intuitively, this notion of security says that, except with probability ϵ (which is usually a function of the probability of solving a hard computational problem under certain constraints), any polynomially bounded adversary can do little better than doing an online dictionary attack on the password and can gain no advantage by doing an offline dictionary attack.

3.2.3 Authentication

There are two directions of authentication that are desired in password-authenticated key exchange: the client should authenticate the server's identity, and the server should authenticate the client's identity.

For a protocol P, an adversary violates **client-to-server authentication** if some server oracle terminates and accepts but has no partner oracle. For a fixed adversary $\mathcal{A}$, the **c2s advantage** is the probability of this event and is denoted by $\mathsf{Adv}^{\mathrm{c2s}}_{P}(\mathcal{A})$.

Similarly, an adversary violates **server-to-client authentication** if some client oracle terminates and accepts but has no partner oracle. The **s2c advantage** is the probability of this event and is denoted by $\mathsf{Adv}_P^{\mathrm{s2c}}(\mathcal{A})$.

Finally, an adversary violates **mutual authentication** if some oracle terminates and accepts but has no partner oracle. Violation of mutual authentication is equivalent to violation of either client-to-server authentication or server-to-client authentication (or both). The **ma advantage** is the probability of this event and is denoted by $\mathsf{Adv}_P^{\mathrm{ma}}(\mathcal{A})$.

As in Section 3.2.2, we overload the Adv notation to allow us to more precisely describe the running time of the adversary. Let t, q_{se}, q_{ex}, and q_{ro} be non-negative integers. Then we define

$$\mathsf{Adv}_P^{\mathrm{c2s}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) = \max_{\mathcal{A}}\{\mathsf{Adv}_P^{\mathrm{c2s}}(\mathcal{A})\} \tag{3.9}$$

$$\mathsf{Adv}_P^{\mathrm{s2c}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) = \max_{\mathcal{A}}\{\mathsf{Adv}_P^{\mathrm{s2c}}(\mathcal{A})\} \tag{3.10}$$

$$\mathsf{Adv}_P^{\mathrm{ma}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) = \max_{\mathcal{A}}\{\mathsf{Adv}_P^{\mathrm{ma}}(\mathcal{A})\} \tag{3.11}$$

where each maximum is taken over all probabilistic algorithms $\mathcal{A}$ running in time at most t, making at most q_{se} queries of type Send, q_{ex} queries of type $\mathsf{Execute}$, and q_{ro} queries to any random oracles.

3.2.4 Security against passive adversaries

The notions of session key security in Section 3.2.2 and authentication in Section 3.2.3 use the full power of an active adversary's queries of Section 3.2.1, and indeed represent very strong notions of security and authentication.

However, as we shall see in the next section, there exist practical protocols for which there are no formal security arguments for the strong security notions described above. Thus, it can be useful to relax the security model in an attempt to describe at least some of the security properties of these protocols.

One such relaxation is to analyze the security of the session key against passive eavesdroppers. In such a setting, we allow an adversary to use all of the queries of Section 3.2.1 except for the Send query. In other words, the adversary may use the following queries: $\mathsf{Execute}$, $\mathsf{RevealSessionKey}$, $\mathsf{RevealPWC}$, $\mathsf{RevealPWS}$, and Test.

We define a new notion of freshness as follows. An instance $\Pi_i^{\hat{U}}$ with partner id $\hat{U}'$ is **fresh against passive adversaries** (with forward secrecy) if and only if none of the following events occur:

1. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}, i)$;
2. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}', j)$, where $\Pi_j^{\hat{U}'}$ is the partner instance of $\Pi_i^{\hat{U}}$;

3. the adversary has issued a Send query;

4. if $\hat{U} \in$ Clients, the adversary has issued either the query RevealPWC$(\hat{U}, \hat{U}')$ or the query RevealPWS$(\hat{U}', \hat{U})$ before the Test query;

5. if $\hat{U} \in$ Servers, the adversary has issued either the query RevealPWS$(\hat{U}, \hat{U}')$ or the query RevealPWC$(\hat{U}', \hat{U})$ before the Test query.

As before, the adversary's goal is to guess the bit b used in the Test query of a fresh-against-passive-adversaries session.

Let $\mathcal{A}$ be a probabilistic algorithm. Let $\mathsf{Succ}_P^{\text{p-ke}}(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh-against-passive-adversaries instance $\Pi_i^{\hat{U}}$ that has terminated and $\mathcal{A}$ eventually outputs a bit b', where $b' = b$ and b is the randomly selected bit in the Test query. The **passive-ke advantage** of $\mathcal{A}$ attacking P is defined to be

$$\mathsf{Adv}_P^{\text{p-ke}}(\mathcal{A}) = 2 \Pr\left(\mathsf{Succ}_P^{\text{p-ke}}(\mathcal{A})\right) - 1 \ . \tag{3.12}$$

Lemma 3.1 applies to $\mathsf{Adv}_P^{\text{p-ke}}(\mathcal{A})$ in an analogous way.

There is no corresponding notion of authentication for passive security, as an adversary trying to break authentication must be active and send at least one message using a Send query.

3.3 Protocols

In this section we describe a few password-authenticated key exchange protocols that will be of later use to us in developing multi-factor password authenticated key exchange protocols.

3.3.1 SRP: Secure Remote Password protocol

The Secure Remote Password protocol (SRP) was introduced by Wu in 1998 [Wu98]; this version of the protocol was called SRP-3. The SRP-3 protocol is a verifier-based protocol meant to offer some resilience in the face of server compromise. It is a very simple protocol and has a very efficient message flow pattern, allowing it to be adapted to fit into existing network protocols.

Unfortunately, however, there is no formal security argument that SRP-3 is a secure password-authenticated key exchange protocol against active adversaries (in the sense of Section 3.2.2) or provides mutual authentication (in the sense of Section 3.2.3); Wu only provides an argument that SRP-3 is a secure key exchange protocol against passive adversaries in the sense of Section 3.2.4 (although Wu's analysis does not use the full language of the model we have stated in that section).

Indeed, in 2001 a problem with the authentication in SRP-3 was discovered by MacKenzie [Mac01], in which an adversary could test two password guesses per online interaction, whereas the most desirable behaviour would be that only one password guess could be eliminated per online interaction. This problem was rectified in SRP-6 [Wu02], and to date no further attacks on SRP-6 have been described in the literature.

Even though SRP-6 has no formal arguments for security or authentication in the face of active adversaries, the protocol has seen some adoption. Libraries exist for using SRP in the Telnet and FTP protocols [Wu08] and an informational Request For Comments exists for using SRP in the popular SSL/TLS protocol [TWMP07].

The SRP-6 protocol consists of two stages: a *user registration stage* and a *login stage*. The user registration stage will typically be invoked once for each user upon account setup, whereas the login stage will be run each time a user attempts to login. All arithmetic is done modulo a large prime n and makes use of a generator (or primitive root modulo n) g, both of which may be standardized or parameters of the scheme.

In the user registration stage, described in Figure 3.1, the client $\hat{C}$ picks a password $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$ and random salt value $s \in_R \mathsf{Salts}$, and uses these values to construct a long-term private value γ and a corresponding verifier Γ. Then, using a private, authentic channel, the client provides $\hat{C}$, s, and Γ to the server $\hat{S}$ which stores them as $\mathsf{pw}_{\hat{S}}[\hat{C}]$.

SRP-6 User Registration		
	Client $\hat{C}$	Server $\hat{S}$
1.	input username $\hat{C}$	
2.	choose password $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$	
3.	$s \in_R \mathsf{Salts}$	
4.	$\gamma = H(s, \hat{C}, \mathsf{pw}_{\hat{C},\hat{S}})$	
5.	$\Gamma = g^{\gamma}$	
6.	$\xrightarrow{\hat{C},s,\Gamma}$	
7.		store $\mathsf{pw}_{\hat{S}}[\hat{C}] = (\hat{C}, s, \Gamma)$

Figure 3.1: SRP-6 protocol user registration stage

In the login stage, described in Figure 3.2, the client $\hat{C}$ provides the server $\hat{S}$ with the username $\hat{C}$ for which the server returns the corresponding salt value s. The client then reconstructs the private value γ. Additionally, the client and server generate ephemeral private values x and y, respectively, and corresponding public values X and Y which they exchange. The client generates S_1 based on the values γ, x, and Y, while the server generates S_2 based on the values Γ, y, and X. When

both parties are honest, the client arrives at the value

$$S_1 = (Y - 3g^\gamma)^{x+u\gamma} = (3\Gamma + g^y - 3g^\gamma)^{x+u\gamma} = (3g^\gamma + g^y - 3g^\gamma)^{x+u\gamma} = g^{xy+uy\gamma} \ , \quad (3.13)$$

while the server arrives at the value

$$S_2 = (X\Gamma^u)^y = (g^x g^{u\gamma})^y = g^{xy+uy\gamma} = S_1 \ . \quad (3.14)$$

The client and server exchange messages M_1 and M_2 which allow them to demonstrate to each other that they know the shared secret value $S_1 = S_2$, which they then convert into a shared session key sk for use in establishing an encrypted channel.

SRP-6 Login			
	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.		$\xrightarrow{\hat{C}}$	
3.			lookup $(\hat{C}, s, \Gamma) = \mathsf{pw}_{\hat{S}}[\hat{C}]$
4.		$\xleftarrow{s}$	
5.	input password $\mathsf{pw}_{\hat{C},\hat{S}}$		
6.	$\gamma = H(s, \hat{C}, \mathsf{pw}_{\hat{C},\hat{S}})$		
7.	$x \in_R \mathbb{Z}_n$		
8.	$X = g^x$		
9.		$\xrightarrow{X}$	
10.			$y \in_R \mathbb{Z}_n$
11.			$Y = 3\Gamma + g^y$
12.		$\xleftarrow{Y}$	
13.			$u = H(X, Y)$
14.			$S_2 = (X\Gamma^u)^y$
15.	$u = H(X, Y)$		
16.	$S_1 = (Y - 3g^\gamma)^{x+u\gamma}$		
17.	$M_1 = H(X, Y, S_1)$		
18.		$\xrightarrow{M_1}$	
19.			verify M_1
20.			$M_2 = H(X, M_1, S_2)$
21.		$\xleftarrow{M_2}$	
22.	verify M_2		
23.	$\mathsf{sk} = H(S_1)$		$\mathsf{sk} = H(S_2)$

Figure 3.2: SRP-6 protocol login stage

Design ideas. A key idea in the design of SRP-6 is that the server in some sense "commits" to its knowledge of the verifier in the first step, and it appears difficult to change its choice of verifier or do a dictionary attack after it receives the client's

M_1 value. In contrast with PAK-Z+ (Section 3.3.3), the server commits to its value before the client does, which allows SRP-6 to fit in the constrained message flow of TLS in which the server sends its key exchange data before the client.

SRP-6 combines design aspects of two protocols. The shielding of the server's ephemeral public key by the verifier is similar to the shielding done in A-EKE [BM94]. Interestingly, the computation of the shared secret is similar to a one-pass version [Ust08b, §5] of the MQV authenticated key exchange protocol with the client's long-term private key being the hash of the password and the server having no long-term private key.

Efficiency. SRP-6 is highly efficient in terms of the number of message flows and in terms of the number of exponentiations performed. Wu [Wu02] provides an optimized message order that requires three (without server-to-client authentication) or four (with server-to-client authentication) message flows.

The client and server each perform three (full-length) exponentiations, only one more than required for the simple, un-authenticated Diffie-Hellman protocol. Since the security argument for security against passive adversaries between SRP-6 and the Computational Diffie-Hellman problem is tight, one could arguably use the same size cryptographic parameters as in Diffie-Hellman to achieve the same level of security, as given in Table 2.1.

Salting. While salting is commonly used in conjunction with passwords to increase the amount of work an attacker must do to obtain raw passwords from a compromised server password database, the salts in SRP-6 do not seem to add any additional security. The goal of salting is that, in the event that an entire server password database is compromised, an attacker should have to do a brute force dictionary attack against each entry in the database. If the passwords were not salted, then an attacker could simply make a table of hashes of all possible passwords (which in many cases could fit on a DVD-ROM, for example), and then compare the database to the pre-computed table. With salts, an attacker must now create a new table for each salt value, which substantially increases the burden on the attacker.

In SRP-6, the password $\mathsf{pw}_{\hat{C},\hat{S}}$ is salted with a value s chosen by the user during the user registration stage, and it is the salted verifier $\Gamma = g^{\gamma}$, where $\gamma = H(s, \hat{C}, \mathsf{pw}_{\hat{C},\hat{S}})$, that is stored on the server. During the login stage, the salt s is returned to the user who then computes the appropriate private value γ based on the salt. We argue that salting the verifier is unnecessary if the goal is to force an attacker to do a brute force dictionary attack against each verifier in the server's database. The value γ is computed as a hash of the salt, the username, and the password. Since each user on the server will have a different username and the hash depends on the username, the username acts as an effective salt across a single server: an attacker has to do a new dictionary attack against each different user's verifier.

Standardization. A variant of SRP-6 has been published as an informational Request for Comments [TWMP07] for use in the SSL/TLS protocol [DR06]. The version in the RFC uses a factor other than 3 (namely, the hash of the prime n and the generator g) in the masking of the verifier in Steps 11 and 16 in Figure 3.2 and explicitly specifies that SHA-1 should be used as the hash function, but is otherwise identical to SRP-6 as presented above. It defines default parameters for 1024-bit primes up to 8192-bit primes and also allows for user-generated parameters.

3.3.1.1 Security of SRP-6

Wu [Wu98] gives an argument that the session key generated by SRP-3 is secure against passive adversaries assuming the hardness of the Computational Diffie-Hellman problem and working in the random oracle model. The same argument applies to SRP-6. We restate the formal security statement for SRP-6 below in the Bellare-Pointcheval-Rogaway model.

Theorem 3.2 *Let G be a finite cyclic group of prime order p generated by g, where g is a primitive root modulo $n = 2p + 1$, where n is prime. Let $\mathcal{A}$ be an adversary that runs in time t. Then*

$$\mathsf{Adv}^{\text{p-ke}}_{\text{PAK}}(\mathcal{A}) = \mathsf{Adv}^{\text{CDH}}_{G,g}(t + t_{\mathsf{exp}}) \ . \tag{3.15}$$

Example instantiation. As a consequence of Theorem 3.2, we can pick a desired security level (as in Table 2.1) and, under the various computational assumptions from Section 2.2, choose a set of parameters that achieve that security level.

Suppose we wish for adversary running in time 2^{80} to have a passive-ke advantage of at most 2^{-20} against SRP.

To give an example instantiation, we have to pick appropriate values for the various parameters in the statement of the theorem. We choose $t = 2^{80}$ and $t_{\mathsf{exp}} = 2^{20}$.

We need $\mathsf{Adv}^{\text{CDH}}_{G,g}(2^{80}) \leq 2^{-20}$. Assuming that the best technique to solve CDH is to solve the Discrete Logarithm problem and that the best method of doing so is as described in Section 2.2.1, we need a group of size $q \geq 2^{2\cdot(20+80)} = 2^{200}$ and a prime of size roughly 2^{1950}.

3.3.2 PAK

The PAK protocol was introduced by Boyko, MacKenzie, and Patel [BMP00a]. It is a symmetric, or non-verifier-based, protocol. The original paper [BMP00a] gave a formal argument that PAK was secure in the simulation model of Shoup [Sho99b] and was later shown to be secure in the BPR model by MacKenzie [Mac02]. The formal statement of security is given in Theorem 3.3 below.

The user registration stage of the PAK protocol is given in Figure 3.3.[5] The client $\hat{C}$ picks a password $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$ and uses this value to construct a long-term private value τ^{-1}. Then, using a private, authenticated channel, the client provides its name $\hat{C}$ and τ^{-1} to the server $\hat{S}$ which stores them as $\mathsf{pw}_{\hat{S}}[\hat{C}]$.

PAK User Registration		
	Client $\hat{C}$	Server $\hat{S}$
1.	input username $\hat{C}$	
2.	choose $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$	
3.	$\tau^{-1} = (H_4(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}}))^{-1}$	
4.	$\xrightarrow{\hat{C},\tau^{-1}}$	
5.		store $\mathsf{pw}_{\hat{S}}[\hat{C}] = \tau^{-1}$

Figure 3.3: PAK protocol user registration stage

The login stage of the PAK protocol is given in Figure 3.4. In this stage, the client picks an ephemeral private key x, shields the corresponding ephemeral public key by the hash τ of the password, and sends this value to the server $\hat{S}$. The server retrieves the stored value for client $\hat{C}$ and uses it to unshield the ephemeral public key sent by the client. The server also generates its own ephemeral private key / public key pair, computes a Diffie-Hellman shared secret σ, and sends its ephemeral public key and a value M_2 proving it knows the value τ^{-1} to the client. The client computes the Diffie-Hellman shared secret value, checks the server's proof of knowledge of τ^{-1}, and then constructs its own proof of knowledge of τ^{-1}. Both parties output a shared session key sk based on the Diffie-Hellman shared secret.

Design ideas. Besides being symmetric as opposed to asymmetric, PAK differs from SRP-6 in that the client sends a message based on its authentication before the server does.

Efficiency. The PAK protocol requires 3 message flows. It requires three exponentiations on the client side and only two exponentiations on the server side, meaning that the computational burden on the server is almost exactly the same as in the simple, unauthenticated Diffie-Hellman protocol. In Section 3.3.2.2, we show how to eliminate one of the exponentiations from the client side for improved efficiency with no effect on security.

Parameter sizes for a particular security level can be calculated from PAK's formal security argument, Theorem 3.3, and are given in the next section.

[5]Note that the numbering of the hash functions in the protocol is irregular but has been chosen to demonstrate the clear connection between PAK and our MFPAK protocol presented in Section 4.4.

PAK Login			
	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.	input password $\mathsf{pw}_{\hat{C},\hat{S}}$		
3.	$\tau = H_4(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}})$		
4.	$x \in_R \mathbb{Z}_q$		
5.	$X = g^x$		
6.	$m = X \cdot \tau$		
7.		$\xrightarrow{\hat{C},m}$	
8.			abort if $\neg\mathsf{Acceptable}(m)$
9.			$y \in_R \mathbb{Z}_q$
10.			$Y = g^y$
11.			lookup $\tau^{-1} = \mathsf{re}_{\hat{S}}[\hat{C}]$
12.			$X = m \cdot \tau^{-1}$
13.			$\sigma = X^y$
14.			$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$
15.			$M_2 = H_5(\mathsf{sid}, \sigma, \tau^{-1})$
16.		$\xleftarrow{Y,M_2}$	
17.	$\sigma = Y^x$		
18.	compute τ^{-1}		
19.	$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$		
20.	abort if $M_2 \neq H_5(\mathsf{sid}, \sigma, \tau^{-1})$		
21.	$M_1 = H_7(\mathsf{sid}, \sigma, \tau^{-1})$		
22.		$\xrightarrow{M_1}$	
23.			abort if $M_1 \neq H_7(\mathsf{sid}, \sigma, \tau^{-1})$
24.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \tau^{-1})$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \tau^{-1})$

Figure 3.4: PAK protocol login stage

3.3.2.1 Security of PAK

MacKenzie [Mac02] gives an argument that PAK is a secure password-authenticated key exchange model in the Bellare-Pointcheval-Rogaway model assuming the hardness of the Computational Diffie-Hellman problem and working in the random oracle model. The formal security statement for PAK is as follows, and includes explicit constants (as opposed to its appearance as Theorem 6.9 in [Mac02]).

Theorem 3.3 (Theorem 6.9, [Mac02]) *Let G be a finite cyclic group generated by g. Assume passwords are uniformly distributed among the set* Passwords. *Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type* Send *and* Execute*, respectively, and at most q_{ro} queries to the random oracles.*

Then, for $t' = t + (4q_{\mathsf{ro}}^2 + q_{\mathsf{se}} + 2q_{\mathsf{ex}})t_{\mathsf{exp}}$,

$$\mathsf{Adv}^{\mathrm{ake}}_{\mathrm{PAK}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \ , \tag{3.16}$$

where

$$\epsilon = 2q_{\mathsf{se}}\mathsf{Adv}^{\mathrm{CDH}}_{G,g}\left(t', q_{\mathsf{ro}}^2\right) + 2\frac{(q_{\mathsf{se}} + q_{\mathsf{ex}})(q_{\mathsf{ro}} + q_{\mathsf{se}} + q_{\mathsf{ex}})}{|G|} \ . \tag{3.17}$$

Moreover, the same bound applies for $\mathsf{Adv}^{\mathrm{ma}}_{\mathrm{PAK}}(\mathcal{A})$.

Example instantiation. As a consequence of Theorem 3.3, we can pick a desired security level (as in Table 2.1) and, under the various computational assumptions from Section 2.2, choose a set of parameters that achieve that security level.

Suppose we wish for an adversary running in time 2^{80} to have an ake advantage of at most 2^{-20} against PAK.

To give an example instantiation, we have to pick appropriate values for the various parameters in the statement of the theorem. We choose

$$\begin{aligned} |\mathsf{Clients}| &= 2^{15} & |\mathsf{Servers}| &= 2^5 & \\ q_{\mathsf{se}} &= 2^{10} & q_{\mathsf{ex}} &= 2^{20} & q_{\mathsf{ro}} &= 2^{40} \\ t &= 2^{80} & t_{\mathsf{exp}} &= 2^{20} \ . & \end{aligned}$$

We need $\frac{q_{\mathsf{se}}}{|\mathsf{Passwords}|} \leq 2^{-21}$. To achieve this, we need $|\mathsf{Passwords}| \geq 2^{31}$, which, on the author's keyboard with 95 distinct printable characters on it, is achieved by having passwords of length 5, assuming passwords are uniformly distributed.

We also need $\epsilon \leq 2^{-21}$. Of the two terms in expression (3.17), the latter is dominated by the former. Noting that $t' = 2^{102}$, we require that $2^{11}\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(2^{102}, 2^{80}) \leq 2^{-21}$. Assuming that the best technique to solve CDH is to solve the Discrete Logarithm problem and that the best method of doing so is as described in Section 2.2.1, we need a group of size $q \geq 2^{2(11+21+102+80)} = 2^{428}$.

3.3.2.2 mePAK: A more efficient PAK

In this section we describe a revised form of the PAK protocol that is more efficient on the client side, in terms of number of operations, and demonstrate that this more efficient protocol has the same security as PAK. The login stage of the PAK protocol requires two exponentiations and one inversion on the client side, and two exponentiations on the server side. By precomputing the inversion and storing this additional value on the server during the user registration stage, we can eliminate the inversion from the login stage and achieve the same number of operations as Diffie-Hellman — two exponentiations — on both client and server.

The user registration stage of our more efficient PAK protocol is given in Figure 3.5. We note that while the protocol states that the server computes τ^{-1}, the

protocol could be altered so that the client computes τ^{-1} and transmits both τ and τ^{-1} with the server; the server would need to verify that the inverse was computed correctly.

mePAK: **More Efficient PAK User Registration**		
	Client $\hat{C}$	Server $\hat{S}$
1.	input username $\hat{C}$	
2.	choose $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$	
3.	$\tau = H_4(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}})$	
4.	$\xrightarrow{\hat{C},\tau}$	
5.		compute τ^{-1}
6.		store $\mathsf{pw}_{\hat{S}}[\hat{C}] = (\tau, \tau^{-1})$

Figure 3.5: mePAK: More Efficient PAK protocol user registration stage

The login stage of our more efficient PAK protocol is given in Figure 3.6. The main difference from the PAK protocol (Figure 3.4) is that hash function calls on lines 15, 19, 20, 22, and 23 use τ instead of τ^{-1}, saving the client the need to perform an inversion.

The following theorem relates the security of our mePAK protocol to the original PAK protocol, showing that the two are essentially equivalent:

Theorem 3.4 *Let G be a finite cyclic group generated by g. Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type* Send *and* Execute*, respectively, and at most q_{ro} queries to the random oracles. Let* mePAK *denote the More Efficient PAK protocol in Figures 3.5 and 3.6. For $t' = t + q_{\mathsf{ro}} t_{\mathsf{exp}}$,*

$$\mathsf{Adv}^{\mathrm{ake}}_{\mathrm{mePAK}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ake}}_{\mathrm{PAK}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) \ . \tag{3.18}$$

Moreover, the corresponding bound applies for $\mathsf{Adv}^{\mathrm{ma}}_{\mathrm{mePAK}}(\mathcal{A})$.

Proof. The main idea of the proof is as follows. We construct a reduction which shows that an adversary who can break mePAK can be used to break the original PAK protocol. Since the only difference between the two protocols is the use of the non-inverted value τ instead of τ^{-1} in hash function calls, we construct a PAK system which is instantiated with hash functions that invert the appropriate argument. Some random oracle queries thus require exponentiations (inversions), so we need to allow for an increase of $q_{\mathsf{ro}} t_{\mathsf{exp}}$ in the runtime of the simulator. Assuming $q_{\mathsf{ro}} t_{\mathsf{exp}} < t$, the runtime of the simulator will be effectively unchanged.

Let $\mathcal{S}_{\mathrm{PAK}}$ denote the PAK system that we will attack. We will construct a modifier $\mathcal{M}$ that will transform an adversary $\mathcal{A}$'s attack against mePAK into an attack against the PAK system $\mathcal{S}_{\mathrm{PAK}}$.

mePAK: **More Efficient PAK Login**			
	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.	input password $\mathsf{pw}_{\hat{C},\hat{S}}$		
3.	$\tau = H_4(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}})$		
4.	$x \in_R \mathbb{Z}_q$		
5.	$X = g^x$		
6.	$m = X \cdot \tau$		
7.		$\xrightarrow{\hat{C},m}$	
8.			abort if $\neg\mathsf{Acceptable}(m)$
9.			$y \in_R \mathbb{Z}_q$
10.			$Y = g^y$
11.			lookup $(\tau, \tau^{-1}) = \mathsf{re}_{\hat{S}}[\hat{C}]$
12.			$X = m \cdot \tau^{-1}$
13.			$\sigma = X^y$
14.			$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$
15.			$M_2 = H_5(\mathsf{sid}, \sigma, \tau)$
16.		$\xleftarrow{Y,M_2}$	
17.	$\sigma = Y^x$		
18.	$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$		
19.	abort if $M_2 \neq H_5(\mathsf{sid}, \sigma, \tau)$		
20.	$M_1 = H_7(\mathsf{sid}, \sigma, \tau)$		
21.		$\xrightarrow{M_1}$	
22.			abort if $M_1 \neq H_7(\mathsf{sid}, \sigma, \tau)$
23.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \tau)$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \tau)$

Figure 3.6: mePAK: More Efficient PAK protocol login stage

Instantiation of PAK **system.** We instantiate the PAK system $\mathcal{S}_{\text{PAK}}$ with the following random oracles:

$$H^*_\ell((\hat{C}, \hat{S}, m, Y), \sigma, \tau') = H_\ell((\hat{C}, \hat{S}, m, Y), \sigma, (\tau')^{-1}) \ , \tag{3.19}$$

for $\ell = 5, 7, 8$. These 'starred' functions are independent random oracles if the corresponding unstarred functions are, since inverting one input to the random oracle only effects a permutation on the domain. Thus, $\mathcal{S}_{\text{PAK}}$ is a true PAK instantiation.

Because each random oracle query now involves an inversion, the modifier requires additional time to answer random oracle queries. In particular, $\mathcal{M}$ requires at most $q_{\mathsf{ro}} t_{\mathsf{exp}}$ additional time.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ passes every query it receives from $\mathcal{A}$ to $\mathcal{S}_{\text{PAK}}$ and returns every result it receives without any changes. In par-

ticular, Test queries are passed directly to $\mathcal{S}_{\text{PAK}}$ unaltered. This means that $\mathcal{A}$'s guess of b in $\mathcal{M}$ corresponds to a guess of b in $\mathcal{S}_{\text{PAK}}$.

$\mathcal{A}$'s view of $\mathcal{M}$. From $\mathcal{A}$'s perspective, $\mathcal{M}$ behaves exactly as a mePAK system should. Because of the substitution of the random oracles H^*_ℓ, all queries are handled exactly as mePAK would. Moreover, the passwords have the same distribution in PAK and mePAK.

Because $\mathcal{S}_{\text{PAK}}$ is exactly a PAK instance and because an attack against mePAK can be directly translated into an attack against PAK with only a small increase in computational time due to random oracle queries, the result follows and mePAK is effectively as secure as PAK. □

3.3.3 PAK-Z+

The PAK-Z+ protocol was introduced by Gentry, MacKenzie, and Ramzan in 2005 [GMR05]. It is an asymmetric, or verifier-based, protocol. The protocol has a formal argument for security in the BPR model, and the formal statement of security is given in Theorem 3.5 below.

The user registration stage of the PAK-Z+ protocol is given in Figure 3.7.[6] The client $\hat{C}$ picks a password $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$. The client generates a private key / public key pair for a digital signature algorithm, then shields the private key using its password. Using a private, authenticated channel, the client provides its name $\hat{C}$, its shielded private key, and some verification values to the server $\hat{S}$ which stores them as $\mathsf{pw}_{\hat{S}}[\hat{C}]$.

PAK-Z+ User Registration		
	Client $\hat{C}$	Server $\hat{S}$
1.	input username $\hat{C}$	
2.	choose $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$	
3.	$(v, V) \leftarrow \mathsf{Gen}(1^\kappa)$	
4.	$\gamma^{-1} = (H_1(\hat{C}, \hat{S}, \mathsf{pw}))^{-1}$	
5.	$v' = H_2(\hat{C}, \hat{S}, \mathsf{pw}) \oplus v$	
6.	$v'' = H_3(v)$	
7.	$\xrightarrow{\hat{C},\gamma^{-1},V,v',v''}$	
8.		store $\mathsf{pw}_{\hat{S}}[\hat{C}] = (\gamma^{-1}, V, v', v'')$

Figure 3.7: PAK-Z+ protocol user registration stage

[6]Note that the numbering of the hash functions in the protocol is irregular but has been chosen to demonstrate the clear connection between PAK-Z+ and our MFPAK protocol presented in Section 4.4.

The login stage of the PAK-Z+ protocol is given in Figure 3.8. In this stage, the client picks an ephemeral private key x, shields the corresponding ephemeral public key by the hash γ of the password, and sends this value to the server $\hat{S}$. The server retrieves the stored value for client $\hat{C}$ and uses it to unshield the ephemeral public key sent by the client. The server also generates its own ephemeral private key / public key pair and computes a Diffie-Hellman shared secret σ. It uses σ to shield the user's signature private key. The server sends the shielded signature private key, its ephemeral public key, and a value M_2 proving it knows γ^{-1} to the client. The client computes the Diffie-Hellman shared secret value, checks the server's proof of knowledge of γ^{-1}, and then constructs its own proof of knowledge of the password based on the digital signature scheme. Both parties output a shared session key sk based on the Diffie-Hellman shared secret.

Design ideas. Like SRP, PAK-Z+ differs from PAK in that it is an asymmetric protocol as opposed to symmetric. The construct that allows asymmetric is more complicated than in SRP, however. A signature scheme is used, the client's signing private key is shielded by the password, and the server cannot determine either of these without doing an offline dictionary attack. The scheme requires, however, that the client send its ephemeral key exchange value first, making it incapable of fitting in the SSL/TLS handshake protocol.

Efficiency. The PAK-Z+ protocol requires 3 message flows. It requires three exponentiations and one signature generation on the client side and only two exponentiations and one signature verification on the server side. If ECDSA is used as the signature algorithm, for example, each side needs approximately one additional exponentiation operation (if multiple point multiplication is roughly the same cost as single point multiplication [HMV04, §3.3.3]), meaning that the computational burden on the server is 50% higher than in the simple, unauthenticated Diffie-Hellman protocol. Parameter sizes for a particular security level can be calculated from Theorem 3.5 and are given in the next section.

3.3.3.1 Security of PAK-Z+

Gentry, MacKenzie, and Ramzan [GMR05] give an argument that PAK-Z+ is a secure password-authenticated key exchange model in the Bellare-Pointcheval-Rogaway model assuming the hardness of the Computational Diffie-Hellman problem and working in the random oracle model. The formal security statement for PAK-Z+ is as follows, and includes explicit constants (as opposed to its appearance as Theorem 5.1 in [GMR05]).

Theorem 3.5 (Theorem 5.1, [GMR05]) *Let G be a finite cyclic group generated by g. Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type* Send *and* Execute*, respectively, and at most q_{ro} queries*

PAK-Z+ Login			
	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.	input password $\mathsf{pw}_{\hat{C},\hat{S}}$		
3.	$\gamma = H_1(\hat{C}, \hat{S}, \mathsf{pw})$		
4.	$x \in_R \mathbb{Z}_q$		
5.	$X = g^x$		
6.	$m = X \cdot \gamma$		
7.		$\xrightarrow{\hat{C},m}$	
8.			abort if $\neg\mathsf{Acceptable}(m)$
9.			$y \in_R \mathbb{Z}_q$
10.			$Y = g^y$
11.			lookup $(\gamma^{-1}, V, v', v'') = \mathsf{pw}_{\hat{S}}[\hat{C}]$
12.			$X = m \cdot \gamma^{-1}$
13.			$\sigma = X^y$
14.			$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$
15.			$M_2 = H_5(\mathsf{sid}, \sigma, \gamma^{-1})$
16.			$a' = H_6(\mathsf{sid}, \sigma, \gamma^{-1})$
17.			$a = a' \oplus v'$
18.		$\xleftarrow{Y,M_2,a,v''}$	
19.	$\sigma = Y^x$		
20.	compute γ^{-1}		
21.	$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$		
22.	abort if $M_2 \neq H_5(\mathsf{sid}, \sigma, \gamma^{-1})$		
23.	$a' = H_6(\mathsf{sid}, \sigma, \gamma^{-1})$		
24.	$v' = a' \oplus a$		
25.	$v = H_2(\hat{C}, \hat{S}, \mathsf{pw}) \oplus v'$		
26.	abort if $v'' \neq H_3(v)$		
27.	$s = \mathsf{Sign}_v(\mathsf{sid})$		
28.		$\xrightarrow{s}$	
29.			abort if $\neg\mathsf{Verify}_V(\mathsf{sid}, s)$
30.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma^{-1})$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma^{-1})$

Figure 3.8: PAK-Z+ protocol login stage

to the random oracles. Assume passwords are uniformly distributed among the set Passwords*. Let* $b_{\mathsf{re}} = 1$ *if* $\mathcal{A}$ *makes a* RevealPWS *query and 0 otherwise. Then, for* $t' = t + (8q_{\mathsf{ro}}^2 + q_{\mathsf{se}} + q_{\mathsf{ex}})t_{\mathsf{exp}}$*,*

$$\mathsf{Adv}^{\mathrm{ake}}_{\text{PAK-Z+}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}(1 - b_{\mathsf{re}}) + q_{\mathsf{ro}} b_{\mathsf{re}}}{|\mathsf{Passwords}|} + \epsilon \ , \tag{3.20}$$

where

$$\epsilon = 2q_{\mathsf{se}}\mathsf{Adv}^{\mathrm{CDH}}_{G,g}\left(t', q^2_{\mathsf{ro}}\right) + 2q_{\mathsf{se}}\mathsf{Succ}^{\text{eu-cma}}_{\kappa}(t', q_{\mathsf{se}}) + \frac{(q_{\mathsf{se}} + q_{\mathsf{ex}})(q_{\mathsf{ro}} + q_{\mathsf{se}} + q_{\mathsf{ex}})}{|G|} \quad . \tag{3.21}$$

Moreover,

$$\mathsf{Adv}^{\mathrm{s2c}}_{\text{PAK-Z+}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \tag{3.22}$$

and

$$\mathsf{Adv}^{\mathrm{c2s}}_{\text{PAK-Z+}}(\mathcal{A}) \leq \frac{q_{\mathsf{se}}(1 - b_{\mathsf{re}}) + q_{\mathsf{ro}}b_{\mathsf{re}}}{|\mathsf{Passwords}|} + \epsilon \quad . \tag{3.23}$$

One interesting aspect of the above theorem is that it explicitly models the asymmetric nature of the security of the password: if the verifier has been compromised ($b_{\mathsf{re}} = 1$), the best attack is an offline dictionary attack, but if the verifier has not been compromised ($b_{\mathsf{re}} = 0$), the best attack is an online attack.

Example instantiation. As a consequence of Theorem 3.5, we can pick a desired security level (as in Table 2.1) and, under the various computational assumptions from Section 2.2, choose a set of parameters that achieve that security level.

Suppose we wish for an adversary running in time 2^{80} to have an ake advantage of at most 2^{-20} against PAK-Z+.

To give an example instantiation, we have to pick appropriate values for the various parameters in the statement of the theorem. We choose the same values as in Section 3.3.2.1.

Before applying the theorem directly, however, we need to interpret the role of b_{re}. Recall that $b_{\mathsf{re}} = 1$ if and only if the adversary makes a RevealPWS query. In the statement of security, the term $(1 - b_{\mathsf{re}})q_{\mathsf{se}}/|\mathsf{Passwords}|$ indicates how many online attacks (when the verifier is not compromised) are required, whereas the term $b_{\mathsf{re}}q_{\mathsf{ro}}/|\mathsf{Passwords}|$ indicates how many random oracle queries are required in an offline dictionary attack when the verifier is compromised. Noting that q_{ro} also plays a role in the online attack analysis, allowing q_{ro} to be a reasonable size in the online attack analysis (say, 2^{40}) would force the password size to be larger than is desirable. In what follows, we choose $|\mathsf{Passwords}|$ based on q_{se}, and then, given that set of passwords, comment on the work required if the verifier is compromised.

Hence, we need $q_{\mathsf{se}}/|\mathsf{Passwords}| \leq 2^{-21}$, or $|\mathsf{Passwords}| \geq 2^{31}$, which, on the author's keyboard with 95 distinct printable characters on it, is achieved by having passwords of length 5, assuming passwords are uniformly distributed. An offline dictionary attack against the verifier would require roughly 2^{31} random oracle queries to succeed with high probability.

We also need $\epsilon \leq 2^{-21}$. Expression (3.21) is dominated by the first term. Noting that $t' = 2^{103}$, we require that $2^{11}\mathsf{Adv}^{\mathrm{CDH}}_{G,g}(2^{103}, 2^{80}) \leq 2^{-21}$. Assuming that the best technique to solve CDH is to solve the Discrete Logarithm problem and that the

best method of doing so is as described in Section 2.2.1, this means we need a group of size $q \geq 2^{2(11+21+103+80)} = 2^{430}$.

3.3.3.2 mePAK-Z+: A more efficient PAK-Z+

In this section we describe a revised form of the PAK-Z+ protocol that is more efficient on the client side, in terms of number of operations, and demonstrate that this more efficient protocol has the same security as PAK-Z+. The login stage of the PAK-Z+ protocol requires two exponentiations, one inversion, and one signing operation on the client side, and three exponentiations on the server side. By precomputing the inversion and storing this additional value on the server during the user registration stage, we can eliminate the inversion from the login stage and achieve a protocol with just one more operation than Diffie-Hellman — three exponentiations — on both client and server.

The user registration stage of our more efficient PAK-Z+ protocol is given in Figure 3.9. We note that while the protocol states that the server computes γ^{-1}, the protocol could be altered so that the client computes γ^{-1} and transmits both γ and γ^{-1} with the server; the server would need to verify that the inverse was computed correctly.

mePAK-Z+: **More Efficient PAK-Z+ User Registration**		
	Client $\hat{C}$	Server $\hat{S}$
1.	input username $\hat{C}$	
2.	choose $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$	
3.	$(v, V) \leftarrow \mathsf{Gen}(1^\kappa)$	
4.	$\gamma = H_1(\hat{C}, \hat{S}, \mathsf{pw})$	
5.	$v' = H_2(\hat{C}, \hat{S}, \mathsf{pw}) \oplus v$	
6.	$v'' = H_3(v)$	
7.		$\xrightarrow{\hat{C},\gamma,V,v',v''}$
8.		compute γ^{-1}
9.		store $\mathsf{pw}_{\hat{S}}[\hat{C}] = (\gamma, \gamma^{-1}, V, v', v'')$

Figure 3.9: PAK-Z+ protocol user registration stage

The login stage of our more efficient PAK-Z+ protocol is given in Figure 3.10. The main difference from the PAK-Z+ protocol (Figure 3.8) is that hash function calls on lines 15, 16, 21, 24, and 29 use γ instead of γ^{-1}, saving the client the need to perform an inversion.

The following theorem relates the security of our mePAK-Z+ protocol to the original PAK-Z+ protocol, showing that the two are essentially equivalent:

Theorem 3.6 *Let G be a finite cyclic group generated by g. Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type* Send *and* Execute*,*

mePAK-Z+: **More Efficient PAK-Z+ Login**			
	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.	input password $\mathsf{pw}_{\hat{C},\hat{S}}$		
3.	$\gamma = H_1(\hat{C}, \hat{S}, \mathsf{pw})$		
4.	$x \in_R \mathbb{Z}_q$		
5.	$X = g^x$		
6.	$m = X \cdot \gamma$		
7.		$\xrightarrow{\hat{C},m}$	
8.			abort if $\neg\mathsf{Acceptable}(m)$
9.			$y \in_R \mathbb{Z}_q$
10.			$Y = g^y$
11.			lookup $(\gamma, \gamma^{-1}, V, v', v'') = \mathsf{pw}_{\hat{S}}[\hat{C}]$
12.			$X = m \cdot \gamma^{-1}$
13.			$\sigma = X^y$
14.			$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$
15.			$M_2 = H_5(\mathsf{sid}, \sigma, \gamma)$
16.			$a' = H_6(\mathsf{sid}, \sigma, \gamma)$
17.			$a = a' \oplus v'$
18.		$\xleftarrow{Y,M_2,a,v''}$	
19.	$\sigma = Y^x$		
20.	$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$		
21.	abort if $M_2 \neq H_5(\mathsf{sid}, \sigma, \gamma)$		
22.	$a' = H_6(\mathsf{sid}, \sigma, \gamma)$		
23.	$v' = a' \oplus a$		
24.	$v = H_2(\hat{C}, \hat{S}, \mathsf{pw}) \oplus v'$		
25.	abort if $v'' \neq H_3(v)$		
26.	$s = \mathsf{Sign}_v(\mathsf{sid})$		
27.		$\xrightarrow{s}$	
28.			abort if $\neg\mathsf{Verify}_V(\mathsf{sid}, s)$
29.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma)$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma)$

Figure 3.10: mePAK-Z+: More Efficient PAK-Z+ protocol login stage

respectively, and at most q_{ro} *queries to the random oracles. Let* mePAK-Z+ *denote the More Efficient PAK-Z+ protocol in Figures 3.9 and 3.10. For* $t' = t + q_{\mathsf{ro}} t_{\mathsf{exp}}$,

$$\mathsf{Adv}^{\mathrm{ake}}_{\text{mePAK-Z+}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{ake}}_{\text{PAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) \ . \tag{3.24}$$

Moreover, the corresponding bounds apply for $\mathsf{Adv}^{\mathrm{c2s}}_{\text{mePAK-Z+}}(\mathcal{A})$ *and* $\mathsf{Adv}^{\mathrm{s2c}}_{\text{mePAK-Z+}}(\mathcal{A})$.

The proof of the theorem proceeds in the same way as the proof of Theorem 3.4.

Chapter 4

Multi-Factor Password-Authenticated Key Exchange

Contents

4.1 Introduction

Multi-factor password-authenticated key exchange aims to enhance password-authenticated key exchange by using multiple authentication factors. A client and a server authenticate each other by proving their knowledge of multiple short secrets without revealing any useful information about the secrets to any other party. Such a protocol should remain secure even if all but one of the authentication secrets has been compromised by an adversary. Moreover, the two parties also agree on a shared session key.

In this chapter, we describe the problem of multi-factor password-authenticated key exchange, propose a formal model for analysing the security of such protocols, and present an efficient two-factor protocol that is secure in this model under standard cryptographic assumptions in the random oracle model.

Practical motivation. Two major security problems on the Internet today are phishing and spyware. **Phishing**, or server impersonation, occurs when a malicious server convinces a user to reveal sensitive personal information, such as a username and password, to a malicious server instead of the real server; the phisher can then use the information obtained to impersonate the user. Additionally, many users' computers are compromised with **spyware**, which can record users' keystrokes (and thus passwords) and transmit this information to a malicious party.

Such attacks are possible not because of the break of any cryptographic protocol but because of externalities such as social engineering and software bugs.

Several techniques to reduce the risks of these attacks are being used in practice. Physical devices that generate one-time passwords are being used to secure corporate virtual private networks (VPNs) and some online banking sessions. Server-side *multi-layer* techniques that take into account additional attributes, such as HTTP cookies, IP address, and browser user agent string, are being deployed as well. These techniques can offer greater assurance as to the identity of the user but, even when deployed over today's web security protocol HTTPS/TLS, remain susceptible to sophisticated impersonation attacks because they do not protect authentication secrets. Moreover, they do not provide strong, intuitive server-to-client authentication.

Password-authenticated key exchange is a strong technique to defend against impersonation attacks and provide server-to-client authentication, but current protocols depend solely on a long-term password, which can be risky when used on a spyware-infested computer.

Multi-factor authentication adds a further degree of assurance to the authentication procedure. Long-term passwords are easily memorized, infrequently changed, and used repeatedly. One-time responses are used once: they change frequently and, though not easily memorized, can be provided by a small electronic token or a sheet of paper. These factors offer different but complementary resistance to different types of compromise. Together, they offer more assurance in authentication because stealing the long-term password alone (for example, by installing spyware) or losing the one-time password card alone is insufficient to compromise the authentication procedure.

We believe that it is important to design a multi-factor protocol that can leverage multiple client authentication attributes and, equally important, to convey them securely in a multi-factor cryptographic protocol. Our approach builds upon previous work on password-authenticated key exchange by combining multiple authentication factors of complementary natures in a multi-factor authenticated key exchange protocol.

Contributions. We provide the first formal security treatment of multi-factor password-authenticated key exchange. We define a formal model which is an extension of the Bellare-Pointcheval-Rogaway model [BPR00] for password-authenticated key exchange. We formalize the security notion that a multi-factor protocol should remain secure even one of the two factors has been compromised by adapting the definition of freshness of a session.

We present an efficient two-factor protocol, MFPAK, that is secure in this model under standard cryptographic assumptions in the random oracle model.

Our multi-factor authentication protocol offers enhanced authentication protection through two complementary factors, a long-term password and a one-time response, and achieves two-factor security with the same computational efficiency as the one-factor protocol mePAK-Z+ from Section 3.3.3.2. The protocol remains secure even if all but one of the authentication factors is fully known to an adversary. Protocols secure in our model are resistant to man-in-the-middle and impersonation attacks, providing enhanced authentication in the face of more complex threats like spyware and phishing.

Our work differs from previous work in password-authenticated key exchange because it utilizes two independent, complementary factors for authentication. Other work has considered some aspects of multi-factor authentication, but these have either used at least one factor that is a long cryptographic secret (as opposed to our work which allows both factors to be short, human-friendly strings), or have not provided strong server-to-client authentication resistant to man-in-the-middle attacks.

Future directions. Our model, in its currently stated form, only addresses two factors, but it can be easily extended to accommodate additional factors if an application demands greater authentication security. We believe that an efficient protocol can be developed for more than two factors by combining additional factors in the same way as we combined aspects of the mePAK and mePAK-Z+ protocols. To bring this work closer to real-world scenarios, another future direction is to reduce the assumption that passwords are uniformly distributed.

Moreover, multi-factor protocols could be designed where some factors are optional and the number of factors used corresponds to differing levels of access depending on the application situation: one factor could be used for read-only access, two factors for small-value transactions, and three factors for large-value transactions.

An interesting future direction would be to integrate "fuzzy" attributes, such as biometric information, into a multi-factor authenticated key exchange protocol. Multi-factor authentication is often colloquially described as being based on 'something you know' (a password), 'something you have' (a one-time response value), and 'something you are' (a biometric attribute, such as a fingerprint). Biometric attributes can be challenging to use in a cryptographic protocol, however, because

they are not perfectly reproduced every time, and thus some techniques must be used to accommodate their fuzziness; one such technique is described in [PZ07], but implementation remains a challenge.

Outline. The rest of this chapter proceeds as follows. Section 4.2 reviews other approaches to multi-factor authentication in the literature. In Section 4.3, we describe the security model for multi-factor password-authenticated key exchange. In Section 4.4, we present our protocol MFPAK and discuss its efficiency. We show through a formal analysis in Section 4.4.4 that the MFPAK protocol is secure in our model and give a sample instantiation for a common security level.

4.2 Literature review

In Section 3.1, we review password-authenticated key exchange protocols. Two protocols that will be of interest to us in this chapter are mePAK (Section 3.3.2.2) and mePAK-Z+ (Section 3.3.3.2).

Although multi-factor authentication has been known for some time, it has been sparingly studied in the literature.

A two-factor authentication scheme for smart cards was proposed by Yang *et al.* [YWWD06a]. Their scheme relies on a smart card storing and returning a cryptographically large (e.g., 160-bit) private value, relies on a public key infrastructure, and requires that the user input a password into the smart card for each login. Other protocols that require the client to store a long cryptographic secret and the server's public key include schemes by Park and Park [PP04] and Yoon and Yoo [YY06]. Pointcheval and Zimmer [PZ07] also provide a multi-factor authentication scheme using a password, a long cryptographic secret, and biometric data; their scheme has a formal security argument in a variant of the Bellare-Pointcheval-Rogaway model.

There are other two-factor authentication schemes used in practice which do not provide cryptographic protection for the two factors. In a **multi-channel** system, the second factor is delivered over a separate channel (e.g., via an SMS text message on a mobile phone), which the user then inputs into their web browser along side their password. In a **multi-layer** system, software installed on the server evaluates additional attributes such as an HTTP cookie, IP address, and browser user agent string to heuristically analyze whether the user is likely to be authentic. Some multi-layer systems try to offer additional reassurance to the user of the server's identity by presenting the user with a customized image or string. While these multi-channel and multi-layer approaches can offer some increased assurance, they can be defeated by non-cryptographic means such as sophisticated man-in-the-middle attacks and spyware, and have been shown to be easily ignored by users [SDOF07].

4.3 Security for multi-factor protocols

In a multi-factor password-authenticated key exchange protocol, multiple authentication secrets of complementary natures, such as a long-term password and a one-time response value, are combined securely to provide mutual authentication and to establish a shared secret key for a private channel.

The authentication secrets must be combined so that the client can convince the server that it knows all the authentication secrets and that the server can convince the client that it knows all the authentication secrets: this provides mutual authentication. However, the protocol must be carefully designed to not reveal any information about the authentication secrets to a passive or even an active adversary.

In addition to providing authentication, the protocol should also establish an ephemeral shared secret key between client and server that can then be used, for example, to establish a private channel using bulk encryption.

4.3.1 Informal security criteria

The general security criteria we use for multi-factor password-authenticated key exchange is that the protocol should remain secure even if all but one authentication factor is fully known to an adversary. In this chapter, we present the first example of such a protocol using two authentication factors. We identify four security properties that such a protocol should have:

1. Strong two-factor server-to-client authentication: without knowledge of both of the authentication factors, a server cannot successfully convince a client of its identity.

2. Strong two-factor client-to-server authentication: without knowledge of both of the authentication factors, a client cannot successfully convince a server of its identity.

3. Authentication secrets protected: no useful information about the authentication secrets is revealed during the authentication process.

4. Secure session key establishment: at the end of the protocol, a client and a server end up with a secure shared session key suitable for bulk encryption only if the mutual authentication is successful; otherwise no session is established.

Figure 4.1 compares our scheme, MFPAK, with existing password-authenticated key exchange protocols, with a two-factor scheme that transmits the password and response value across an SSL channel, and with a multi-layer scheme that uses non-cryptographic attributes, such as browser version and IP address, for additional assurance.

<table>
<tr><th>Security property</th><th>SSL +
multi-channel schemes</th><th>SSL + password +
one-time response or
SSL + multi-layer schemes</th><th>Existing password
auth. key exchange
protocols</th><th>MFPAK</th></tr>
<tr><td>1. Strong two-factor server auth.</td><td colspan="2">Susceptible to man-in-the-middle attacks.
Server authenticated only by X.509 certificate.</td><td>Only one factor.</td><td>Yes.</td></tr>
<tr><td rowspan="2">2. Strong two-factor client auth.</td><td colspan="2">Susceptible to man-in-the-middle attacks.</td><td rowspan="2">Only one factor.</td><td rowspan="2">Yes.</td></tr>
<tr><td>Needs second channel.</td><td></td></tr>
<tr><td>3. Auth. secrets protected</td><td colspan="2">Authentication occurs after session key established.
User authentication secrets sent directly to server.</td><td>Yes.</td><td>Yes.</td></tr>
<tr><td>4. Secure session key establishment</td><td colspan="2">Yes.</td><td>Yes.</td><td>Yes.</td></tr>
</table>

Figure 4.1: Comparison of security properties of various schemes

This table shows that other two-factor schemes that the financial industry is deploying today, such as transmitting one-time values over TLS, will not address the phishing problem. Solving this problem requires a fundamental change in the underlying cryptographic protocol. Our scheme provides a secure solution to this problem, in the form of multi-factor password-authenticated key exchange.

Nature of the factors. The common scenario for password-authenticated key exchange is to have a fixed long-term password for each client-server pair; we call this the **first factor**. For multi-factor password-based authentication, we add a **second factor**: a changing, short-term password for each client-server pair that varies with time or other environment factors. Both factors are short strings, which contrasts with the work of Pointcheval and Zimmer [PZ07] who use a short string, a long private key, and a biometric. Although using a long cryptographic secret can provide greater security, this technique requires client certificates which can be difficult to manage for non-technical users and are not very portable. Our focus is on users being able to carry their authentication secrets easily from computer to computer. By using two factors, we can provide robustness against different types of compromise. For example, to be able to use the second factor, an adversary has to steal the one-time password generator from the client or compromise the server's database of one-time passwords.

4.3.2 Formal model

We define a formal model for the security of multi-factor password-authenticated key exchange that allows us to show a protocol secure by giving upper bounds on the probability that an adversary can break server-to-client or client-to-server authentication, or determine the session key established; the authentication secrets are protected as well.

This formal model is an extension of the model for password-authenticated key exchange proposed by Bellare, Pointcheval, and Rogaway [BPR00] and modified by Gentry, MacKenzie, and Ramzan [GMR05], presented in Section 3.2. We state our model for two factors, but it could easily be extended for an arbitrary number of factors. We only note below how our model differs from the previous presentation.

Passwords. Authentication secrets are short strings selected uniformly at random from an appropriate set. There are two sets of authentication secrets: long-term passwords are chosen from the set $\mathsf{Passwords}$, and short-term passwords are chosen from the set $\mathsf{Responses}$.[1]

For each client-server pair $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$, two authentication secrets exist: there is a long-term password $\mathsf{pw}_{\hat{C},\hat{S}} \in \mathsf{Passwords}$ and a short-term password

[1] We call the set of short-term passwords $\mathsf{Responses}$ because it may often be the case that these values are the response to challenges issued by the server.

$\mathsf{re}_{\hat{C},\hat{S}} \in \mathsf{Responses}$, and corresponding $\mathsf{pw}_{\hat{S}}[\hat{C}]$ and $\mathsf{re}_{\hat{S}}[\hat{C}]$. We treat the long-term password as an asymmetric factor and the short-term password as a symmetric factor, so $\mathsf{re}_{\hat{S}}[\hat{C}]$ is a trivial transformation of $\mathsf{re}_{\hat{C},\hat{S}}$.

Powers of the adversary. In addition to the standard queries allowed in Section 3.2 (Send, Execute, RevealSessionKey, Test), we allow individual queries for each authentication secret.

$\mathsf{RevealPWC}_P(\hat{C}, \hat{S})$: Returns the long-term password $\mathsf{pw}_{\hat{C},\hat{S}}$ of client $\hat{C}$ with server $\hat{S}$. This query is used to model compromise of the client's long-term password, for example by the use of spyware installed on the client's computer.

$\mathsf{RevealPWS}_P(\hat{S}, \hat{C})$: Returns the transformed long-term password $\mathsf{pw}_{\hat{S}}[\hat{C}]$ of client $\hat{C}$ on server $\hat{S}$. This query is used to model the compromise of the server's password database, which, in the verifier-based model, may not have the same effect as the compromise of the client's password.

$\mathsf{RevealRe}_P(\hat{C}, \hat{S})$: Returns the short-term password $\mathsf{re}_{\hat{C},\hat{S}}$ of client $\hat{C}$ with server $\hat{S}$. This query is used to model compromise of the client's short-term password, for example by phishing or the use of spyware installed on the client's computer.

Freshness. We adapt the notion of freshness to accommodate the idea that a session should remain fresh even if all but one of the authentication factors has been fully compromised. An instance $\Pi_i^{\hat{U}}$ with partner id $\hat{U}'$ is **fresh in the first factor** (with forward-secrecy) if and only if none of the following events occur:

1. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}, i)$;
2. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}', j)$, where $\Pi_j^{\hat{U}'}$ is the partner instance of $\Pi_i^{\hat{U}}$;
3. if $\hat{U} \in \mathsf{Clients}$, the adversary has issued either the query $\mathsf{RevealPWC}(\hat{U}, \hat{U}')$ or the query $\mathsf{RevealPWS}(\hat{U}', \hat{U})$ before the Test query, and has issued the query $\mathsf{Send}(\hat{U}, i, M)$ for some string M;
4. if $\hat{U} \in \mathsf{Servers}$, the adversary has issued the query $\mathsf{RevealPWC}(\hat{U}, \hat{U}')$ before the Test query, and has issued the query $\mathsf{Send}(\hat{U}, i, M)$ for some string M.

An instance $\Pi_i^{\hat{U}}$ with partner id $\hat{U}'$ is **fresh in the second factor** (with forward-secrecy) if and only if none of the following events occur:

1. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}, i)$;
2. the adversary has issued the query $\mathsf{RevealSessionKey}(\hat{U}', j)$, where $\Pi_j^{\hat{U}'}$ is the partner instance of $\Pi_i^{\hat{U}}$;

3. if $\hat{U} \in \mathsf{Clients}$, the adversary has issued any query $\mathsf{RevealRe}$ before the Test query, and has issued the query $\mathsf{Send}(\hat{U}, i, M)$ for some string M;

4. if $\hat{U} \in \mathsf{Servers}$, the adversary has issued any query $\mathsf{RevealRe}$ before the Test query, and has issued the query $\mathsf{Send}(\hat{U}, i, M)$ for some string M.

If a session is fresh in both the first and second factors, then it is also fresh in the original notion of freshness for password-authenticated key exchange.

Pointcheval and Zimmer, in their paper on multi-factor authenticated key exchange [PZ07], provide a similar variation on the BPR model. However, their notion of freshness differs from ours: they choose to have a single notion of freshness which describes the security if all but one factor is compromised, and thus the security of the protocol is given solely in terms of the security of the weakest factor. By having two notions of freshness — one for each factor — we are able to more precisely quantify how the security changes depending on which factor is compromised. This additional precision can provide guidance on how particular factors should be deployed and physically secured.

Session key security. The goal of an adversary is to guess the bit b used in the Test query of a fresh in the first (or second) factor session. This corresponds to the ability of an adversary to distinguish the session key from a random string of the same length. Let $\mathsf{Succ}_P^{\text{ake-f1}}(\mathcal{A})$ (respectively, $\mathsf{Succ}_P^{\text{ake-f2}}(\mathcal{A})$) be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh in the first (respectively, second) factor instance $\Pi_i^{\hat{U}}$ that has terminated and $\mathcal{A}$ eventually outputs a bit b', where $b' = b$ and b is the randomly selected bit in the Test query. The **ake-f1 advantage** of $\mathcal{A}$ attacking P is defined to be $\mathsf{Adv}_P^{\text{ake-f1}}(\mathcal{A}) = 2\Pr\left(\mathsf{Succ}_P^{\text{ake-f1}}(\mathcal{A})\right) - 1$, and analogously for the **ake-f2 advantage**.

Since a session that is fresh in both the first and second factors is also fresh in the original ake notion of password-authenticated key exchange, we have that

$$\mathsf{Adv}_P^{\text{ake}}(\mathcal{A}) \leq \min\left\{\mathsf{Adv}_P^{\text{ake-f1}}(\mathcal{A}), \mathsf{Adv}_P^{\text{ake-f2}}(\mathcal{A})\right\} \quad . \tag{4.1}$$

Authentication. We can define similar notions for *client-to-server*, *server-to-client*, and *mutual* authentication for adversaries attempting to break two-factor authentication.

For a protocol P, an adversary violates **client-to-server authentication** if some server oracle terminates and accepts but has no partner oracle; we can refine this notion based on which factor remains uncompromised. For a fixed adversary $\mathcal{A}$, we define $\mathsf{Adv}_P^{\text{c2s-f1}}(\mathcal{A})$ (resp., $\mathsf{Adv}_P^{\text{c2s-f2}}(\mathcal{A})$) to be the probability that a server instance $\Pi_j^{\hat{S}}$ with partner id $\hat{C}$ terminates without having a partner oracle before a $\mathsf{RevealPWC}_P(\hat{C}, \hat{S})$ (resp., $\mathsf{RevealRe}_P(\hat{C}, \hat{S})$) query.

Similarly, an adversary violates **server-to-client authentication** if some client oracle terminates and accepts but has no partner oracle; again, we can refine this notion based on which factor remains uncompromised. We define $\mathsf{Adv}_P^{\text{s2c-f1}}(\mathcal{A})$ (resp., $\mathsf{Adv}_P^{\text{s2c-f2}}(\mathcal{A})$) to be the probability that a client instance $\Pi_i^{\hat{C}}$ with partner id $\hat{S}$ terminates without having a partner oracle before either a $\mathsf{RevealPWC}_P(\hat{C}, \hat{S})$ or $\mathsf{RevealPWS}_P(\hat{S}, \hat{C})$ (resp., $\mathsf{RevealRe}_P(\hat{C}, \hat{S})$) query.

Finally, an adversary violates **mutual authentication** if some oracle terminates and accepts but has no partner oracle. Violation of mutual authentication is equivalent to violation of either client-to-server authentication or server-to-client authentication (or both). We define $\mathsf{Adv}_P^{\text{ma-f}i}(\mathcal{A})$ to be the probability that any instance terminates without having a partner oracle, for each notion i above.

Security. We say that a protocol P is a **secure multi-factor password authenticated key agreement protocol** if there exists a constant δ and a negligible ϵ such that, for all probabilistic algorithms $\mathcal{A}$ running in polynomial time and making at most q_{se} queries of type Send,

$$\mathsf{Adv}_P^{\text{ake-f1}}(\mathcal{A}) \leq \frac{\delta q_{\mathsf{se}}}{|\mathsf{Passwords}|} + \epsilon \quad \text{and} \quad \mathsf{Adv}_P^{\text{ake-f2}}(\mathcal{A}) \leq \frac{\delta q_{\mathsf{se}}}{|\mathsf{Responses}|} + \epsilon \ , \qquad (4.2)$$

and corresponding bounds apply for $\mathsf{Adv}_P^{\text{ma-f1}}(\mathcal{A})$ and $\mathsf{Adv}_P^{\text{ma-f2}}(\mathcal{A})$. Intuitively, this notion of security says that, except with probability ϵ (which is usually a function of the probability of solving a hard computational problem under certain constraints), any polynomially bounded adversary can do little better than doing an online dictionary attack on any unknown factors and can gain no advantage by doing an offline dictionary attack.

4.3.3 Using one-time passwords

While the security of long-term passwords is straightforward, the model for compromise of the short-term password can vary. Our model and proofs allow the second factor to be reused as much as the first factor, but in practice it is desirable to use short-term or one-time passwords for resistance against spyware.

True one-time passwords require the client and server to pre-share a large number of random passwords. However, they offer enhanced security and a multi-factor protocol using true one-time passwords would be secure even if previous one-time passwords were compromised: our notion of fresh in the second factor would not need to preclude all $\mathsf{RevealRe}$ queries prior to the Test query, only that no $\mathsf{RevealRe}$ query happens for the one-time password corresponding to the Test session. All multi-factor password authenticated key exchange protocols secure using arbitrary passwords with our given definition of fresh in the second factor are also secure using one-time passwords with this revised definition of fresh in the second factor.

An alternative approach, such as in Abdalla *et al.* [ACP05a], is to use pseudorandomly generated one-time passwords. The server stores $f^n(\mathsf{seed})$ (where f^n

denotes the n-fold application of f) for a one-way function f, and then the client provides the next verifier $f^{n-1}(\mathsf{seed})$ upon successfully logging in with the current value. In this case, the compromise of the server cannot be used to impersonate the client at the next login, but compromise of the client's seed value (for example, by spyware) leaves the client open to future impersonation. Here, we now have two types of values that could be revealed: individual values $f^i(\mathsf{seed})$, or the value seed itself, modeled by appropriate queries. A multi-factor password-authenticated key exchange protocol should remain secure if only some of the individual values are revealed, and the notion of fresh in the second factor could be suitably altered to suit this case as well.

A one-time password can also be a time-sensitive password that can be generated by a small device such as the RSA SecureID [RSA]. For example, there may be a seed value for each client-server pair, and then, at time t (suitably rounded), the correct value is $f(\mathsf{seed}, t)$ for some pseudorandom function f. While the above technique of Abdalla *et al.* requires passwords to be generated in a very specific way, this setting is more compatible with many of the various deployed one-time password generators. Challenge-response protocols, where for example the client may have a personalized piece of paper with a variety of responses on it to particular challenges, can be modelled in a similar way by having a function $f(\mathsf{seed}, \mathsf{ch})$ that describes the response for a challenge ch and a client-server pair-specific seed. Again, there are two types of values that could be compromised (individual values or the seed) and could be modeled by additional queries.

In Section 3.2.1, we discussed the assumption that passwords are uniformly distributed, and this assumption is likely to be better satisfied when one-time passwords are generated using hardware devices as discussed above.

4.4 MFPAK

MFPAK is the first password-authenticated key exchange protocol to be based on multiple authentication factors. It uses two factors: the first is a long-term user-memorized password, and the second is a short-term password, meant to represent a dynamic one-time response value, although it can be used as a static value as well.

4.4.1 Design ideas

We designed MFPAK by considering two existing one-factor protocols as our building blocks: the asymmetric password protocol mePAK-Z+ (Section 3.3.3.2) for the long-term password, and the symmetric password protocol mePAK (Section 3.3.2.2) for the one-time response values. These two protocols have characteristics that we use in the design of our two-factor protocol and have formal security arguments. Both factors are tightly integrated into the authentication and key exchange processes. The underlying session key agreement comes from a hashed Diffie-Hellman

construct. Authentication in the first factor is done using a digital signature scheme, while in the second factor it is done using hash functions.

Shielded ephemeral key. One of the main efficiency and security gains in the MFPAK protocol comes in the first flow from the client to the server. In this flow, the client shields its ephemeral public key by multiplying it by the first factor and the second factor. The client is made to commit to those values, thereby preventing a malicious client from making an offline dictionary attack later on. Moreover, the server must use the same values to unshield the client's ephemeral public key or Diffie-Hellman key agreement will fail, thereby committing the server to its choice of values. By doing this double shielding operation, the client and server achieve mutual authentication, the client saves expensive operations compared to running two protocols separately, and the authentication secrets are protected.

Digital signature for asymmetric first factor. The asymmetric nature of the first factor comes from using a digital signature scheme. At registration time, the user generates a private key / public key pair for a digital signature scheme, then shields the private key with her password and stores the shielded value on the server. During the login stage of the protocol, the server returns the shielded private key, which the client can unwrap only if she knows the correct password. The client uses the private key to perform a signing operation which the server verifies using the public key. This allows for asymmetry in the first factor: the compromise of the server's database is not enough to impersonate the client to the server without a dictionary attack.

Hash function for symmetric second factor. The hash of the second factor is stored on the server as a symmetric factor. The server proves its knowledge of the second factor by hashing it with the session key; the client does the same.

4.4.2 Protocol specification

The user registration stage of MFPAK is given in Figure 4.2 below. This stage should be completed over a private, authentic channel. We assume the second factor value re is fixed for each client-server pair, but the scenario could allow for a challenge/response or pseudorandomly generated response value, as discussed in Section 4.3.3.

The login stage of MFPAK is given in Figure 4.3. This stage can be completed over a public, untrusted channel. A client $\hat{C}$ initiates the login stage with a server $\hat{S}$. The client knows the password $\mathsf{pw}_{\hat{C},\hat{S}}$ and response $\mathsf{re}_{\hat{C},\hat{S}}$ that was previously established in the registration stage, and the server $\hat{S}$ has its databases $\mathsf{pw}_{\hat{S}}[\hat{C}]$ and $\mathsf{re}_{\hat{S}}[\hat{C}]$, for all $\hat{C} \in \mathsf{Clients}$, of corresponding values. If the response values are meant to be responses to challenges issued by the server, an initial message from

MFPAK User Registration		
	Client $\hat{C}$	Server $\hat{S}$
1.	input username $\hat{C}$	
2.	choose $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$	
3.	choose $\mathsf{re}_{\hat{C},\hat{S}} \in_R \mathsf{Responses}$	
4.	$(v, V) \leftarrow \mathsf{Gen}(1^\kappa)$	
5.	$\gamma = H_1(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}})$	
6.	$v' = H_2(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}}) \oplus v$	
7.	$v'' = H_3(v)$	
8.	$\tau = H_4(\hat{C}, \hat{S}, \mathsf{re}_{\hat{C},\hat{S}})$	
9.	$\xrightarrow{\hat{C},\gamma,V,v',v'',\tau}$	
10.		compute γ^{-1}
11.		compute τ^{-1}
12.		store $\mathsf{pw}_{\hat{S}}[\hat{C}] = (\gamma, \gamma^{-1}, V, v', v'')$
13.		store $\mathsf{re}_{\hat{S}}[\hat{C}] = (\tau, \tau^{-1})$

Figure 4.2: The user registration stage of the MFPAK protocol

the server to the client conveying the challenge can be added to the beginning of the login stage of the protocol.

It will be helpful to be able to refer to the action of a party upon receipt of a message. We use the notation ClientActioni_P and ServerActioni_P to refer to the portion of the protocol P performed by the client or server, respectively, after the ith flow. Thus, MFPAK as described in Figure 4.3 specifies ClientAction0_{MFPAK}, ServerAction1_{MFPAK}, ClientAction2_{MFPAK}, and ServerAction3_{MFPAK}.

4.4.3 Efficiency

In many e-commerce and online banking situations, the performance-limiting factor is the number of connections a server can handle, and this is in turn limited by the number of expensive operations required by the cryptographic protocol. MFPAK can increase security without substantial additional computational burden on the server.

One approach to achieving multi-factor password-authenticated key exchange would be to independently run one protocol for each factor, accept authentication only if all the independent protocols accept, and then compute a single session key dependent on all the independent session keys. This approach can place a heavy computational burden on the client and the server which we can avoid by using a single protocol such as MFPAK.

Figure 4.4 compares the number of expensive operations (group exponentiations, group inversions, and signature generation / verification) performed by a combination of PAK and PAK-Z+ and the MFPAK protocol. On the server side,

MFPAK Login			
	Client $\hat{C}$		Server $\hat{S}$
1.	input username $\hat{C}$		
2.	input password $\mathsf{pw}_{\hat{C},\hat{S}}$		
3.	input response $\mathsf{re}_{\hat{C},\hat{S}}$		
4.	$\gamma = H_1(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}})$		
5.	$\tau = H_4(\hat{C}, \hat{S}, \mathsf{re}_{\hat{C},\hat{S}})$		
6.	$x \in_R \mathbb{Z}_q$		
7.	$X = g^x$		
8.	$m = X \cdot \gamma \cdot \tau$		
9.		$\xrightarrow{\hat{C},m}$	
10.			abort if $\neg\mathsf{Acceptable}(m)$
11.			$y \in_R \mathbb{Z}_q$
12.			$Y = g^y$
13.			lookup $(\gamma, \gamma^{-1}, V, v', v'') = \mathsf{pw}_{\hat{S}}[\hat{C}]$
14.			lookup $(\tau, \tau^{-1}) = \mathsf{re}_{\hat{S}}[\hat{C}]$
15.			$X = m \cdot \gamma^{-1} \cdot \tau^{-1}$
16.			$\sigma = X^y$
17.			$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$
18.			$k = H_5(\mathsf{sid}, \sigma, \gamma, \tau)$
19.			$a' = H_6(\mathsf{sid}, \sigma, \gamma, \tau)$
20.			$a = a' \oplus v'$
21.		$\xleftarrow{Y,k,a,v''}$	
22.	$\sigma = Y^x$		
23.	$\mathsf{sid} = (\hat{C}, \hat{S}, m, Y)$		
24.	abort if $k \neq H_5(\mathsf{sid}, \sigma, \gamma, \tau)$		
25.	$k' = H_7(\mathsf{sid}, \sigma, \gamma, \tau)$		
26.	$a' = H_6(\mathsf{sid}, \sigma, \gamma, \tau)$		
27.	$v' = a' \oplus a$		
28.	$v = H_2(\hat{C}, \hat{S}, \mathsf{pw}_{\hat{C},\hat{S}}) \oplus v'$		
29.	abort if $v'' \neq H_3(v)$		
30.	$s = \mathsf{Sign}_v(\mathsf{sid})$		
31.		$\xrightarrow{k',s}$	
32.			abort if $k' \neq H_7(\mathsf{sid}, \sigma, \gamma, \tau)$
33.			abort if $\neg\mathsf{Verify}_V(\mathsf{sid}, s)$
34.	$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma, \tau)$		$\mathsf{sk} = H_8(\mathsf{sid}, \sigma, \gamma, \tau)$

Figure 4.3: The login stage of the MFPAK protocol

MFPAK has the same number of expensive operations as PAK-Z+, and two fewer operations on the client side due to the removal of group inversion operations as in mePAK and mePAK-Z+. This makes MFPAK much more efficient, in terms of number of expensive operations, than if one were to make a multi-factor scheme simply by running PAK and PAK-Z+ in parallel independently. Although we have

to use slightly larger key sizes (for example, 458 bits for MFPAK compared to 430 bits for PAK-Z+) because of the lack of tightness of the security argument (see Section 4.4.5), the reduced number of operations more than compensates.

Operation	**PAK & PAK-Z+**		**MFPAK**	
	Client	Server	Client	Server
exponentiations	4	4	2	2
inversions	2 (0*)	0	0	0
signature generation	1	0	1	0
signature verification	0	1	0	1

Figure 4.4: Comparison of expensive operations for combined PAK & PAK-Z+ and MFPAK. Entries with * indicate the cost when mePAK and mePAK-Z+ are used instead of PAK and PAK-Z+.

In designing MFPAK for improved efficiency, we only focused on reducing the number of exponentiations and inversions, as the other operations are relatively much less time consuming. It may be possible to also eliminate at least one hash function evaluation on each side, namely those involving H_7 (lines 25 and 32): the test on the server side on line 32 involving H_7 intuitively ought to be satisfied only if the signature verification on line 33 is also satisfied. However, cases 3 and 4 of our formal security analysis in Section 4.4.4 require the use of k' to relate the security in those cases to PAK; an alternative proof may be able to eliminate those tests.

4.4.4 Security analysis of MFPAK

In this section, we show that MFPAK is a secure multi-factor password-authenticated key exchange protocol. The main idea of the argument is to show that, if one factor remains uncompromised, then the difficulty of breaking MFPAK is related to the difficulty of breaking the corresponding one of either mePAK or mePAK-Z+, which in turn are related to solving the Computational Diffie-Hellman problem.

For each of the two factors (password and response), we describe a procedure (specified by a modifier $\mathcal{M}$) to transform an attack by an adversary $\mathcal{A}$ against MFPAK with the specified factor uncompromised into an attack $\mathcal{A}^*$ against the corresponding one of the two underlying protocols (mePAK-Z+ and mePAK, respectively). The transformations are such that, if the oracle instance in MFPAK against which the Test query is directed is fresh in the first (resp., second) factor, then the corresponding oracle instance is also fresh in the corresponding attack on mePAK-Z+ (resp., mePAK). This is possible because of the design of the MFPAK protocol: it essentially runs both mePAK and mePAK-Z+ together while still capturing the security of each independently. This design characteristic allows the relatively straightforward (although lengthy) security argument.

Our formal argument proceeds by considering four cases, two corresponding to the password being uncompromised and two corresponding to the response being

uncompromised. The cases are:

1. First factor uncompromised, $\hat{U}^* \in \mathsf{Clients}$: no $\mathsf{RevealPWC}_{\mathrm{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ or $\mathsf{RevealPWS}_{\mathrm{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ query (Section 4.4.4.1).
2. First factor uncompromised, $\hat{U}^* \in \mathsf{Servers}$: no $\mathsf{RevealPWC}_{\mathrm{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ query (Section 4.4.4.2).
3. Second factor uncompromised, $\hat{U}^* \in \mathsf{Clients}$: no $\mathsf{RevealRe}_{\mathrm{MFPAK}}$ query (Section 4.4.4.3).
4. Second factor uncompromised, $\hat{U}^* \in \mathsf{Servers}$: no $\mathsf{RevealRe}_{\mathrm{MFPAK}}$ query (Section 4.4.4.4).

These four cases are combined probabilistically to give the overall result in Section 4.4.4.5.

Throughout, we assume that passwords are uniformly distributed among the set $\mathsf{Passwords}$ and responses are uniformly distributed among the set $\mathsf{Responses}$.

4.4.4.1 Case 1: Attacking a client instance, first factor uncompromised

This case addresses impersonation of the server when the session being attacked is a client instance and the first factor remains uncompromised.

Let $\mathcal{S}_{\text{mePAK-Z+}}$ denote the mePAK-Z+ system that we will attack. The modifier $\mathcal{M}$ first uniformly at randomly guesses $\hat{U}^* \in_R \mathsf{Clients}$ and $\hat{U}'^* \in_R \mathsf{Servers}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. If the attacker ends up attacking the pair of users the modifier has guessed, then we will show how to transform the attack into an attack $\mathcal{A}^*$ on mePAK-Z+.

Let $\mathsf{GuessCS}$ be the event that the modifier $\mathcal{M}$ correctly guesses $\hat{U}^*$ and $\hat{U}'^*$. Then

$$\Pr(\mathsf{GuessCS}) = \Pr((\hat{U}^* \text{ correct}) \wedge (\hat{U}'^* \text{ correct})) \geq \frac{1}{|\mathsf{Clients}| \cdot |\mathsf{Servers}|} \ . \qquad (4.3)$$

For this case, we assume that neither the query $\mathsf{RevealPWC}_{\mathrm{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ nor the query $\mathsf{RevealPWS}_{\mathrm{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ is issued against $\mathcal{M}$: this case models server impersonation in the first factor, which is why no $\mathsf{RevealPWS}_{\mathrm{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ query is allowed. Furthermore, no $\mathsf{RevealPWC}_{\mathrm{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ is allowed because an adversary can easily recover the verifier $\mathsf{pw}_{\hat{U}'^*}[\hat{U}^*]$ from the password $\mathsf{pw}_{\hat{U}^*,\hat{U}'^*}$ and one interaction with $\hat{U}'^*$.

The modifier $\mathcal{M}$ does the following to convert an MFPAK adversary $\mathcal{A}$ into a mePAK-Z+ adversary $\mathcal{A}^*$.

Password and response preparation. For each $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{re}_{\hat{C},\hat{S}} \in_R \mathsf{Responses}$ and constructs the corresponding $\mathsf{re}_{\hat{S}}[\hat{C}]$. In particular, $\mathcal{M}$ sets $\tau^* = H_4(\hat{U}^*, \hat{U}'^*, \mathsf{re}_{\hat{U}^*,\hat{U}'^*})$ and computes $(\tau^*)^{-1}$. For each $(\hat{C}, \hat{S}) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(\hat{U}^*, \hat{U}'^*)\}$, $\mathcal{M}$ sets $\mathsf{pw}_{\hat{C},\hat{S}} = \mathsf{RevealPWC}_{\text{mePAK-Z+}}(\hat{C}, \hat{S})$ and $\mathsf{pw}_{\hat{S}}[\hat{C}] = \mathsf{RevealPWS}_{\text{mePAK-Z+}}(\hat{S}, \hat{C})$. Of all the password and response values, only $\mathsf{pw}_{\hat{U}^*,\hat{U}'^*}$ and $\mathsf{pw}_{\hat{U}'^*}[\hat{U}^*]$ remain unknown to $\mathcal{M}$ at this point.

Instantiation of mePAK-Z+ **system.** We instantiate the mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ with the following random oracles: $H^*_\ell = H_\ell$, for $\ell = 1, 2, 3$, and

$$H^*_\ell((\hat{C}, \hat{S}, m, Y), \sigma, \gamma) = H_\ell((\hat{C}, \hat{S}, m \cdot \tau^*, Y), \sigma, \gamma, \tau^*) \ , \tag{4.4}$$

for $\ell = 5, 6, 8$.[2] These 'starred' functions are independent random oracles if the corresponding unstarred functions are. The above construction is possible since τ^* is fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

Further, $\mathcal{S}_{\text{mePAK-Z+}}$ is instantiated with the following signature scheme:

$$\begin{aligned}\mathsf{Sign}^*_v((\hat{C}, \hat{S}, m, Y)) &= \mathsf{Sign}_v((\hat{C}, \hat{S}, m \cdot \tau^*, Y)) \\ \mathsf{Verify}^*_V((\hat{C}, \hat{S}, m, Y), s) &= \mathsf{Verify}_V((\hat{C}, \hat{S}, m \cdot \tau^*, Y), s) \ .\end{aligned}$$

Since the transformation that sends $(\hat{C}, \hat{S}, m, Y) \mapsto (\hat{C}, \hat{S}, m \cdot \tau^*, Y)$ is just a permutation, it follows that $(\mathsf{Gen}, \mathsf{Sign}^*, \mathsf{Verify}^*)$ is an eu-cma signature scheme whenever $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$. The main goal is for $\mathcal{M}$ to simulate all queries except for ones that are related to the $\hat{U}^*$ and $\hat{U}'^*$ guessed at the beginning of the case: these queries are passed to the underlying mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$.

1. $\underline{\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{C}, \hat{S})}$:
   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$: Return $\mathsf{pw}_{\hat{C},\hat{S}}$.
   (b) If $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case was incorrect.
2. $\underline{\mathsf{RevealPWS}_{\text{MFPAK}}(\hat{S}, \hat{C})}$:
   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$: Return $\mathsf{pw}_{\hat{S}}[\hat{C}]$.
   (b) If $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case was incorrect.

[2] Note that we do not need to instantiate H^*_4 and H^*_7 because these oracles are not used by mePAK-Z+.

3. $\underline{\mathsf{RevealRe}_{\text{MFPAK}}(\hat{C}, \hat{S})}$: Return $\mathsf{re}_{\hat{C},\hat{S}}$.

4. $\underline{\mathsf{Test}_{\text{MFPAK}}(\hat{U}, i)}$:

   (a) If $\hat{U} = \hat{U}^*$: Send a $\mathsf{Test}_{\text{mePAK-Z+}}(\hat{U}, i)$ query to mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ and return the result to $\mathcal{A}$.

   (b) If $\hat{U} \neq \hat{U}^*$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ at the beginning of this case was incorrect.

5. $\underline{\mathsf{RevealSessionKey}_{\text{MFPAK}}(\hat{U}, i)}$:

   (a) If $\hat{U} = \hat{U}^*$ or $\hat{U} = U'^*$: Send a $\mathsf{RevealSessionKey}_{\text{mePAK-Z+}}(\hat{U}, i)$ query to mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ and return the result to $\mathcal{A}$.

   (b) Otherwise: Return sk for instance $\Pi_i^{\hat{U}}$.

6. $\underline{\mathsf{Execute}_{\text{MFPAK}}(\hat{C}, i, \hat{S}, j)}$:

   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$: $\mathcal{M}$ performs $\mathsf{Execute}_{\text{MFPAK}}(\hat{C}, i, \hat{S}, j)$ with all the values it has and returns the transcript.

   (b) If $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$: $\mathcal{M}$ will use the mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ to obtain a transcript for this query.

      i. Send an $\mathsf{Execute}_{\text{mePAK-Z+}}(\hat{C}, i, \hat{S}, j)$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive $(\hat{C}, m, Y, k, a, v'', s)$.

      ii. Set $\hat{m} = m \cdot \tau^*$.

      iii. Set $\hat{k'} \in_R \text{range}(H_7)$.

      iv. Return $(\hat{C}, \hat{m}, Y, k, a, v'', s, \hat{k'})$ to $\mathcal{A}$.

7. $\underline{\mathsf{Send}_{\text{MFPAK}}(\hat{U}, i, M)}$:

   (a) If M is not a valid protocol message in a meaningful sequence, then abort as would be done in MFPAK.

   (b) If $M =$ ("start", $\hat{S}$) and $(\hat{U}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$: Perform $\textsc{ClientAction0}_{\text{MFPAK}}$ and return $(\hat{U}, m)$.

   (c) If $M =$ ("start", $\hat{S}$) and $(\hat{U}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$:

      i. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, M)$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive $(\hat{U}, m)$.

      ii. Set $\hat{m} = m \cdot \tau^*$.

      iii. Return $(\hat{U}, \hat{m})$.

   (d) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) \neq (\hat{U}^*, \hat{U}'^*)$: Perform $\textsc{ServerAction1}_{\text{MFPAK}}$ and return (Y, k, a, v'').

   (e) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) = (\hat{U}^*, \hat{U}'^*)$:

      i. Set $\hat{m} = m \cdot (\tau^*)^{-1}$.

ii. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, (\hat{C}, \hat{m}))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive (Y, k, a, v'').

iii. Return (Y, k, a, v'').

(f) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') \neq (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform ClientAction2$_{\text{MFPAK}}$ and return (k', s).

(g) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') = (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, (Y, k, a, v''))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive (s).

ii. Set $\hat{k}' \in_R \text{range}(H_7)$ and store.

iii. Return $(\hat{k}', s)$.

(h) If $M = (k', s)$ and $(\hat{U}', \hat{U}) \neq (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform ServerAction3$_{\text{MFPAK}}$.

(i) If $M = (k', s)$ and $(\hat{U}', \hat{U}) = (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Abort if k' is not the same as the $\hat{k}'$ generated in Case 6 above.

ii. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, (s))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$.

Differences from an MFPAK system. We must now analyze the differences between a true MFPAK system and the view presented to the MFPAK adversary $\mathcal{A}$ by the modifier $\mathcal{M}$.

First we note that the distributions of generated passwords and responses exactly match the MFPAK specifications. Furthermore, all the generated passwords exactly match the mePAK-Z+ specifications.

Next, we note that $\mathcal{M}$'s handling of $\mathcal{A}$'s queries precisely matches what an MFPAK system would do except in a small number of cases. The messages received from and forwarded from the use of the mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ can by inspection be seen to match what an MFPAK system would do because $\mathcal{S}_{\text{mePAK-Z+}}$ is using the specially constructed random oracles H^*_ℓ. The differences between $\mathcal{M}$ and what a true MFPAK system would do are as follows:

- $\mathsf{RevealPWC}(\hat{C}, \hat{S})$ when $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$, $\mathsf{RevealPWS}(\hat{S}, \hat{C})$ when $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$, and $\mathsf{Test}(\hat{U}, i)$ when $\hat{U} \neq \hat{U}^*$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK system should not. If $\mathcal{M}$ correctly guessed $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case, then none of these queries would occur, for if one did then the session in which a Test query is directed to $\Pi_i^{\hat{U}^*}$ would not be fresh.

- $\mathsf{Execute}(\hat{C}, i, \hat{S}, j)$ when $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$, $\mathsf{Send}(\hat{U}, i, M)$ when $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') = (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$, and $\mathsf{Send}(\hat{U}, i, M)$ when $M = (k', s)$ and $(\hat{U}, \hat{U}') = (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:
  The modifier $\mathcal{M}$ generated a random value $\hat{k}'$ for this session instead of generating $k' = H_7(\mathsf{sid}, \sigma, \gamma, \tau)$. Since H_7 is a random oracle, this substitution is

distinguishable by the adversary $\mathcal{A}$ if and only if $\mathcal{A}$ queries H_7 on the arguments $\mathsf{sid}, \sigma, \gamma, \tau$. But if that occurs, then $\mathcal{A}$ must know γ. These are the same inputs to the H_8^* oracle used to compute the session key in the mePAK-Z+ simulation $\mathcal{S}_{\text{mePAK-Z+}}$, so the same adversary could distinguish the output of $\mathsf{Test}_{\text{mePAK-Z+}}(\hat{U}^*, i)$ received from $\mathcal{S}_{\text{mePAK-Z+}}$. The latter event corresponds to the event $\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}$, and so the substitution is distinguishable with probability at most $\Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(\mathcal{A}))$.

Let $\mathsf{Dist}_1|\mathsf{GuessCS}$ be the event that the simulation $\mathcal{M}$ is distinguishable from a real MFPAK system from $\mathcal{A}$'s perspective given that the modifier correctly guessed $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case. Then $\Pr(\mathsf{Dist}_1|\mathsf{GuessCS}) \leq 3\Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(\mathcal{A}))$ by the argument above.

Result for case 1. Let $\hat{U}^* \in \mathsf{Clients}$, $\hat{U}'^* \in \mathsf{Servers}$ and let E_1 be the event that neither $\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ nor $\mathsf{RevealPWS}_{\text{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ occurs. The session involving $\hat{U}^*, \hat{U}'^*$ in $\mathcal{S}_{\text{mePAK-Z+}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the first factor Thus, if event E_1 occurs and event $\mathsf{GuessCS}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\text{mePAK-Z+}}$, except with probability at most $\Pr(\mathsf{Dist}_1|\mathsf{GuessCS})$. Therefore,

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.5)$$

where $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + z + 1 + 6q_{\mathsf{ex}} + 4q_{\mathsf{se}}$, $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(2t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. Moreover,

$$\begin{aligned} &\big|\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}) \\ &- \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS})\big| \\ &\qquad \leq \Pr(\mathsf{Dist}_1|\mathsf{GuessCS}) \ . \end{aligned} \quad (4.6)$$

Combining these two expressions yields the following result:

Lemma 4.1 *Let $\hat{U}^* \in \mathsf{Clients}$, $\hat{U}'^* \in \mathsf{Servers}$, and suppose that neither the query $\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ nor the query $\mathsf{RevealPWS}_{\text{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ occurs (which is event E_1). Then*

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_1, \mathsf{GuessCS}) \leq 4\Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.7)$$

where $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + z + 1 + 6q_{\mathsf{ex}} + 4q_{\mathsf{se}}$, $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(2t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}^{\text{s2c-f1}}_{\text{MFPAK}}$.

4.4.4.2 Case 2: Attacking a server instance, first factor uncompromised

This case addresses impersonation of the client when the session being attacked is a server instance and the first factor remains uncompromised.

Let $\mathcal{S}_{\text{mePAK-Z+}}$ denote the mePAK-Z+ system that we will attack. The modifier $\mathcal{M}$ first uniformly at randomly guesses $\hat{U}^* \in_R \mathsf{Servers}$ and $\hat{U}'^* \in_R \mathsf{Clients}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. Let $\mathsf{GuessSC}$ be the event that the modifier $\mathcal{M}$ correctly guesses $\hat{U}^*$ and $\hat{U}'^*$. We note that $\Pr(\mathsf{GuessSC}) = \Pr(\mathsf{GuessCS})$.

For this case, we assume that no $\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ query is issued against $\mathcal{M}$: this case models client impersonation in the first factor, which is why this query is not allowed.

The modifier $\mathcal{M}$ does the following to convert an MFPAK adversary $\mathcal{A}$ into an attack $\mathcal{A}^*$ on mePAK-Z+.

Password and response preparation. For each $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{re}_{\hat{C},\hat{S}} \in_R \mathsf{Responses}$ and constructs the corresponding $\mathsf{re}_{\hat{S}}[\hat{C}]$. In particular, $\mathcal{M}$ sets $\tau^* = H_4(\hat{U}'^*, \hat{U}^*, \mathsf{re}_{\hat{U}^*,\hat{U}'^*})$ and computes $(\tau^*)^{-1}$. For each $(\hat{C}, \hat{S}) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(\hat{U}'^*, \hat{U}^*)\}$, $\mathcal{M}$ sets $\mathsf{pw}_{\hat{C},\hat{S}} = \mathsf{RevealPWC}_{\text{mePAK-Z+}}(\hat{C}, \hat{S})$ and $\mathsf{pw}_{\hat{S}}[\hat{C}] = \mathsf{RevealPWS}_{\text{mePAK-Z+}}(\hat{S}, \hat{C})$. Finally, $\mathcal{M}$ sets $\mathsf{pw}_{\hat{U}^*}[\hat{U}'^*] = \mathsf{RevealPWS}_{\text{mePAK-Z+}}(\hat{U}^*, \hat{U}'^*)$ (but only if $\mathcal{M}$ receives a $\mathsf{RevealPWS}_{\text{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ query). Of all the password and response values, only $\mathsf{pw}_{\hat{U}'^*,\hat{U}^*}$ remains unknown to $\mathcal{M}$.

Instantiation of mePAK-Z+ **system.** We instantiate the mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ with the following random oracles: $H^*_\ell = H_\ell$, for $\ell = 1, 2, 3$, and

$$H^*_\ell((\hat{C}, \hat{S}, m, Y), \sigma, \gamma) = H_\ell((\hat{C}, \hat{S}, m \cdot \tau^*, Y), \sigma, \gamma, \tau^*) \ , \tag{4.8}$$

for $\ell = 5, 6, 8$. These 'starred' functions are independent random oracles if the corresponding unstarred functions are. The above construction is possible since τ^* is fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

Further, $\mathcal{S}_{\text{mePAK-Z+}}$ is instantiated with the following signature scheme:

$$\begin{aligned} \mathsf{Sign}^*_v((\hat{C}, \hat{S}, m, Y)) &:= \mathsf{Sign}_v((\hat{C}, \hat{S}, m \cdot (\tau^*)^{-1}, Y)) \\ \mathsf{Verify}^*_V((\hat{C}, \hat{S}, m, Y), s) &:= \mathsf{Verify}_V((\hat{C}, \hat{S}, m \cdot (\tau^*)^{-1}, Y), s) \ . \end{aligned}$$

As before, we note that $(\mathsf{Gen}, \mathsf{Sign}^*, \mathsf{Verify}^*)$ is an eu-cma signature scheme.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$.

1. $\underline{\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{C}, \hat{S})}$:
   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}'^*, \hat{U}^*)$: Return $\mathsf{pw}_{\hat{C},\hat{S}}$.
   (b) If $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case was incorrect.

2. $\underline{\mathsf{RevealPWS}_{\text{MFPAK}}(\hat{S}, \hat{C})}$:
   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}'^*, \hat{U}^*)$: Return $\mathsf{pw}_{\hat{S}}[\hat{C}]$.
   (b) If $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$:
      i. Send a $\mathsf{RevealPWS}_{\text{mePAK-Z+}}(\hat{U}^*, \hat{U}'^*)$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive $\mathsf{pw}_{\hat{U}^*}[\hat{U}'^*]$.
      ii. Return $\mathsf{pw}_{\hat{U}^*}[\hat{U}'^*]$.
3. $\underline{\mathsf{RevealRe}_{\text{MFPAK}}(\hat{C}, \hat{S})}$: Return $\mathsf{re}_{\hat{C},\hat{S}}$.
4. $\underline{\mathsf{Test}_{\text{MFPAK}}(\hat{U}, i)}$:
   (a) If $\hat{U} = \hat{U}^*$: Send a $\mathsf{Test}_{\text{mePAK-Z+}}(\hat{U}, i)$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and return the result to $\mathcal{A}$.
   (b) If $\hat{U} \neq \hat{U}^*$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ at the beginning of this case was incorrect.
5. $\underline{\mathsf{RevealSessionKey}_{\text{MFPAK}}(\hat{U}, i)}$:
   (a) If $\hat{U} = \hat{U}^*$ or $\hat{U} = U'^*$: Send a $\mathsf{RevealSessionKey}_{\text{mePAK-Z+}}(\hat{U}, i)$ query to mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ and return the result to $\mathcal{A}$.
   (b) Otherwise: Return sk for instance $\Pi_i^{\hat{U}}$.
6. $\underline{\mathsf{Execute}_{\text{MFPAK}}(\hat{C}, i, \hat{S}, j)}$:
   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}'^*, \hat{U}^*)$: $\mathcal{M}$ performs $\mathsf{Execute}_{\text{MFPAK}}(\hat{C}, i, \hat{S}, j)$ with all the values it has and returns the transcript.
   (b) If $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$: $\mathcal{M}$ will use the mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ to obtain a transcript for this query.
      i. Send an $\mathsf{Execute}_{\text{mePAK-Z+}}(\hat{C}, i, \hat{S}, j))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive $(\hat{C}, m, Y, k, a, v'', s)$.
      ii. Set $\hat{m} = m \cdot \tau^*$.
      iii. Set $\hat{k}' \in_R \text{range}(H_7)$.
      iv. Return $(\hat{C}, \hat{m}, Y, k, a, v'', \hat{k}', s)$.
7. $\underline{\mathsf{Send}_{\text{MFPAK}}(\hat{U}, i, M)}$:
   (a) If M is not a valid protocol message in a meaningful sequence, then abort as would be done in MFPAK.
   (b) If $M =$ ("start", $\hat{S}$): Perform $\text{ClientAction0}_{\text{MFPAK}}$ and return $(\hat{U}, m)$.
   (c) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) \neq (\hat{U}'^*, \hat{U}^*)$: Perform $\text{ServerAction1}_{\text{MFPAK}}$ and return (Y, k, a, v'').
   (d) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) = (\hat{U}'^*, \hat{U}^*)$:

i. Set $\hat{m} = m \cdot (\gamma^*)^{-1}$.

ii. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, (\hat{C}, \hat{m}))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive (Y, k, a, v'').

iii. Return (Y, k, a, v'').

(e) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') \neq (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform ClientAction2$_{\text{MFPAK}}$ and return (k', s).

(f) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') = (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, (Y, k, a, v''))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$ and receive (s).

ii. Set $\hat{k}' \in_R \text{range}(H_7)$ and store.

iii. Return $(\hat{k}', s)$.

(g) If $M = (k', s)$ and $(\hat{U}', \hat{U}) \neq (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform ServerAction3$_{\text{MFPAK}}$.

(h) If $M = (k', s)$ and $(\hat{U}', \hat{U}) = (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Abort if k' is not the same as the $\hat{k}'$ generated in Case 5 above.

ii. Send a $\mathsf{Send}_{\text{mePAK-Z+}}(\hat{U}, i, (s))$ query to $\mathcal{S}_{\text{mePAK-Z+}}$.

Differences from MFPAK system. We must now analyze the differences between a true MFPAK system and the view presented to the MFPAK adversary $\mathcal{A}$ by the modifier $\mathcal{M}$.

First we note that the distributions of generated passwords and responses exactly match the MFPAK specifications. Furthermore, all the generated passwords exactly match the mePAK-Z+ specifications.

Next, we note that $\mathcal{M}$'s handling of $\mathcal{A}$'s queries precisely matches what an MFPAK system would do except in a small number of cases. The messages received from and forwarded from the use of the mePAK-Z+ system $\mathcal{S}_{\text{mePAK-Z+}}$ can by inspection be seen to match what the MFPAK system would do because $\mathcal{S}_{\text{mePAK-Z+}}$ is using the specially constructed random oracles H^*. The differences between $\mathcal{M}$ and what a true MFPAK system would are as follows:

- $\mathsf{RevealPWC}(\hat{C}, \hat{S})$ when $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$ and $\mathsf{Test}(\hat{U}, i)$ when $\hat{U} \neq \hat{U}^*$: The modifier $\mathcal{M}$ aborts here, while a true MFPAK system should not. If $\mathcal{M}$ correctly guessed $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case, then this query would never occur, for if it did then the session in which a Test query is directed to $\Pi_i^{\hat{U}^*}$ would not be fresh.

- $\mathsf{Execute}(\hat{C}, i, \hat{S}, j)$ when $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$, $\mathsf{Send}(\hat{U}, i, M)$ when $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') = (\hat{U}'^*, \hat{U}^*)$ where $\hat{U}'$ is the partner of $\hat{U}$, and $\mathsf{Send}(\hat{U}, i, M)$ when $M = (k', s)$ and $(\hat{U}', \hat{U}) = (\hat{U}'^*, \hat{U}^*)$ where $\hat{U}'$ is the partner of $\hat{U}$:

The modifier $\mathcal{M}$ generated a random value $\hat{k'}$ for this session instead of generating $k' = H_7(\mathsf{sid}, \sigma, \gamma, \tau)$. Since H_7 is a random oracle, this substitution is distinguishable by the adversary $\mathcal{A}$ if and only if $\mathcal{A}$ queries H_7 on the arguments $\mathsf{sid}, \sigma, \gamma, \tau$. But if that occurs, then $\mathcal{A}$ must know γ. These are the same inputs to the H_8^* oracle used to compute the session key in the mePAK-Z+ simulation $\mathcal{S}_{\text{mePAK-Z+}}$, so the same adversary could distinguish the output of $\mathsf{Test}(\hat{U}^*, i)$ received from $\mathcal{S}_{\text{mePAK-Z+}}$. The latter event corresponds to the event $\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}$, and so the substitution is distinguishable with probability at most $\Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(\mathcal{A}))$.

Let $\mathsf{Dist}_2|\mathsf{GuessSC}$ be the event that the simulation $\mathcal{M}$ is distinguishable from a real MFPAK system from $\mathcal{A}$'s perspective given that the modifier correctly guessed $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case. Then $\Pr(\mathsf{Dist}_2|\mathsf{GuessSC}) \leq 3\Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(\mathcal{A}))$ by the argument above.

Result for case 2. Let $\hat{U}^* \in \mathsf{Servers}$, $\hat{U}'^* \in \mathsf{Clients}$ and let E_2 be the event that query $\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ does not occur. The session involving $\hat{U}'^*, \hat{U}^*$ in $\mathcal{S}_{\text{mePAK-Z+}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the first factor. Thus, if event E_2 occurs and event $\mathsf{GuessSC}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\text{mePAK-Z+}}$, except with probability at most $\Pr(\mathsf{Dist}_2|\mathsf{GuessSC})$, since. Therefore,

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_2, \mathsf{GuessSC}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.9)$$

where $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + 1 + z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. Moreover,

$$\begin{aligned} &\big|\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_2, \mathsf{GuessSC}) \\ &- \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_2, \mathsf{GuessSC})\big| \\ &\qquad \leq \Pr(\mathsf{Dist}_2|\mathsf{GuessSC}) \ . \end{aligned} \quad (4.10)$$

Combining these two expressions yields the following result:

Lemma 4.2 *Let $\hat{U}^* \in \mathsf{Servers}$, $\hat{U}'^* \in \mathsf{Clients}$, and suppose that the query $\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ does not occur (which is event E_2). Then*

$$\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})|\mathsf{E}_2, \mathsf{GuessSC}) \leq 4\Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.11)$$

where $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + 1 + z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}^{\text{c2s-f1}}_{\text{MFPAK}}$.

4.4.4.3 Case 3: Attacking a client instance, second factor uncompromised

This case addresses impersonation of the server when the session being attacked is a client instance and the second factor remains uncompromised.

Let $\mathcal{S}_{\text{mePAK}}$ denote the mePAK system that we will attack. The modifier $\mathcal{M}$ first uniformly at randomly guesses $\hat{U}^* \in_R \mathsf{Clients}$ and $\hat{U}'^* \in_R \mathsf{Servers}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. The event that the modifier correctly guesses these values is $\mathsf{GuessCS}$ and $\Pr(\mathsf{GuessCS})$ is given in (4.3).

For this case, we assume that no $\mathsf{RevealRe}_{\text{MFPAK}}(\hat{U}^*, \hat{U}'^*)$ query is issued against $\mathcal{M}$: this case models server impersonation in the second factor, which is why this query is not allowed.

The modifier $\mathcal{M}$ does the following to convert an MFPAK adversary $\mathcal{A}$ into a mePAK adversary $\mathcal{A}^*$.

Password and response preparation. For each $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$ and $(v, V) \xleftarrow{R} \mathsf{Gen}(1^\kappa)$, and constructs the corresponding $\mathsf{pw}_{\hat{S}}[\hat{C}]$. In particular, $\mathcal{M}$ sets $\gamma^* = H_1(\hat{U}^*, \hat{U}'^*, \mathsf{pw}_{\hat{U}^*,\hat{U}'^*})$ and computes $(\gamma^*)^{-1}$. For each $(\hat{C}, \hat{S}) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(\hat{U}^*, \hat{U}'^*)\}$, $\mathcal{M}$ sets $\mathsf{re}_{\hat{C},\hat{S}} \in_R \mathsf{Responses}$. Of all the password and response values, only $\mathsf{re}_{\hat{U}^*,\hat{U}'^*}$ remains unknown to $\mathcal{M}$ at this point.

Instantiation of mePAK **system.** We instantiate the mePAK system $\mathcal{S}_{\text{mePAK}}$ with the following random oracles: $H_4^* = H_4$,

$$H_5^*((\hat{C}, \hat{S}, m, Y), \sigma, \tau) = H_5((\hat{C}, \hat{S}, m{\cdot}\gamma^*, Y), \sigma, \gamma^*, \tau) || H_6((\hat{C}, \hat{S}, m{\cdot}\gamma^*, Y), \sigma, \gamma^*, \tau) \ , \tag{4.12}$$

and

$$H_\ell^*((\hat{C}, \hat{S}, m, Y), \sigma, \tau) = H_\ell((\hat{C}, \hat{S}, m \cdot \gamma^*, Y), \sigma, \gamma^*, \tau) \tag{4.13}$$

for $\ell = 7, 8$. These 'starred' functions are independent random oracles if the component unstarred functions are. The above construction is possible since γ^* is fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$.

1. $\underline{\mathsf{RevealPWC}_{\text{MFPAK}}(\hat{C}, \hat{S})}$: Return $\mathsf{pw}_{\hat{C},\hat{S}}$.
2. $\underline{\mathsf{RevealPWS}_{\text{MFPAK}}(\hat{S}, \hat{C})}$: Return $\mathsf{pw}_{\hat{S}}[\hat{C}]$.
3. $\underline{\mathsf{RevealRe}_{\text{MFPAK}}(\hat{C}, \hat{S})}$:
   (a) If $(\hat{C}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$: Return $\mathsf{re}_{\hat{C},\hat{S}}$.
   (b) If $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case was incorrect.
4. $\underline{\mathsf{Test}_{\text{MFPAK}}(\hat{U}, i)}$:

(a) If $\hat{U} = \hat{U}^*$: Send a $\mathsf{Test}_{\text{mePAK}}(\hat{U}, i)$ query to $\mathcal{S}_{\text{mePAK}}$ and return the result to $\mathcal{A}$.

(b) If $\hat{U} \neq \hat{U}^*$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ at the beginning of this case was incorrect.

5. $\underline{\mathsf{RevealSessionKey}_{\text{MFPAK}}(\hat{U}, i)}$:

(a) If $\hat{U} = \hat{U}^*$ or $\hat{U} = U'^*$: Send a $\mathsf{RevealSessionKey}_{\text{mePAK}}(\hat{U}, i)$ query to $\mathcal{S}_{\text{mePAK}}$ and return the result to $\mathcal{A}$.

(b) Otherwise: Return sk for instance $\Pi_i^{\hat{U}}$.

6. $\underline{\mathsf{Execute}_{\text{MFPAK}}(\hat{C}, i, \hat{S}, j)}$:

(a) If $(\hat{C}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$: $\mathcal{M}$ performs $\mathsf{Execute}_{\text{MFPAK}}(\hat{C}, i, \hat{S}, j)$ with all the values it has and returns the transcript.

(b) If $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$: $\mathcal{M}$ will use the mePAK system $\mathcal{S}_{\text{mePAK}}$ to help construct a full transcript by performing the following sequence of operations:

i. Send an $\mathsf{Execute}_{\text{mePAK}}(\hat{C}, i, \hat{S}, j)$ query to $\mathcal{S}_{\text{mePAK}}$ and receive $(\hat{C}, m, Y, k, k')$.

ii. Set

$$\begin{aligned} \hat{m} &= m \cdot \gamma^* \\ \hat{k} &= \text{substring}_1(k) \\ \hat{a'} &= \text{substring}_2(k) \\ \hat{a} &= \hat{a'} \oplus v' \\ \hat{s} &= \mathsf{Sign}_v((\hat{C}, \hat{S}, \hat{m}, Y)) \ . \end{aligned}$$

iii. Return $(\hat{C}, \hat{m}, Y, \hat{k}, \hat{a}, v'', k', s)$ to $\mathcal{A}$.

7. $\underline{\mathsf{Send}_{\text{MFPAK}}(\hat{U}, i, M)}$:

(a) If M is not a valid protocol message in a meaningful sequence, then abort as would be done in MFPAK.

(b) If $M = (\text{“start”}, \hat{S})$ and $(\hat{U}, \hat{S}) \neq (\hat{U}^*, \hat{U}'^*)$:
Perform $\textsc{ClientAction0}_{\text{MFPAK}}$ and return $(\hat{U}, m)$.

(c) If $M = (\text{“start”}, \hat{S})$ and $(\hat{U}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$:

i. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (\text{“start”}, \hat{S}))$ query to $\mathcal{S}_{\text{mePAK}}$ and receive $(\hat{U}, m)$.

ii. Set $\hat{m} = m \cdot \gamma^*$ and store.

iii. Return $(\hat{U}, \hat{m})$.

(d) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) \neq (\hat{U}^*, \hat{U}'^*)$: Perform ServerAction1$_{\text{MFPAK}}$ and return (Y, k, a, v'').

(e) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) = (\hat{U}^*, \hat{U}'^*)$:

i. Set $\hat{m} = m \cdot (\gamma^*)^{-1}$ and store.

ii. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (\hat{C}, \hat{m}))$ query to $\mathcal{S}_{\text{mePAK}}$ and receive (Y, k).

iii. Set

$$\begin{aligned} \hat{k} &= \text{substring}_1(k) \\ \hat{a'} &= \text{substring}_2(k) \\ \hat{a} &= \hat{a'} \oplus v' \ . \end{aligned}$$

iv. Return $(Y, \hat{k}, \hat{a}, v'')$.

(f) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') \neq (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform ClientAction2$_{\text{MFPAK}}$ and return (k', s).

(g) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') = (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Set $\hat{a'} = a \oplus v'$ and $\hat{k} = k \mid\mid \hat{a'}$.

ii. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (Y, \hat{k}))$ query to $\mathcal{S}_{\text{mePAK}}$ and receive (k') or abort.

iii. Set $\hat{s} = \mathsf{Sign}_v((\hat{U}^*, \hat{U}'^*, \hat{m}, Y))$ where $\hat{m}$ is the value generated in step 2.

iv. Return $(k', \hat{s})$.

(h) If $M = (k', s)$ and $(\hat{U}', \hat{U}) \neq (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform ServerAction3$_{\text{MFPAK}}$.

(i) If $M = (k', s)$ and $(\hat{U}', \hat{U}) = (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Abort if $\neg\mathsf{Verify}_V((\hat{U}^*, \hat{U}'^*, \hat{m}, Y), s)$ where $\hat{m}$ is the value generated in step 4.

ii. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (k'))$ query to $\mathcal{S}_{\text{mePAK}}$.

Differences from MFPAK system. We must now analyze the differences between a true MFPAK system and the view presented to the MFPAK adversary $\mathcal{A}$ by the modifier $\mathcal{M}$.

First we note that the distributions of generated passwords and responses exactly match the MFPAK specifications. Furthermore, all the generated responses exactly match the mePAK specifications.

Next, we note that $\mathcal{M}$'s handling of $\mathcal{A}$'s queries precisely matches what an MFPAK system would do except in a small number of cases. The messages received from and forwarded from the use of the mePAK system $\mathcal{S}_{\text{mePAK}}$ can by inspection be seen to match what the MFPAK system would do because $\mathcal{S}_{\text{mePAK}}$ is using the specially constructed random oracles H^*. The differences between $\mathcal{M}$ and what a true MFPAK system would do are as follows:

- $\mathsf{RevealRe}(\hat{C}, \hat{S})$ when $(\hat{C}, \hat{S}) = (\hat{U}^*, \hat{U}'^*)$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK system should not. If $\mathcal{M}$ correctly guessed $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case, then this query would never occur, for if it did then the session in which a Test query is directed to $\Pi_i^{\hat{U}^*}$ would not be fresh.

- $\mathsf{Test}(\hat{U}, i)$ when $\hat{U} \neq \hat{U}^*$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK system should not. If $\mathcal{M}$ correctly guessed $\hat{U}^*$ at the beginning of this case, then this query would never occur, for if it did then the session in which a Test query is directed to $\Pi_i^{\hat{U}^*}$ would not be fresh.

In particular, we note that, when the event $\mathsf{GuessCS}$ occurs, the handling of $\mathcal{A}$'s $\mathsf{Execute}$ and Send queries exactly matches the behaviour and distributions of a true MFPAK system.

Result for case 3. Let $\hat{U}^* \in \mathsf{Clients}$, $\hat{U}'^* \in \mathsf{Servers}$ and let E_3 be the event that no $\mathsf{RevealRe}_{\mathrm{MFPAK}}$ query occurs. If event E_3 occurs, the session involving $\hat{U}^*, \hat{U}'^*$ in $\mathcal{S}_{\mathrm{mePAK}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the second factor. Thus, if event E_3 occurs and event $\mathsf{GuessCS}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\mathrm{mePAK}}$. Therefore,

$$\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessCS}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\mathrm{mePAK}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.14)$$

where $q'_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. Moreover,

$$\begin{aligned} &\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathrm{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessCS}) \\ &= \Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessCS}) \ . \end{aligned} \quad (4.15)$$

Combining these two expressions yields the following result:

Lemma 4.3 *Let $\hat{U}^* \in \mathsf{Clients}$, $\hat{U}'^* \in \mathsf{Servers}$, and suppose that no $\mathsf{RevealRe}_{\mathrm{MFPAK}}$ query occurs (which is event E_3). Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type Send and $\mathsf{Execute}$, respectively, and at most q_{ro} random oracle queries. Then*

$$\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathrm{MFPAK}}(\mathcal{A}) | \mathsf{E}_3, \mathsf{GuessCS}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\mathrm{mePAK}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.16)$$

where $q'_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, and a similar bound exists for $\mathsf{Adv}^{\text{s2c-f2}}_{\mathrm{MFPAK}}$.

4.4.4.4 Case 4: Attacking a server instance, second factor uncompromised

This case addresses impersonation of the client when the session being attacked is a server instance and the second factor remains uncompromised.

The modifier $\mathcal{M}$ first uniformly at randomly guesses $\hat{U}^* \in_R \mathsf{Servers}$ and $\hat{U}'^* \in_R \mathsf{Clients}$ as its guess of who the adversary $\mathcal{A}$ will end up attacking. The event that the modifier correctly guesses these values is $\mathsf{GuessSC}$ which has the same probability as $\mathsf{GuessCS}$.

For this case, we assume that no $\mathsf{RevealRe}_{\mathrm{MFPAK}}(\hat{U}'^*, \hat{U}^*)$ query is issued against $\mathcal{M}$: this case models client impersonation in the second factor, which is why this query is not allowed.

The modifier $\mathcal{M}$ does the following to convert an MFPAK adversary $\mathcal{A}$ into a mePAK adversary $\mathcal{A}^*$.

Password and response preparation. For each $(\hat{C}, \hat{S}) \in \mathsf{Clients} \times \mathsf{Servers}$, $\mathcal{M}$ sets $\mathsf{pw}_{\hat{C},\hat{S}} \in_R \mathsf{Passwords}$ and $(v, V) \xleftarrow{R} \mathsf{Gen}(1^\kappa)$, and constructs the corresponding $\mathsf{pw}_{\hat{S}}[\hat{C}]$. In particular, $\mathcal{M}$ sets $\gamma^* = H_1(\hat{U}'^*, \hat{U}^*, \mathsf{pw}_{\hat{U}'^*,\hat{U}^*})$ and computes $(\gamma^*)^{-1}$. For each $(\hat{C}, \hat{S}) \in (\mathsf{Clients} \times \mathsf{Servers}) \setminus \{(\hat{U}'^*, \hat{U}^*)\}$, $\mathcal{M}$ sets $\mathsf{re}_{\hat{C},\hat{S}} \in_R \mathsf{Responses}$. Of all the password and response values, only $\mathsf{re}_{\hat{U}'^*,\hat{U}^*}$ remains unknown to $\mathcal{M}$ at this point.

Instantiation of mePAK **system.** $\mathcal{M}$ instantiates the mePAK system $\mathcal{S}_{\mathrm{mePAK}}$ with the following random oracles: $H_4^* = H_4$,

$$H_5^*((\hat{C}, \hat{S}, m, Y), \sigma, \tau) = H_5((\hat{C}, \hat{S}, m{\cdot}\gamma^*, Y), \sigma, \gamma^*, \tau) || H_6((\hat{C}, \hat{S}, m{\cdot}\gamma^*, Y), \sigma, \gamma^*, \tau) \ , \tag{4.17}$$

and

$$H_\ell^*((\hat{C}, \hat{S}, m, Y), \sigma, \tau) = H_\ell((\hat{C}, \hat{S}, m \cdot \gamma^*, Y), \sigma, \gamma^*, \tau) \ , \tag{4.18}$$

for $\ell = 7, 8$. These 'starred' functions are independent random oracles if the component unstarred functions are. The above construction is possible since γ^* is fixed and known to $\mathcal{M}$ because of the guesses made at the beginning of this case. By using a concatenation of random oracles, the mePAK system computes the values we need in $\mathcal{M}$'s handling of $\mathsf{Execute}$ queries.

$\mathcal{M}$'s handling of $\mathcal{A}$'s queries. The modifier $\mathcal{M}$ performs the following modifications to the queries of $\mathcal{A}$.

1. $\underline{\mathsf{RevealPWC}_{\mathrm{MFPAK}}(\hat{C}, \hat{S})}$: Return $\mathsf{pw}_{\hat{C},\hat{S}}$.
2. $\underline{\mathsf{RevealPWS}_{\mathrm{MFPAK}}(\hat{S}, \hat{C})}$: Return $\mathsf{pw}_{\hat{S}}[\hat{C}]$.
3. $\underline{\mathsf{RevealRe}_{\mathrm{MFPAK}}(\hat{C}, \hat{S})}$:

(a) If $(\hat{C}, \hat{S}) \neq (\hat{U}'^*, \hat{U}^*)$: Return $\mathsf{re}_{\hat{C},\hat{S}}$.

(b) If $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case was incorrect.

4. $\underline{\mathsf{Test}_{\mathrm{MFPAK}}(\hat{U}, i)}$:

(a) If $\hat{U} = \hat{U}^*$: Send a $\mathsf{Test}_{\mathrm{mePAK}}(\hat{U}, i)$ query to $\mathcal{S}_{\mathrm{mePAK}}$ and return the result to $\mathcal{A}$.

(b) If $\hat{U} \neq \hat{U}^*$: Abort; if this query occurs, then $\mathcal{M}$'s guess of $\hat{U}^*$ at the beginning of this case was incorrect.

5. $\underline{\mathsf{RevealSessionKey}_{\mathrm{MFPAK}}(\hat{U}, i)}$:

(a) If $\hat{U} = \hat{U}^*$ or $\hat{U} = U'^*$: Send a $\mathsf{RevealSessionKey}_{\mathrm{mePAK}}(\hat{U}, i)$ query to $\mathcal{S}_{\mathrm{mePAK}}$ and return the result to $\mathcal{A}$.

(b) Otherwise: Return sk for instance $\Pi_i^{\hat{U}}$.

6. $\underline{\mathsf{Execute}_{\mathrm{MFPAK}}(\hat{C}, i, \hat{S}, j)}$:

(a) If $(\hat{C}, \hat{S}) \neq (\hat{U}'^*, \hat{U}^*)$: $\mathcal{M}$ performs $\mathsf{Execute}_{\mathrm{MFPAK}}(\hat{C}, i, \hat{S}, j)$ with all the values it has and returns the transcript.

(b) If $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$: $\mathcal{M}$ will use $\mathcal{S}_{\mathrm{mePAK}}$ to help construct a full transcript by performing the following sequence of operations:

i. Send an $\mathsf{Execute}_{\mathrm{mePAK}}(\hat{C}, i, \hat{S}, j)$ query to $\mathcal{S}_{\mathrm{mePAK}}$ and receive $(\hat{C}, m, Y, k, k')$.

ii. Set

$$\begin{aligned} \hat{m} &= m \cdot \gamma^* \\ \hat{k} &= \mathrm{substring}_1(k) \\ \hat{a}' &= \mathrm{substring}_2(k) \\ \hat{a} &= \hat{a}' \oplus v' \\ \hat{s} &= \mathsf{Sign}_v((\hat{C}, \hat{S}, \hat{m}, Y)) \ . \end{aligned}$$

iii. Return $(\hat{C}, \hat{m}, Y, \hat{k}, \hat{a}, v'', k', s)$ to $\mathcal{A}$.

7. $\underline{\mathsf{Send}_{\mathrm{MFPAK}}(\hat{U}, i, M)}$

(a) If M is not a valid protocol message in a meaningful sequence, then abort as would be done in MFPAK.

(b) If $M =$ ("start", $\hat{S}$) and $(\hat{U}, \hat{S}) \neq (\hat{U}'^*, \hat{U}^*)$:
Perform ClientAction0$_{\mathrm{MFPAK}}$ and return $(\hat{U}, m)$.

(c) If $M =$ ("start", $\hat{S}$) and $(\hat{U}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$:

i. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (\text{“start”}, \hat{S}))$ query to $\mathcal{S}_{\text{mePAK}}$ and receive $(\hat{U}, m)$.

ii. Set $\hat{m} = m \cdot \gamma^*$ and store.

iii. Return $(\hat{U}, \hat{m})$.

(d) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) \neq (\hat{U}'^*, \hat{U}^*)$: Perform $\textsc{ServerAction1}_{\text{MFPAK}}$ and return (Y, k, a, v'').

(e) If $M = (\hat{C}, m)$ and $(\hat{C}, \hat{U}) = (\hat{U}'^*, \hat{U}^*)$:

i. Set $\hat{m} = m \cdot (\gamma^*)^{-1}$ and store.

ii. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (\hat{C}, \hat{m}))$ query to $\mathcal{S}_{\text{mePAK}}$ and receive (Y, k).

iii. Set

$$\begin{aligned} \hat{k} &= \text{substring}_1(k) \\ \hat{a'} &= \text{substring}_2(k) \\ \hat{a} &= \hat{a'} \oplus v' \ . \end{aligned}$$

iv. Return $(Y, \hat{k}, \hat{a}, v'')$.

(f) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') \neq (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform $\textsc{ClientAction2}_{\text{MFPAK}}$ and return (k', s).

(g) If $M = (Y, k, a, v'')$ and $(\hat{U}, \hat{U}') = (\hat{U}'^*, \hat{U}^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Set $\hat{a'} = a \oplus v'$ and $\hat{k} = k \mid\mid \hat{a'}$.

ii. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (Y, \hat{k}))$ query to $\mathcal{S}_{\text{mePAK}}$ and receive (k') or abort.

iii. Set $\hat{s} = \mathsf{Sign}_v((\hat{U}'^*, \hat{U}^*, \hat{m}, Y))$ where $\hat{m}$ is the value generated in step 2.

iv. Return $(\hat{k'}, \hat{s})$.

(h) If $M = (k', s)$ and $(\hat{U}, \hat{U}') \neq (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$: Perform $\textsc{ServerAction3}_{\text{MFPAK}}$.

(i) If $M = (k', s)$ and $(\hat{U}, \hat{U}') = (\hat{U}^*, \hat{U}'^*)$, where $\hat{U}'$ is the partner of $\hat{U}$:

i. Abort if $\neg\mathsf{Verify}_V((\hat{U}'^*, \hat{U}^*, \hat{m}, Y), s)$ where $\hat{m}$ is the value generated in step 4.

ii. Send a $\mathsf{Send}_{\text{mePAK}}(\hat{U}, i, (k'))$ query to $\mathcal{S}_{\text{mePAK}}$.

Differences from MFPAK system. We must now analyze the differences between a true MFPAK system and the view presented to the MFPAK adversary $\mathcal{A}$ by the modifier $\mathcal{M}$.

First we note that the distributions of generated passwords and responses exactly match the MFPAK specifications. Furthermore, all the generated passwords exactly match the mePAK specifications.

Next, we note that $\mathcal{M}$'s handling of $\mathcal{A}$'s queries precisely matches what an MFPAK system would do except in a small number of cases. The messages received from and forwarded from the use of the mePAK system $\mathcal{S}_{\text{mePAK}}$ can by inspection be seen to match what the MFPAK system would do because $\mathcal{S}_{\text{mePAK}}$ is using the specially constructed random oracles H^*. The differences between $\mathcal{M}$ and what a true MFPAK system would do are as follows:

- $\mathsf{RevealRe}(\hat{C}, \hat{S})$ when $(\hat{C}, \hat{S}) = (\hat{U}'^*, \hat{U}^*)$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK system should not. If $\mathcal{M}$ correctly guessed $\hat{U}^*$ and $\hat{U}'^*$ at the beginning of this case, then this query would never occur, for if it did then the session in which a Test query is directed to $\Pi_i^{\hat{U}^*}$ would not be fresh.

- $\mathsf{Test}(\hat{U}, i)$ when $\hat{U} \neq \hat{U}^*$:
  The modifier $\mathcal{M}$ aborts here, while a true MFPAK system should not. If $\mathcal{M}$ correctly guessed $\hat{U}^*$ at the beginning of this case, then these queries would never occur, for if they did then the session in which a Test query is directed to $\Pi_i^{\hat{U}^*}$ would not be fresh.

In particular, we note that, when the event $\mathsf{GuessSC}$ occurs, the handling of $\mathcal{A}$'s $\mathsf{Execute}$ and Send queries exactly matches the behaviour and distributions of a true MFPAK system.

Result for case 4. Let $\hat{U}^* \in \mathsf{Servers}$, $\hat{U}'^* \in \mathsf{Clients}$ and let E_3 be (as before) the event that no $\mathsf{RevealRe}_{\text{MFPAK}}$ query occurs. If event E_3 occurs, then the session involving $\hat{U}'^*, \hat{U}^*$ in $\mathcal{S}_{\text{mePAK}}$ is fresh if and only if the corresponding session in $\mathcal{M}$ is fresh in the second factor. Thus, if event E_3 occurs and event $\mathsf{GuessSC}$ occurs, then, whenever $\mathcal{A}$ wins against $\mathcal{M}$, $\mathcal{A}^*$ wins against $\mathcal{S}_{\text{mePAK}}$. Therefore,

$$\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessSC}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \quad (4.19)$$

where $q'_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. Moreover,

$$\begin{aligned} &\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessSC}) \\ &= \Pr(\mathsf{Succ}^{\text{ake-f2}}_{\mathcal{M}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessSC}) \ . \end{aligned} \quad (4.20)$$

Combining these two expressions yields the following result:

Lemma 4.4 *Let $\hat{U}^* \in \mathsf{Servers}$, $\hat{U}'^* \in \mathsf{Clients}$, and suppose that no $\mathsf{RevealRe}_{\text{MFPAK}}$ query occurs (which is event E_3). Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type Send and $\mathsf{Execute}$, respectively, and at most q_{ro} random oracle queries. Then*

$$\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(\mathcal{A}) | \mathsf{E}_3, \mathsf{GuessSC}) \leq \Pr(\mathsf{Succ}^{\text{ake}}_{\text{mePAK}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ ,$$

where $q'_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, *and* $z = \min\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$, *and a similar bound exists for* $\mathsf{Adv}^{\text{c2s-f2}}_{\text{MFPAK}}$.

4.4.4.5 Overall result

By combining the cases 1 and 2, we can obtain a result for sessions that are fresh in the first factor, and by combining cases 3 and 4 we can obtain a result for sessions that are fresh in the second factor. For the ake-f1 advantage, we have

$$\begin{aligned}
&\Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})) \\
&\quad \leq \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_1, \mathsf{GuessCS}) / \Pr(\mathsf{GuessCS}) \\
&\qquad + \Pr(\mathsf{Succ}^{\text{ake-f1}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_2, \mathsf{GuessSC}) / \Pr(\mathsf{GuessSC}) \qquad (4.21) \\
&\quad \leq |\mathsf{Clients}| \cdot |\mathsf{Servers}| \cdot 8 \Pr(\mathsf{Succ}_{\text{mePAK-Z+}}(t', q_{\mathsf{se}}, q_{\mathsf{ex}}, q'_{\mathsf{ro}})) \ , \qquad (4.22)
\end{aligned}$$

where $t' \leq t + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, $q'_{\mathsf{ro}} \leq q_{\mathsf{ro}} + 1 + z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, and $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$.

For the ake-f2 advantage, we have

$$\begin{aligned}
&\Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}})) \\
&\quad \leq \Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessCS}) / \Pr(\mathsf{GuessCS}) \\
&\qquad + \Pr(\mathsf{Succ}^{\text{ake-f2}}_{\text{MFPAK}}(t, q_{\mathsf{se}}, q_{\mathsf{ex}}, q_{\mathsf{ro}}) | \mathsf{E}_3, \mathsf{GuessSC}) / \Pr(\mathsf{GuessSC}) \qquad (4.23) \\
&\quad \leq |\mathsf{Clients}| \cdot |\mathsf{Servers}| \cdot 2 \Pr(\mathsf{Succ}_{\text{mePAK}}(t'', q_{\mathsf{se}}, q_{\mathsf{ex}}, q''_{\mathsf{ro}})) \ , \qquad (4.24)
\end{aligned}$$

where $q''_{\mathsf{ro}} \leq 2q_{\mathsf{ro}} + 1 + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, $t'' \leq t + z \cdot t_{\mathsf{Gen}} + t_{\mathsf{exp}} + q_{\mathsf{ex}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}}) + q_{\mathsf{se}}(3t_{\mathsf{exp}} + t_{\mathsf{sig}})$, and $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$.

Similar bounds apply for $\mathsf{Adv}^{\text{ma-f1}}_{\text{MFPAK}}$ and $\mathsf{Adv}^{\text{ma-f2}}_{\text{MFPAK}}$.

Substituting the security statements for mePAK (Theorem 3.4) and mePAK-Z+ (Theorem 3.6) and simplifying the expressions, we obtain the following theorem showing that MFPAK is a secure multi-factor password-authenticated key exchange protocol assuming the hardness of the Computational Diffie-Hellman problem and working in the random oracle model:

Theorem 4.5 MFPAK *is a secure multi-factor password authenticated key exchange protocol.*

More precisely, let G be a finite cyclic group generated by g for which the Computational Diffie-Hellman problem is hard, and let $\mathcal{S}$ be an eu-cma secure signature scheme with security parameter κ. Let $\mathcal{A}$ be an adversary that runs in time t and makes at most q_{se} and q_{ex} queries of type Send *and* Execute*, respectively, and at most q_{ro} queries to the random oracle. Let $b_{\mathsf{re}} = 1$ if $\mathcal{A}$ makes a* RevealPWS *query to a server, and 0 otherwise. Assume passwords are uniformly distributed among the set* Passwords *and responses are uniformly distributed among the set* Responses.

Then

$$\mathsf{Adv}^{\text{ake-f1}}_{\text{MFPAK}}(\mathcal{A}) \leq \frac{16\delta((1-b_{\mathsf{re}})q_{\mathsf{se}} + b_{\mathsf{re}}q_{\mathsf{ro}})}{|\mathsf{Passwords}|} + \epsilon \tag{4.25}$$

and

$$\mathsf{Adv}^{\text{ake-f2}}_{\text{MFPAK}}(\mathcal{A}) \leq \frac{4\delta q_{\mathsf{se}}}{|\mathsf{Responses}|} + \epsilon \ , \tag{4.26}$$

where $\epsilon = 8q_{\mathsf{se}}\mathsf{Adv}^{\text{CDH}}_{G,g}(t', q'^2_{\mathsf{ro}}) + 6q_{\mathsf{se}}\mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(t', q_{\mathsf{se}}) + \frac{5(q_{\mathsf{se}}+q_{\mathsf{ex}})(q_{\mathsf{ro}}+q_{\mathsf{se}}+q_{\mathsf{ex}})}{|G|}$ *and* $\delta = |\mathsf{Clients}| \cdot |\mathsf{Servers}|$, *for* $t' = t + (z + 8(q'^2_{\mathsf{ro}} + q_{\mathsf{se}} + q_{\mathsf{ex}}))t_{\mathsf{exp}}$, $q'_{\mathsf{ro}} = 2q_{\mathsf{ro}} + 4z + 6q_{\mathsf{ex}} + 5q_{\mathsf{se}}$, *and* $z = \max\{q_{\mathsf{se}} + q_{\mathsf{ex}}, |\mathsf{Clients}| \cdot |\mathsf{Servers}|\}$. *Moreover, similar bounds exist for* $\mathsf{Adv}^{\text{ma-f1}}_{\text{MFPAK}}(\mathcal{A})$ *and* $\mathsf{Adv}^{\text{ma-f2}}_{\text{MFPAK}}(\mathcal{A})$.

4.4.5 Example instantiation

As a consequence of Theorem 4.5, we can pick a desired security level (as in Table 2.1) and, under the various computational assumptions from Section 2.2, choose a set of parameters that achieve that security level.

Suppose we wish for an adversary running in time 2^{80} to have ake-f1 and ake-f2 advantages of at most 2^{-20} against MFPAK.

To give an example instantiation, we have to pick appropriate values for the various parameters in the statement of the theorem. We choose the same values as in Section 3.3.2.1.

With this choice of parameters, we find that $z \doteq 2^{20}$, $q'_{\mathsf{ro}} \doteq 2^{43}$, $t' \doteq 2^{109}$, and $\epsilon \doteq 2^{13} \cdot \mathsf{Adv}^{\text{CDH}}_{G,g}(2^{109}, 2^{86}) + 2^{13} \cdot \mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(2^{109}, 2^{10}) + \frac{2^{63}}{|G|}$.

We need $|\mathsf{Passwords}| = |\mathsf{Responses}| = 2^{55}$, which, on the author's keyboard with 95 distinct printable characters on it, is achieved by having passwords and responses as strings of length 8, assuming passwords and responses are uniformly distributed. If $b_{\mathsf{re}} = 1$, that is, if a RevealPWS query occurs, then $|\mathsf{Passwords}| = 2^{55}$ implies an adversary must do 2^{30} random oracle queries to have a good chance of finding the password given the output of RevealPWS.

We also need $\epsilon \leq 2^{-21}$. In other words, we need $2^{13} \cdot \mathsf{Adv}^{\text{CDH}}_{G,g}(2^{109}, 2^{86}) \leq 2^{-21}$, $2^{13} \cdot \mathsf{Succ}^{\text{eu-cma}}_{\mathcal{S},\kappa}(2^{109}, 2^{10}) \leq 2^{-21}$, and $|G| \geq 2^{84}$. This means we need a group of size $q \geq 2^{458}$; in other words, we need a 458-bit elliptic curve group. Using the signature scheme ECDSA over the same size group would suffice; finally, we can instantiate the hash function with SHA-512.

MFPAK can achieve the same security level (for example, 2^{80}) as PAK or PAK-Z+ at only slightly larger key sizes (458 bits versus 430 bits), albeit with longer passwords (8 characters instead of 5 characters). It is possible that a different formal security argument could give a tighter reduction in Theorem 1 and allow for smaller password sizes; regardless, the analysis still provides strong security with realizable parameter sizes.

Chapter 5

Denial-of-Service-Resilient Authenticated Key Exchange

Contents

5.1 Introduction

Cryptographic protocols such as key exchange usually require expensive computations such as finite field exponentiations or elliptic curve scalar-point multiplications. In an adversarial environment such as the Internet, an attacker could effect a **denial of service** attack against a server by forcing the server to perform many instances of a cryptographic protocol. However, by forcing clients to perform expensive operations themselves before the server is willing to expend resources, we

can offer some resilience to these attacks. Careful attention must be paid, however, to how these denial of service countermeasures are used in order to ensure that they actually protect the server and do not open up new avenues of attack.

In this chapter, we describe a technique for formally modeling the denial of service resilience of authenticated key exchange protocols, provide a mechanism for achieving denial-of-service-resilient protocols, and analyze existing techniques. Note that in this chapter, we deal with authenticated key exchange, rather than password-authenticated key exchange as in Chapters 3 and 4. In **authenticated key exchange** the client and server each have long-term private key / public key pairs which they use for authentication, instead of short shared secrets as in password-authenticated key exchange.

Practical motivation. Secure, reliable, and fast communication is essential for commercial success on today's Internet. Key exchange provides a technique for establishing a secure connection to a web server, but it is computationally demanding and as a result may slow down service time. If a web page is slow to load, a client may switch to an alternative business, leading to a loss of customers for the initial service provider.

However, maintaining a sufficiently powerful server can be an expensive venture. The servers that host the web page have limited resources, which include the amount of traffic the server can handle, the time required to establish a connection, and the number of active connections that can run concurrently. This in effect bounds the number of connections a company's server can honour during a given period in time.

Malicious parties have recognized that exhausting an honest server's limited resources can be used to disrupt others' services for their own gain. The term *denial of service* was coined to describe attacks that aim to disrupt, destroy, or render services unavailable. A typical denial of service attack exhausts the resources of the server under attack. The server is rendered unavailable for honest clients, who then proceed to request similar services from competitors. To prevent malicious requests, a server needs to filter out bogus connection requests and honour those from legitimate clients.

Security is an important aspect of online services. Many connections between a client and a server need to be secured against third parties; financial transactions are the most common case. Key exchange is used to produce a shared secret that is used to encrypt subsequent communication. Key exchange involves computationally expensive public key algorithms, and hence may dominate server-side run time, thereby limiting the number of serviced clients. This makes key exchange an enticing target for denial of service attacks, since a malicious party can easily issue a request for key exchange. Hence, it is advantageous to try to reject as many bogus connections as possible during key exchange. In this chapter we are concerned with providing a means for deterring malicious parties from initiating denial of service attacks based on key exchange.

Practitioners and standardization bodies have recognized the importance of denial of service resilience, but researchers have been slow to respond with a formal treatment of the subject. In some sense, addressing denial of service resembles the initial approach to key exchange: rather than constructing an overall model, a list of *ad hoc* goals is selected and then it is shown that a protocol meets those goals. As a result, it is difficult to compare and analyze two different protocols with respect to denial of service resilience. Furthermore, if the lists of goals are different then it is often unclear how two protocols can be compared.

Contributions. In this chapter we propose a formal model based on the extended Canetti-Krawczyk (eCK) model for secure and authentic shared key establishment that takes into account denial of service (DoS) attacks. Whereas previous work is mainly focused on proper DoS countermeasures, we are also interested in the *integration* of DoS countermeasures with key establishment: just giving the client a puzzle and checking that it was solved is not enough to guarantee denial of service resilience for key agreement.

We analyze many of the DoS attacks presented in the literature and argue that they do not stem from a weak DoS countermeasure but from incorrect integration into the key agreement protocol. Existing formal methods have already led to the discovery of some novel DoS attacks, but not all attacks can be identified. Our model covers a wider range of DoS goals and provides a DoS framework which can be used to analyze and compare DoS countermeasures. Previous formal treatments of denial of service deal with the two party setting: an honest server and an adversary. These models do not capture goals related to hijacking of connections, but do allow for a fine analysis of the strength of a DoS countermeasure. Our model can be used in conjunction with previous analyses: we deal with the integration of DoS countermeasures and key agreement protocols, and previous work can be used to analyze the strength of the countermeasures.

Future directions. This new framework for analyzing denial of service resilience can be applied in conjunction with other goals for key agreement protocols. For example, the JFKi and JFKr protocols [ABB+04] aim to offer additional privacy features: JFKi protects the initiator's identity and JFKr protects the responder's identity. Future work in this field could involve designing denial-of-service-resilient protocols in our framework with similar privacy measures.

This model can also be applied to give denial of service resilience to other types of key agreement protocols, for example password-authenticated key agreement protocols. Adapting this technique for use in IPsec or TLS would provide denial of service resilience in important Internet protocols. Our framework is appropriate for challenge-response type puzzles for denial of service resilient (**two-pass protocols**). Protocols in which the client generates the challenge herself (**one-pass protocols**) may be of interest in asynchronous or limited connectivity environments: an open

problem is to modify our framework to describe denial of service resilience for 1-pass protocols.

Outline. The remainder of this chapter is organized as follows. In Section 5.2 we review previous developments of key exchange and denial of service resilience in key exchange protocols. We describe the goals of key exchange and denial of service resilience both informally and in a formal model in Section 5.3. We present our DoS-CMQV protocol in Section 5.4 and show that it is a secure denial-of-service-resilient key exchange protocol and describe how it can be instantiated. We discuss other constructions for denial of service resilience and relate our model to the JFKi and HIP protocols in Section 5.5.

5.2 Literature review

Key exchange. Key exchange is an important cryptographic primitive typically used for building secret channels between two parties. Designing and analysing key exchange protocols is a non-trivial task. Early approaches to defining security listed various goals and protocols were shown to satisfy each goal, but soon this gave way to a more rigorous, systematic approach.

Bellare and Rogaway [BR93a, BR95] and Blake-Wilson, Johnson, and Menezes [BWJM97] presented the first formal models for authenticated key exchange that allowed complexity-theoretic security arguments, for the symmetric and asymmetric key exchange settings, respectively. The advantage of this model-based approach is that multiple security goals can be considered at once, and model-based approaches have been widely adopted in the contemporary literature. Even if a security property was not considered when the model was designed, it is still possible to check if the model covers the new property. The formal models for password-authenticated key exchange in Section 3.2 and for multi-factor password-authenticated key exchange in Section 4.3.2 are examples of the model-based approach.

The initial security models and definitions for authenticated key exchange were extended to incorporate wider security goals. The work of Canetti and Krawczyk [CK01a], known as the **Canetti-Krawczyk model**, or **CK01 model**, is one of the most influential extensions to the original model. Their work was later augmented by Krawzcyk [Kra05a] and LaMacchia, Lauter and Mityagin [LLM07] to capture a wider range of desirable security properties, such as protection against malicious insiders and key compromise impersonation [JV96]; this is referred to as the **extended Canetti-Krawczyk model**, or **eCK model**. We discuss our denial of service extension to the eCK model in Section 5.3.2.

These security models have not yet been considered in light of denial of service resilience. In the CK01/eCK model, for example, the adversary controls *all* communication links. In this setting it is not immediately clear how denial of service can be considered alongside key establishment: the adversary may simply choose

not to deliver any of the messages addressed to a target party. Nonetheless, we believe that in this setting we can still incorporate meaningful goals related to denial of service. Moreover, in practice it is not always the case that the adversary can destroy a link at will.

A number of protocols have been developed with formal arguments for security in the extended Canetti-Krawczyk models, including NAXOS [LLM07] and CMQV [Ust08b], which has a security argument using the Gap Diffie-Hellman assumption, and, quite recently, NAXOS+ [LP08], which has a security argument using the Computational Diffie-Hellman assumption. Our work in developing a denial-of-service-resilient key exchange protocol will build upon the CMQV protocol.

Denial of service. There are two main types of denial of service attacks (see [BM03, §1.6.6] for example): **resource depletion attacks** and **connection depletion attacks**. Resource depletion attacks capture the case where a malicious party attempts to drain the computational or memory resources of a server. By contrast, connection depletion attacks aim to exhaust the number of allowed connections to the server. A protocol can aim to defend against either or both of these types of attacks.

Distributed denial of service (DDoS) attacks, in which many distributed client computers attack a single server, are of significant concern on the Internet today. Distributed attacks are very difficult to defend against. One known technique, which we use in this paper, is to allow a server to adjust its denial of service countermeasure based on the load it experiences, but this is in essence a side-effect of the approach. Puzzle auctions [WR03] are one such implementation of tunable puzzles. Our model does not aim to address distributed denial of service attacks.

Aura and Nikander [AN97] introduced the notion of **stateless connections**, in which stateful connections are transformed into stateless ones by attaching the state information to the message and using a message authentication code for integrity checking. This enables some protection against denial of service attacks by saving the server from having to store session information until later in the exchange when more assurance is possible.

Meadows [Mea99] offered the first formal framework for denial-of-service-resilient protocols, based on the causal sequencing language of fail-stop protocols of Gong and Syverson [GS95]. To avoid connection depletion, Meadows suggests that each message be authenticated with increasingly complex levels of authentication; cookies could be used for the early authentication scheme. Meadows then applies this framework to the Station-to-Station protocol [DvOW92] to identify potential DoS attacks but does not provide a denial-of-service-resilient protocol. An application of Meadow's cost-based framework to JFK revealed a potential DoS attack, and a solution to this problem was proposed using client puzzles [SGNB06].

Cookies. One of the first techniques used to defend protocols against denial of service attacks was cookies. First introduced in the Photuris protocol (published

in 1999 as [KS99] but drafts appeared earlier), **cookies** are small authentication tokens returned by a server upon initial connection by the client. In order for the client to be allowed to continue with the connection, the client must echo the cookie back to the server. The server does not store the cookie, instead using the stateless connection technique to check the authenticity of the cookie which the client includes in subsequent messages.

Krawczyk's SIGMA protocol [Kra03a] has been adapted to have denial of service resilience in the form of cookies in its implementation in the Internet Key Exchange protocol version 2 (IKEv2) [Kau05], a proposal for the successor of the Internet Key Exchange protocol (IKE) [HC98] used in IPsec [Atk95]. Cookies are also used in the Just Fast Keying protocol (JFK) proposed by Aiello *et al.* [ABB+04]. JFK allows the server to reuse its ephemeral private-public key pair across multiple sessions to reduce the server's computational overload, at the expense of increasing the potential damage should an ephemeral private key be leaked.

Cookies are a valuable first-order denial of service countermeasure and have been used extensively as described above. However, they are a weak form of denial of service resilience because they do not require an attacker to do anything other than faithfully relay a previously received cookie.

Protocols using cookies can also be susceptible to other types of attacks as shown by Mao and Paterson [MP02, §2.2], who describe a denial of service attack against IKEv2. In their attack, a malicious party who controls a popular server $\mathcal{M}$ can redirect legitimate traffic from $\mathcal{M}$ towards another target server $\mathcal{M}'$, thereby effecting a denial of service attack against $\mathcal{M}'$. The attack only costs $\mathcal{M}$ bandwidth, not computation or memory, and is resilient to standard DoS countermeasures like cookies. This attack is possible because there is no strong binding between the DoS countermeasure and the name of the server that the client wishes to connect to: we codify this notion in our security criterion DoS-2 in Section 5.3.1. Despite key agreement and denial of service being orthogonal issues, combining them is no trivial task, as demonstrated by this attack.

Puzzles. Dwork and Naor [DN92] introduced the notion of **client puzzles** to defend against denial of service attacks. A server under a denial of service attack can require clients to find the solution to a puzzle before the server allocates resources: the puzzle should be hard to solve but the solution should be easy to verify. Back [Bac97] and Juels and Brainard [JB99] suggested using a hash function so that a client must perform a large number of operations to find the solution; this is a **computationally bounded** puzzle. We build on their approach by specifying how puzzles should be integrated with key exchange. Waters *et al.* [WJHF04] describe how puzzles can be distributed across multiple servers for coordinated access.

Abadi *et al.* [ABMW03] suggested another class of puzzles called **memory-bounded puzzles**, in which computing the solution to the puzzle requires a large number of memory accesses; memory-bound puzzles have closer running times across varied hardware because memory access times vary less than processor speed

between, for example, low-power embedded devices and data centre servers. Other memory-bound puzzles have subsequently been proposed by Dwork *et al.* [DGN03] and Doshi *et al.* [DMR06].

Puzzles can be used as a suitable protection against connection depletion attacks. We demonstrate this by carefully integrating puzzles with a secure key exchange protocol. Furthermore, puzzles can be used to ensure that the client requesting the connection performs a sufficient amount of computational effort relative to the server's work. In this way the server obtains assurance that either the client is honest or the adversary's computational power is too strong to prevent denial of service attacks. Indeed, for an honest client, performing one time-consuming computation is not inconvenient, but when the work required is multiplied a thousandfold, an adversary would need to access powerful, distributed resources to mount such an attack. We must be careful, however, to not place too high a burden on the client; otherwise, just by using the protocol they may experience slowdowns comparable to what might be effected by an actual denial of service attack. What clients may consider unacceptable is based on the application and usage of the software, not the cryptography, and hence is not formally modeled.

Aura, Nikander, and Leiwo [ANL00] give a framework for using hash function preimages as a denial of service in authentication protocols, and lay out the basic principle that "the client should always commit its resources to the authentication protocol first and the server should be able to verify the client commitment before allocating its own resources". We apply this principle to our development of the model for denial-of-service-resilient key agreement. However, the technique of [ANL00] is not sufficient to defend against the attack of Mao and Paterson [MP02]; our approach does.

5.3 Security and denial of service resilience

In this section, we present a model for the security of authenticated key exchange and describe how denial of service can be integrated with this model.

While the goals of denial of service resilience and secure key agreement are, as others such as Krawczyk [Kra03a, §2.3] have noted, orthogonal issues, it is useful to be able to discuss them in a common framework. We also must be careful to integrate the two issues sufficiently well to avoid the types of attacks proposed by Mao and Paterson [MP02].

5.3.1 Informal security and denial of service criteria

Session key security and authentication. The security model presented herein is based on the extended Canetti-Krawczyk model, which has some similarities to the Bellare-Pointcheval-Rogaway model for password-authenticated key exchange presented in Section 3.2, but with one essential difference: in the eCK model,

participants use long term private key / public key pairs rather than passwords. Consequently, we can consider security when the adversary is allowed to learn one of either the long-term private key or the ephemeral private key, whereas we could not allow the adversary to learn the ephemeral private key in password-authenticated key exchange.

The criteria for session key security are conceptually the same as in Section 4.3.1: a session key is secure if, for a sufficiently uncompromised ("fresh") session, an adversary is unable to distinguish the session key from a random string except with negligible probability. The model is formally described in Section 5.3.2 and the definitions of security are given in Section 5.3.2.1.

In this setting, we consider the case where parties have long-term private key / public key pairs which are used in the key exchange. As a result, security of the session key leads to implicit mutual authentication: if session key establishment is successful, then only parties named in the session know the key. Explicit mutual authentication can be achieved by adding key confirmation (*cf.* [MvOV01, §12.1]).

Denial of service resilience. We are concerned about the situation in which a malicious party on the network can cause a server to perform many expensive operations (and key agreement is one such expensive operation) for no good reason, eventually consuming all of the server's available resources.

But since the server is willing to place itself on the network for the use of all users, how can the server know if it is doing work for a good reason or not? Distinguishing legitimate requests from malicious requests is an essential element of denial of service resilience.

While one can never be certain about the good intentions of another party on the network, a common approach is for a server to be more likely to believe that a client is making a legitimate request if the client is willing to commit some expensive resources — computation, memory, etc. — to the connection request in order to show its good faith. However, if a client does do something expensive to prove its good faith, then a good protocol should protect the client from being exploited by a malicious party aiming to steal the client's work.

These ideas lead us to the following five informal criteria for a denial-of-service-resilient protocol:

DoS-1. An uncompromised honest server does not perform any expensive operations with a client unless it is convinced the client is trying to make a legitimate connection.

DoS-2. Moreover, a server $\hat{B}$ does not perform any expensive operations unless it is convinced that the client wants to talk to $\hat{B}$ and not another server $\hat{M}$.

DoS-3. A client $\hat{A}$ who commits significant resources to prove its legitimate intentions cannot have her work stolen: the work that $\hat{A}$ does to convince $\hat{B}$ that it wants to communicate legitimately with $\hat{B}$ cannot convince anyone of anything else.

DoS-4. A malicious party must use a very large amount of resources if it wishes to prepare sufficiently many connection requests and flood a server with many valid connection requests.

DoS-5. A server can adjust the amount of work a client has to do in times of higher or lower load.

In Section 5.3.2.3, we give a formal definition of denial of service resilience and describe in Section 5.3.3 how our model achieves each of the goals DoS-1 through DoS-4 above; goal DoS-5 is a property of the specific countermeasure in the protocol and not of the formal model.

In the first two goals above, we aim to protect the server from performing unnecessary expensive operations. While what qualifies as an expensive operation can vary depending on the setting, we identify three main classes of expensive operations for the purposes of denial of service and some examples in each class, most of which are resource depletion attacks:

- *Memory denial of service*: forcing the server to perform slow, expensive memory accesses or commit to using a large block of memory. Expensive operations include storing long-term data (on disk), loading long-term data (from disk), and use of large amounts of short-term or long-term memory space.
- *Computational denial of service*: forcing the server to perform operations that require significant computational time. Expensive operations include exponentiation (e.g., modulo a large prime or in an elliptic curve group) and a large number of simpler operations (e.g., hash function calls, MAC evaluations, etc.) that in total rival the cost of an exponentiation.
- *Transmission denial of service*: forcing the server to expend its resources available for connections. Expensive operations include committing one of a limited number of connection ports (also called connection depletion attacks), transmitting large amounts of data, and receiving large amounts of data.

In various environments, there can be different notions of "expensive". For example, in mobile environments, transmitting and receiving take a lot of time and power, so a large number of message flows may lead to transmission denial of service.

To achieve denial of service resilience, we require that a client solve a puzzle. Our key idea is that we also tightly bind the puzzles with the identities of the parties involved to avoid attacks in which work can be stolen or redirected. This is another example of the wise principles for protocol design laid out by Abadi and Needham:

> If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message. [AN96, p. 8]

5.3.2 Formal model

We present a model that extends the eCK model for authenticated key exchange. On one hand we extend the power of the adversary relative to extended Canetti-Krawczyk model by allowing the adversary to obtain session-specific information related to the test session. On the other hand the adversary is no longer able to select session identifiers on behalf of the parties. In other words parties agree on a common session identifier string. In the original model description, Canetti and Krawczyk argue that the string may in fact be selected by communicating parties in advance. A more common approach is that the session should be identified via concatenations of exchanged messages similar to the Bellare-Rogaway model [BR93a].

Participants. Key exchange protocols take place among parties $\hat{A}, \hat{B}, \ldots \in \mathsf{Parties}$. Each party is a probabilistic polynomial-time Turing machine.

Keys. Each participant has a static key pair that was certified by a certification authority; the CA should also verify possession of the private key. We assume that the certification authority verified that each static public key belongs to a suitable cryptographically strong group G. Each party can also possess static information that is not certified which may be either private or public.

Execution of the protocol. The protocol is a probabilistic algorithm on strings which specifies how each party responds to messages. Each party may have multiple instances of the protocol running during execution. Parties are activated via incoming messages. Upon activation, parties process messages and either submit outgoing responses or indicate if the processing of the incoming message resulted in failure or success.

An execution of the protocol is called a **session**.

Within a party the execution of the subroutines between the session request and either accepting or rejecting the request to initiate a session is called **pre-session**. Note that it is possible that a single request to initiate a session may result in many sessions executed within a party: the model does not prevent a protocol from accepting multiple responses to the same request.

Pre-session. A party may receive an incoming *request* to initiate a session via a message of the form (i) $(\hat{A}, \hat{B})$ or (ii) $(\hat{A}, \hat{B}, \text{“hello”})$.

In the former case, (i), $\hat{A}$ is called *client* or *initiator*. Party $\hat{A}$ creates a separate *session request* that contains the identity $\hat{B}$ and the initiation request “hello”. The outgoing message designated for $\hat{B}$ is $(\hat{B}, \hat{A}, \text{“hello”})$.

In the latter case, (ii), $\hat{A}$ is called *server* or *responder*. Party $\hat{A}$ selects a fresh (unique within $\hat{A}$) challenge ch and sends outgoing message $(\hat{B}, \hat{A}, \text{“hello”}, \mathsf{ch})$.

Our motivation for introducing pre-sessions is that the denial of service countermeasure will be modeled in the pre-session and the expensive resources committed to be a server will be modeled in the session. A session can only be reached after a successful pre-session: expensive resources will only be committed once the denial of service countermeasure is accepted.

Session creation. A party $\hat{A}$ can be activated to *create* a session with a message of the following form (i) $(\hat{A}, \hat{B}, \text{“hello”}, \mathsf{ch})$ or (ii) $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$.

If the activation is of type (i), then $\hat{A}$, who is the initiator, prepares a suitable re that passes all conditions required by the protocol and creates a session that is labeled *active*. The string re is unique within $\hat{A}$ and the outgoing message is $(\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re})$.

If the activation is of type (ii), then $\hat{A}$, who is the responder in this case, verifies that the message $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ satisfies the protocol requirements; if so, a new active session is created; otherwise the message is ignored. If a new session is created by the responder $\hat{A}$, then the outgoing message is (Ψ, M), where M is a message prepared by $\hat{A}$ in accordance with the protocol and Ψ is a string used to identify sessions within $\hat{A}$ and $\hat{B}$.

The string Ψ is called a **session string identifier** and is derived from $(\mathsf{ch}, \mathsf{re})$. The uniqueness conditions imposed on ch and re allow for the derivation of a string unique within both $\hat{A}$ and $\hat{B}$.

Upon creating a session, $\hat{A}$ also creates a separate *session state* that contains session-specific information. Session-specific information can be either private or public. The private information is required to compute a secret shared session key. The public information is $(\hat{A}, \hat{B}, \Psi, \mathsf{role}, \mathsf{otherinfo})$, where $\hat{B}$ is the purported *session peer*, role is either “initiator” or “responder”, and $\mathsf{otherinfo}$ is any other public information required by the protocol. The session is globally identified by the **session identifier** $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$, where Ψ identifies the session within $\hat{A}$ and is derived from $(\mathsf{ch}, \mathsf{re})$.

Two sessions $\mathsf{sid}_1 = [\hat{A}, \hat{B}, \Psi]$ and $\mathsf{sid}_2 = [\hat{C}, \hat{D}, \Psi']$ are said to be *matching* if $\Psi = \Psi'$, $\hat{A} = \hat{D}$ and $\hat{C} = \hat{B}$. In other words matching sessions have the same session identifier string and the same communicating parties.

The protocol may require parties to perform certain validation steps during the protocol execution. If any validation step fails, the party erases all session specific private information and aborts the session with output $(\hat{A}, \hat{B}, \Psi, \perp)$, where $\perp$ indicates the session is *aborted*. At any stage a session is in exactly one of the following states: active, completed or aborted.

Off-line requests. In some cases it may be desirable to allow the client to select the string ch. In that case the pre-session stage is combined with session creation. In particular, parties are activated to create sessions with messages of the form

$(\hat{A}, \hat{B})$ when $\hat{A}$ is the client and $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ when $\hat{A}$ is the server. When $\hat{A}$ is the server, $\hat{A}$ may have to, in addition to the validation steps, send some additional information to allow $\hat{B}$ to complete its derivation of the session string identifier Ψ.

Session update. A party $\hat{A}$ can be activated to update a session via a message of the form $(\Psi; \mathrm{M})$. Messages are delivered via the Send query described below. Upon receipt of this message $\hat{A}$ verifies that Ψ identifies a session within $\hat{A}$. If no such session exists the activation is ignored and appropriate error message output. Otherwise $\hat{A}$ updates the session state corresponding to sid as required by the protocol. If the protocol requires a response by $\hat{A}$, then $\hat{A}$ prepares the required response M'; and submits an outgoing message $(\Psi; \mathrm{M}')$ designated for the peer identified in the session state's public part. If the protocol specifies that no further messages will be received, then the session *completes* by producing local output $(\hat{A}, \hat{B}, \Psi, K)$, where K is private information.

Powers of the adversary. The adversary $\mathcal{A}$ is modeled as a probabilistic Turing machine and controls *all* communications. Parties submit outgoing messages to $\mathcal{A}$, who makes decisions about their delivery. The adversary communicates with the parties via **queries** and receives responses. As in Section 3.2, there are three classes of queries: (1) queries that model the transmission of messages across communication links, (2) queries that model the adversary learning certain information by compromising a party in some way, and (3) queries that have been added to allow the formulation of a game for the adversary to win.

The first query, Send, models the transmission of messages across communication links.

Send(M): Sends message M. Here, the message M also includes information identifying the intended recipient and session, in contrast to the Send query of Section 3.2. The recipient performs the appropriate portion of the protocol based on its current state and the message M, updates its state as appropriate, and returns any resulting messages.

The next five queries model the adversary's ability to compromise certain pieces of information held by the parties.

RevealStaticKey($\hat{A}$): Returns $\hat{A}$'s static private key.

RevealEphemeralKey(sid): Returns the ephemeral private key of the owner in session sid. We assume that $\mathcal{A}$ issues this query only to sessions that hold an ephemeral private key.

RevealSessionKey(sid): If sid has completed, then returns the session key in sid. We assume that $\mathcal{A}$ issues this query only to sessions that have completed.

EstablishParty($\hat{M}, M$): This query allows $\mathcal{A}$ to register an identifier $\hat{M}$ and a static public key M on behalf of a party. The adversary totally controls that party, thus permitting the modeling of attacks by malicious insiders. Parties that were

established by $\mathcal{A}$ using EstablishParty are called *adversary-controlled*. If a party is not adversary-controlled it is said to be **honest**.

DoSExpose($\hat{A}$): For an honest party $\hat{A}$, returns the non-certified private information that belongs to $\hat{A}$, excluding the session-related ephemeral private keys. Parties against which this query was issued are called **DoS-exposed**, otherwise they are called **DoS-unexposed**.

Additionally, the use of a hash function may be modeled as an oracle query when working in the random oracle model.

The role of the new DoSExpose query in our model is to allow us to identify the parties which ought to still be resilient to denial of service attacks, namely those parties which are DoS-unexposed. For example, adversary-controlled parties are not relevant to the DoS portion of the protocol. Moreover, a separate DoSExpose query allows us to separate denial of service resilience from key-exchange security: compromise of key-exchange secrets can be an orthogonal issue to compromise of denial of service.

5.3.2.1 Session key security

To capture the indistinguishability requirement for security of the session key, $\mathcal{A}$ is allowed to make a special query Test(sid) to a fresh session sid. In response, $\mathcal{A}$ is given with equal probability either the session key held by sid or a random key. $\mathcal{A}$ meets its goal if it guesses correctly whether the key is random or not. Note that $\mathcal{A}$ can continue interacting with the parties after issuing the Test query, but must ensure that the test session remains fresh throughout $\mathcal{A}$'s experiment.

Test(sid): If session sid has completed, then causes the following to happen: choose $b \in_R \{0,1\}$; if $b = 1$, then return the session key held by sid, otherwise return a random string chosen from the same distribution as the session key. This query may only be asked once.

Freshness. Let sid be the identifier of a completed session, owned by an honest party $\hat{A}$ with peer $\hat{B}$, who is also honest. Let sid* be the identifier of the matching session of sid, if it exists. Then sid is **fresh** if none of the following conditions hold:

1. $\mathcal{A}$ issued RevealSessionKey(sid) or RevealSessionKey(sid*) (if sid* exists).
2. sid* exists and $\mathcal{A}$ issued one of the following: either
   (a) both RevealStaticKey($\hat{A}$) and RevealEphemeralKey(sid), or
   (b) both RevealStaticKey($\hat{B}$) and RevealEphemeralKey(sid*).
3. sid* does not exist and $\mathcal{A}$ issued one of the following: either
   (a) both RevealStaticKey($\hat{A}$) and RevealEphemeralKey(sid), or
   (b) RevealStaticKey($\hat{B}$).

Adversary's goals. The goal of an adversary is to guess the bit b used in the Test query of a fresh session. As in Section 3.2.2, we let $\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A})$ be the event that the adversary $\mathcal{A}$ makes a single Test query to some fresh session and eventually outputs a bit b', where $b' = b$ and b is the randomly selected bit in the Test query. Similarly, we define $\mathsf{Adv}^{\mathrm{ake}}(\mathcal{A}) = 2\Pr\left(\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A})\right) - 1$.

Definition 5.1 (Secure key exchange protocol) *A key exchange protocol P is* **secure** *if the following conditions hold:*

1. *If two honest parties complete matching sessions then, except with negligible probability, they both compute the same session key.*
2. *No polynomially bounded adversary $\mathcal{A}$ can distinguish the session key of a fresh session from a randomly chosen session key with probability greater than $\frac{1}{2}$ plus a negligible fraction (in some security parameter):* $\mathsf{Adv}^{\mathrm{ake}}_P(\mathcal{A}) \leq \frac{1}{2} + \epsilon$.

5.3.2.2 Authentication

A secure key exchange protocol as in Definition 5.1 also provides implicit mutual authentication. Condition 2 of the definition implies that, when two honest parties have completed matching fresh sessions, no adversary can distinguish the session key from a random string: in other words, no one except the two parties know anything about the session key. As a result, only they can obtain meaningful information from encryption or other operations based on the session key, and when they communicate successfully this will implicitly authenticate each party to the other.

Alternatively, the two parties can have explicit mutual authentication by adding a short *key confirmation* protocol following the main protocol. One standard technique is to exchange hashes of the session identifier and session key under new independent hash functions; in the random oracle model, this provides no information about session key. We do this in our denial-of-service-resilient key exchange protocol DoS-CMQV (Figure 5.2, lines 23 and 29) in the messages involving M_1 and M_2.

5.3.2.3 Denial of service resilience

The main idea of our denial of service resilience definition is that, in the execution of the protocol, there is a test that the server performs on some of the messages received to determine if the client has done sufficient work to merit the server performing expensive operations. The work the client has to do is modeled by a "puzzling relation" and is used to define the denial-of-service-resilience of a protocol.

Definition 5.2 (Puzzling relation) *Let* Challenges *and* Responses *be sets. A relation* $\mathcal{R} \subseteq \mathsf{Parties} \times \mathsf{Parties} \times \mathsf{Challenges} \times \mathsf{Responses}$ *is a* **puzzling relation** *if*

1. *deciding if* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ *is "easy", and*

2. *given* $\hat{A}, \hat{B}, \mathsf{ch}$*, and an oracle* U *that, on input* $(\hat{A}', \hat{B}', \mathsf{ch}')$*, returns* re' *such that* $(\hat{A}', \hat{B}', \mathsf{ch}', \mathsf{re}') \in \mathcal{R}$*, it is "hard" to produce* re *such that* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ *and* re *was not a response generated by the oracle* U *upon input* $(\hat{A}, \hat{B}, \mathsf{ch})$*.*

The notion of "expensive operation", "easy", and "hard" will depend on the application context; some examples of expensive operations are given in Section 5.3.1. Although we have left these notions vague, they can be formalized, for example as a proof of work [JJ99]. We omit this formalization as the focus of our work is on the integration of denial of service resilience techniques into key exchange protocols, not the construction of suitable puzzles.

Definition 5.3 (Acceptable pre-session) *A pre-session* $[\hat{A}, \hat{B}, \mathsf{ch}]$ *is an* **acceptable pre-session** *for* $\hat{B}$ *if* $\hat{B}$ *generated* ch*.*

Definition 5.4 (Denial-of-service-resilient protocol) *Let* $\mathcal{R}$ *be a puzzling relation. A protocol* P *is* **denial-of-service-resilient** *if the following hold for every DoS-unexposed server* $\hat{B}$*:*

1. $\hat{B}$ *only performs expensive operations (a) in a session, or (b) for some (low frequency) periodic update of its non-certified information* ρ*, and*

2. $\hat{B}$ *only establishes a session* $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$ *if the pre-session* $[\hat{A}, \hat{B}, \mathsf{ch}]$ *was an acceptable pre-session for* $\hat{B}$ *and* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$*.*

Note that we have explicitly avoided merging Definitions 5.3 and 5.4 (as Definition 5.3 could be part of Condition 2 of Definition 5.4) because we acknowledge that there might be protocols that require a different notion of acceptable pre-session. In particular, it appears to suffice that $\hat{B}$ has an assurance that ch was generated independently at random before re. A one-pass denial of service resilient protocol might take this form when both ch and re are generated by the initiator, and is the subject of future work.

5.3.2.4 Comparing security models

Since we have presented two major security models, the BPR00 model for password-authenticated key exchange in Section 3.2, and the eCK model for authenticated key exchange in this section, it is instructive to compare the two models, which we do in Figure 5.1.

One notable difference between the two models is the RevealEphemeralKey query in the eCK model. This query does not appear in the BPR00 model for password-authenticated key exchange because of the need to protect password-authenticated key exchange queries against dictionary attacks. Intuitively, if we allowed the

Characteristic	**BPR00**	**eCK**
Authentication secret	short password	private key / public key pair
Transmission queries	Send Execute	Send
Compromise queries	RevealSessionKey RevealPWC RevealPWS	RevealSessionKey RevealStaticKey RevealEphemeralKey EstablishParty
Our additional compromise queries	RevealRe (Section 4.3.2)	DoSExpose (Section 5.3.2)
Game queries	Test	Test
Authentication	Explicit	Implicit

Figure 5.1: Comparison of BPR00 and eCK security models

ephemeral key to be revealed, then in some sense the only value left unknown to the adversary in a particular session would be the low-entropy password. The ephemeral key shields this low-entropy value, preventing dictionary attacks. In the eCK model, the long-term private key is cryptographically strong and thus sufficiently large to render brute force attack computationally infeasible.

5.3.3 Model implications

In this section we explain how our formal model of denial of service resilience in Definition 5.4 satisfies the informal goals stated in Section 5.3.1. We omit goal DoS-5 because it is a property of a particular countermeasure in a protocol, not of the model itself.

DoS-1. *An uncompromised honest server does not perform any expensive operations with a client unless it is convinced the client is trying to make a legitimate connection.*

Argument. By condition 1 of Definition 5.4, a server does not perform any expensive operation with a client until it has established a session, and by condition 2 it will not establish a session until it received a response to its challenge that satisfies the puzzling relation. In order to satisfy the puzzling relation, the client must do a significant amount of work because of condition 2 of Definition 5.2; by doing this work, the client convinces the server that it is trying to make a legitimate connection. Moreover, since sessions are unique within a party, replay attacks of legitimate connection requests are prevented. □

DoS-2. *Moreover, a server $\hat{B}$ does not perform any expensive operations unless it is convinced that the client wants to talk to $\hat{B}$ and not another server $\hat{M}$.*

Argument. Condition 2 of Definition 5.2 allows us to meet this criterion: even if an adversary obtains any tuple $(\hat{A}, \hat{M}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$, the tuple $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ is unlikely (except with negligible probability) to be in $\mathcal{R}$ and moreover it remains hard to produce a response re' such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}') \in \mathcal{R}$. Note that, since it is hard to create such a tuple given oracle access to $\mathcal{R}$, then it is still hard to construct any tuple in $\mathcal{R}$ without oracle access. □

Our approach avoids the attack of Mao and Paterson [MP02] against IKEv2 in which an attacker can redirect traffic from his server towards other servers and can cause the receiving server to deplete its connection resources at low expense to the attacker. That attack is possible because there is no cryptographic binding between the denial of service countermeasure and the identities of the parties involved. By including the names of the client and server in the puzzling relation, a server $\hat{B}$ can be assured that whoever solved the puzzle intended to communicate with $\hat{B}$.

In order to achieve DoS-2, it appears we do not strictly need to include the client's identity $\hat{A}$ in the puzzling relation, which could help us achieve greater client anonymity. However, an alternative approach would then be needed to prevent this party's work from being stolen as in DoS-3.

DoS-3. *A client $\hat{A}$ who commits significant resources to prove its legitimate intentions cannot have her work stolen: the work that $\hat{A}$ does to convince $\hat{B}$ that it wants to communicate legitimately with $\hat{B}$ cannot convince anyone of anything else.*

Argument. Suppose $\hat{B}$ is a DoS-unexposed server and suppose an honest client $\hat{A}$ starts a pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$, and then finds a value re such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$. The client wishes that the response value should not be useful to anyone else trying to establish a session; in other words, no one should be able to steal $\hat{A}$'s work and use it in another pre-session.

Suppose $[\hat{A}', \hat{B}', \mathsf{ch}']$ is another pre-session, $[\hat{A}', \hat{B}', \mathsf{ch}'] \neq [\hat{A}, \hat{B}, \mathsf{ch}]$. Given these values, it is hard to produce re' such that $(\hat{A}', \hat{B}', \mathsf{ch}', \mathsf{re}') \in \mathcal{R}$, even with help from another pre-session such as $[\hat{A}, \hat{B}, \mathsf{ch}]$, because $\mathcal{R}$ is a puzzling relation. The help given from the other pre-session can be modeled as one response of the oracle U in Definition 5.2 for another pre-session, and this is of no help in a puzzling relation.

Thus, an honest client's work in solving a puzzle cannot be of use to anyone else responding to a different challenge, or with a different server, or with a different user name. □

We note that, if the adversary $\hat{M}$ simply relays $\hat{A}$'s entire response and then participates in $\hat{A}$'s place, the server will proceed with key exchange but this session will ultimately fail, since it is a secure authenticated key exchange protocol according to Definition 5.1, and thus $\hat{M}$ cannot complete a session masquerading as $\hat{A}$.

DoS-4. *A malicious party must use a very significant amount of resources if it wishes to prepare sufficiently many connection requests and "flood" a server with many valid connection requests.*

As noted in DoS-1, a server will only perform expensive operations if it has been convinced by a client that the client is trying to make a legitimate connection, and this corresponds to the client having solved an instance of the puzzling relation, which is "hard" for the client and requires a significant amount of resources. An attacker who wishes to exhaust a server's connection resources must have substantially greater resources available to mount the attack.

Suppose that for an attacker to start a session requires t steps of computation (e.g., t may be the number of cycles it takes to solve a computationally bound puzzling relation), and a server has enough computational resources to support n connections per second. Then, roughly speaking, an attacker's computers must be able to perform tn steps of computation per second to sustainably render the server unavailable through a denial of service attack. Mounting this form of attack may then require the adversary to use a distributed denial of service attack, which is both more difficult to mount and more difficult to defend against. Our approach, while not completely defending against such powerful distributed attacks, at least allows the amount of denial of service resilience to be tuned in the event of heavy traffic.

Consider in particular the case of *replay attacks*, in which an attacker replays many messages to a server. Suppose in particular that an attacker replays a response value re for a pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$. This set of values leads to the server session $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$, which already exists in the server. Since sessions must be unique in the Canetti-Krawczyk model, the server will not start a new session and hence commit no new resources as a result of this replay. This requires the server to store a table of session identifiers, but this does not result in a denial of service attack in theory since it is committing memory resources for the entries in the table only once the puzzling relation has been passed. To limit the size of the table, the server could roll over the non-certified information ρ periodically. When receiving a previous challenge and response, an entirely acceptable action would be for the server to respond to the replay with the same response it gave previously. Now, if the attacker were to compute a different response value re' for the pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$, then the server would commit new resources to the new session, but this is acceptable since the attacker solved the puzzling relation, just as a legitimate client must.

Since in the Canetti-Krawczyk model we allow the adversary to control the delivery of messages, an adversary may choose not to deliver the final message from the client to the server and leave the server with an incomplete session (similar to the half-open connections of TCP SYN flood attacks [Edd07]). This however is not a denial of service attack on key agreement: the puzzling relation in the pre-session was passed before the server committed its resources in the session. In general, this type of attack cannot be easily incorporated into a Canetti-Krawczyk-type model

where the adversary has the power to initiate sessions.

5.4 DoS-CMQV

In this section, we describe DoS-CMQV, our denial-of-service-resilient key exchange protocol.

Our DoS-CMQV protocol, given in Figure 5.2, is an adaptation of the CMQV [Ust08b] secure authenticated key exchange protocol. We use the problem of finding preimages for a random hash function as the expensive puzzle at the heart of the puzzling relation that a client needs to solve.

5.4.1 Design ideas

We designed DoS-CMQV by taking the CMQV protocol [Ust08b] and modifying the pre-session to include a denial of service countermeasure based on hash functions.

Denial of service countermeasure. A client wishing to prove its legitimate intention must solve a computational puzzle that involves finding a preimage in a hash function. Differing from previous approaches, we include the client identity, server identity, and client ephemeral public key in the hash function input. This ties the puzzle solution to a particular session so it cannot be used for other purposes.

Session key security. For key agreement, we use the CMQV protocol without significant modification. It combines the static and ephemeral keys of each party at various stages through the protocol to achieve strong security and authentication. By hashing these values together in the early stages of the protocol, the protocol attains resilience against ephemeral private key leakage without using non-standard assumptions such as the Knowledge of Exponent Assumption [BP04] used in the argument for HMQV [Kra05b]. By combining these in the computation of the session key, implicit authentication is achieved since only a party who knows both the static and ephemeral private keys can construct the correct session key. The security argument rests on the Gap Diffie-Hellman assumption and the random oracle model.

5.4.2 Protocol specification

The DoS-CMQV protocol is given in Figure 5.2 below.

Notation. DoS-CMQV operates over a finite cyclic group G for which the Gap Diffie-Hellman (GDH) assumption holds. Let λ be a security parameter. The notation $L[i]$ denotes the ith component in the tuple L: for example, if $\mathsf{ch} = (i, j)$, then $\mathsf{ch}[1] = i$ and $\mathsf{ch}[2] = j$. H_0 and H_1 are random hash functions [BR04] that return bit strings; all other hash functions are random hash functions that return integers between 1 and $q - 1$, where q is the order of the group generated by g. We use the notation $x_{[1...w]}$ to denote the first w bits of x.

Note that according to our definition the protocol starts by establishing the session. The steps before the establishment of a session are the pre-session.

For our protocol as described in Figure 5.2, the effects of the information compromise queries of the adversary are as follows:

- $\mathsf{RevealStaticKey}(\hat{A})$: returns client $\hat{A}$'s long-term private key a; this is $\hat{A}$'s certified private information.
- $\mathsf{RevealStaticKey}(\hat{B})$: returns server $\hat{B}$'s long-term private key b; this is $\hat{B}$'s certified private information.
- $\mathsf{RevealEphemeralKey}([\hat{A}, \hat{B}, \mathsf{ch}, *])$: returns client $\hat{A}$'s ephemeral private key $\tilde{x}$.
- $\mathsf{RevealEphemeralKey}([\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}])$: returns server $\hat{B}$'s ephemeral private key $\tilde{y}$.
- $\mathsf{RevealSessionKey}(\mathsf{sid})$: returns the session key K.
- $\mathsf{DoSExpose}(\hat{B})$: reveals server $\hat{B}$'s DoS private value ρ; this is $\hat{B}$'s non-certified private information.

The session identifier held by the client $\hat{A}$ is $[\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}]$, and the session identifier held by the server $\hat{B}$ is $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$.

Tuning the puzzling relation. The puzzles used in our DoS-CMQV protocol can be tuned by the server based on the load it experiences. The client must find a preimage for the hash function H_1; for concreteness, we have specified that this should have 20 bits of output, but the length could be a parameter w set by the server depending on its current load. The server would need to include w in the computation of j on line 2, return w as part of ch on line 3, and include w in the check on line 9.

Efficiency. In terms of the number of group exponentiations performed, DoS-CMQV attains the same efficiency as CMQV, described in Section 3.3 of [Ust08b]: 3 exponentiations in the elliptic curve case, which can be reduced to 2.25 exponentiations by improvements such as Shamir's trick for simultaneous multiple (batched) exponentiation [MvOV01, Algorithm 14.88].

DoS-CMQV			
	Client $\hat{A}$		Server $\hat{B}$
0.	$g, a, A = g^a, B$		$g, b, B = g^b, A, \rho \in_R \{0,1\}^\lambda$
1.		$\xrightarrow{\text{"hello"}, \hat{A}, \hat{B}}$	$i \in_R \{0,1\}^\lambda$
2.			$j = H_0(\rho, \hat{A}, \hat{B}, i)$
3.	$\tilde{x} \in_R \{0,1\}^\lambda$	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} = (i, j)$
4.	$x = H_2(\tilde{x}, a)$		
5.	$X = g^x$		
6.	find ℓ s.t. $H_1(\hat{A}, \hat{B}, \mathsf{ch}, X, \ell)_{[1\ldots20]} = 0\ldots0$		
7.	$\mathsf{re} = (X, \ell)$		
8.	establish session $[\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}]$		
9.	$\Psi = (\mathsf{ch}, \mathsf{re})$	$\xrightarrow{\hat{A}, \mathsf{ch}, \mathsf{re}}$	verify $\mathsf{ch}[2] = H_0(\rho, \hat{A}, \hat{B}, \mathsf{ch}[1])$
10.			verify $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots20]} = 0\ldots0$
11.			establish unique session $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$
12.			store $\hat{A}, \Psi = (\mathsf{ch}, \mathsf{re})$
13.			$X = \mathsf{re}[1]$
14.			verify $X \in \mathcal{G}$
15.			$\tilde{y} \in_R \{0,1\}^\lambda$
16.			$y = H_2(\tilde{y}, b)$
17.			store $Y = g^y$
18.			$d = H_3(X, \hat{A}, \hat{B})$
19.			$e = H_3(Y, \hat{A}, \hat{B})$
20.			$\sigma = (XA^d)^{y+eb}$
21.			store $M_1 = H_4$("server finished", $\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma$)
22.			store $M_2 = H_4$("client finished", $\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma$)
23.	verify $Y \in \mathcal{G}$	$\xleftarrow{\Psi, Y, M_1}$	store $K = H_5(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
24.	$d = H_3(X, \hat{A}, \hat{B})$		
25.	$e = H_3(Y, \hat{A}, \hat{B})$		
26.	$\sigma = (YB^e)^{x+da}$		
27.	verify M_1		
28.	$M_2 = H_5$("client finished", $\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma$)		
29.	$K = H_5(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$	$\xrightarrow{\Psi, M_2}$	verify M_2

Figure 5.2: DoS-CMQV: A denial-of-service-resilient adaptation of the CMQV protocol.

As for the efficiency of the denial of service countermeasure we have added, it follows from our denial of service resilience analysis in Section 5.4.4 that the server is not significantly burdened by this addition. The burden on the client is substantial, but this is by design: the reasoning behind the denial of service countermeasure is that the client must commit significant computational resources if the connection request is to be perceived as legitimate.

5.4.3 Security analysis of DoS-CMQV

In this section, we show that DoS-CMQV is a secure authenticated key exchange protocol. The argument follows from the argument presented for CMQV in [Ust08b]. We relate the difficulty of distinguishing the session key of a fresh session to the difficulty of solving the Computational Diffie-Hellman problem in a group where the Gap Diffie-Hellman assumption holds.

Theorem 5.5 *If $H_0, \ldots, H_5$ are random oracles, and G is a group where the Gap Diffie-Hellman assumption holds, then the DoS-CMQV protocol is a secure key exchange protocol.*

Argument. We will argue that the security of DoS-CMQV follows from the security of CMQV. Verifying condition 1 of Definition 5.1 is straightforward. It remains to verify condition 2.

We note that in our model of Section 5.3.2, parties possess additional (uncertified) private information ρ, which the adversary can obtain via $\mathsf{DoSExpose}$ query. We construct a polynomially bounded algorithm $\mathcal{S}$ that transforms an attack on DoS-CMQV into a solution to a Gap Diffie-Hellman instance. It does this by establishing and simulating parties as in the CMQV analysis in Section 4 of [Ust08b]. The only difference is that when parties are established the solver selects randomly the value ρ for each party. The $\mathsf{DoSExpose}$ queries are answered faithfully and they do not affect the freshness of the session. Since this new query is not relevant to the security analysis, $\mathcal{S}$ can transform the DoS-CMQV adversary to a GDH solver with similar success and running time as a CMQV adversary. Since CMQV is a secure key agreement protocol in the extended Canetti-Krawczyk model, so too is DoS-CMQV a secure key agreement protocol under Definition 5.1. □

Security of CMQV. We now give a brief overview of the security argument for CMQV, based on the presentation in [SU08]. For the detailed argument, we refer the reader to [Ust08b, §4].

Argument. Let λ be the security parameter and suppose there exists a polynomially bounded (in λ) CMQV adversary $\mathcal{A}$, that operates in an environment with $n(\lambda)$ honest parties and activates each party to create at most $s(\lambda)$ sessions, for polynomials $p(\lambda), s(\lambda)$. Suppose further that the adversary makes h_i queries to oracle H_i for $i = 1, \ldots, 5$.

Let $\mathsf{sid}_t = (\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ be the session identifier of the Test session selected by the adversary, where the peers of sid_t are the honest parties $\hat{A}$ and $\hat{B}$. Without loss of generality, let sid_t's owner, $\hat{A}$, also be the initiator. Denote the ephemeral public keys of sid_t with X and Y. Let H denote the event that the adversary queries H_5 with $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$, where $\sigma = \mathsf{DH}(XA^d, YB^e)$ and let $\overline{H}$ be the complement of H.

We construct a polynomially bounded algorithm $\mathcal{S}$ that uses $\mathcal{A}$ to solve a GDH instance.

Since the key derivation function includes the session identifier and the exchanged ephemeral public keys (since re contains the client's ephemeral public key X), the adversary cannot obtain any information about the Test session key from the session keys of non-matching sessions. Then

$$\Pr(\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A})) = \Pr(\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A}) \wedge H) + \Pr(\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A}) \wedge \overline{H}) \tag{5.1}$$

$$\leq \Pr(\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A}) \wedge H) + \frac{1}{2} \tag{5.2}$$

since H_5 is a random oracle and no advantage can be gained if H_5 is not queried (which corresponds to the event $\overline{H}$).

It remains, then, to show that the probability of a forging attack — namely, $\Pr(\mathsf{Succ}^{\mathrm{ake}}(\mathcal{A}) \wedge H)$ — is negligible in λ. The analysis proceeds by considering two main complementary cases:

- Event E1 [Ust08b, §4.1]: There is an honest party $\hat{B}$ such that $\mathcal{A}$ queries $H_2(*, b)$ before issuing a $\mathsf{RevealStaticKey}(\hat{B})$ query. In this case, a simulation $\mathcal{S}$ that is perfect (except with negligible probability) can be efficiently constructed with probability at least $1/p(\lambda)$ where p is a polynomial in λ.
- Event E2 [Ust08b, §4.2]: For every honest party $\hat{B}$ for which $\mathcal{A}$ queries $H_2(*, b)$, $\mathcal{A}$ queried $\mathsf{RevealStaticKey}(\hat{B})$ before the first $H_2(*, b)$ query.
  - Event E2 $\wedge$ M1: The test session has a matching session owned by an honest party.
  - Event E2 $\wedge$ M2: No party owns a matching session to the test session.

  Similarly to Event E1, each of (E2 $\wedge$ M1) and (E2 $\wedge$ M2) can be efficiently solved with polynomially bounded probability.

Since these cases are complementary, combining them shows that a simulation $\mathcal{S}$ can be constructed efficiently that transforms an attack on the security of CMQV into a solver for the Gap Diffie-Hellman problem. Under the Gap Diffie-Hellman assumption, this is impossible, so the CMQV protocol is secure. □

5.4.4 Denial of service resilience analysis

In this section, we show that DoS-CMQV is denial-of-service-resilient according to Definition 5.4. Since this definition (and the subsequent definition of a puzzling relation) includes the intentionally vague terms "expensive operation", "easy", and "hard", we need to define what these terms mean for a concrete instantiation of the definition.

For our purposes, an *expensive operation* is one of the following operations: storing a per-connection or per-session value in memory (other than a long-term value), performing a group exponentiation, or making a large number of calls (say, more than 2^{10}) to a hash oracle.

We first argue that the hash function construction we use is a puzzling relation, then that DoS-CMQV is a denial-of-service-resilient protocol using that puzzling relation.

Lemma 5.6 *Let $\mathcal{R}$ be defined such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ if and only if*

$$H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots20]} = 0 \ldots 0 \ ,$$

where H_1 is a random hash function. Then $\mathcal{R}$ is a puzzling relation, where "hard" means requiring approximately 2^{20} hash function queries on average, and "easy" is something that is not an "expensive operation".

Argument. Deciding membership in $\mathcal{R}$ is easy for a particular tuple because it involves only a single call to H_1.

Given $\hat{A}, \hat{B}, \mathsf{ch}$, we need to produce re such that $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots20]} = 0 \ldots 0$. To find such an re requires finding a preimage for the random hash function. The oracle U helps us find other preimages of H_1, but we cannot ask U for preimages involving $(\hat{A}, \hat{B}, \mathsf{ch})$. Our task, then is to find a preimage of the correct format involving $\hat{A}$, $\hat{B}$, and ch. Since H_1 is a random hash function outputting 20 bits, this is a hard task that requires approximately 2^{20} queries on average. □

This hash puzzle is similar to the partial inversion proof of work (PIPOW) problem of Jakobsson and Juels [JJ99, §3.1], and the exact tradeoff between work and probability of success can be calculated. By their Claim 1, we know that any prover $\hat{A}$ with memory bounded by m who performs on average at most w steps of computation, and is given $(\hat{A}, \hat{B}, \mathsf{ch})$, can find a response re such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ with probability at most $p + o(m/2^{20})$ where $p = 1/(2^{20} - w)$.

We now show that our protocol is resilient to denial of service attacks by the following claim:

Theorem 5.7 *The DoS-CMQV protocol is a denial-of-service-resilient protocol, where "easy", "hard", and "expensive operation" are defined as above.*

Argument. Let $\mathcal{R}$ be the puzzling relation from Lemma 5.6.

Let $[\hat{A}, \hat{B}, \mathsf{ch}]$ be a pre-session. According to the protocol, $\hat{B}$ does not perform any expensive operation until line 13, which is not reached unless the server's checks on lines 9 and 10 are passed and a new session is established on line 11.

If the check on line 10 is passed, namely if $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}[1], \mathsf{re}[2])_{[1\ldots20]} = 0 \ldots 0$, then $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$.

If the check on line 9 is passed, namely if $\mathsf{ch}[2] = H_0(\rho, \hat{A}, \hat{B}, \mathsf{ch}[1])$, then, except with negligible probability, $\mathsf{ch}[2]$ was generated only be someone who knew both ρ and $\mathsf{ch}[1]$. Since $\hat{B}$ is a DoS-unexposed party, no $\mathsf{DoSExpose}(\hat{B})$ query could have been issued and since ρ is only ever used as an input to a random oracle, only $\hat{B}$ knows ρ. Thus, $[\hat{A}, \hat{B}, \mathsf{ch}]$ is an acceptable pre-session.

Hence, $\hat{B}$ establishes a session only if the corresponding pre-session is acceptable and the tuple is in the puzzling relation $\mathcal{R}$. Note that since sessions must be unique within a party, the server only performs these expensive operations once per session.

Thus, DoS-CMQV is a denial-of-service-resilient protocol in the sense of Definition 5.4. □

5.4.5 Instantiation

Since it builds on the CMQV protocol, DoS-CMQV can be instantiated over any group for which the Gap Diffie-Hellman problem is hard. As we note in Section 2.2.2, one such group is the set of points under addition on certain elliptic curves of the form $y^2 = x^3 + 2x \pm 1$ over fields $\mathbb{F}_{3^\ell}$. The security analysis in Theorem 5.5 does not include explicit constants: it refers back to the security analysis of [Ust08b], which is only a polynomial space/time analysis. In particular, a key constant resulting from the use of the Forking Lemma [PS96, PS00] remains uncalculated. However, a CK01-secure protocol is given with a full analysis in [Ust08a, §2.3.2].

In practice, H_1 as used in the puzzling relation could be implemented by using a standard cryptographic hash function, such as SHA-1, and truncating the output to the first w bits. The puzzle used in our DoS-CMQV protocol could be tuned by the server based on the load it experiences. In times of light load, the server could require that clients truncate only to the first 5 or 10 bits of output, but in heavier load could require that clients truncate to 20 or 25 bits of output to make the cost of mounting a denial of service attack higher. For $w = 20$, it takes just under 3 seconds to perform 2^{20} SHA-1 evaluations on one core of our 2 GHz Intel Core 2 Duo processor using OpenSSL 0.9.7ℓ. This may be an acceptable computational burden for the client in many scenarios.

5.5 Other constructions

In this section, we describe other constructions for achieving denial of service resilience in key exchange protocols. In Section 5.5.1, we discuss implementing the puzzling relation with other types of puzzles. In Section 5.5.2, we also compare our construction with the denial of service techniques used in other protocols: the JFKi protocol (for which we show that our model can accommodate the technique used in that protocol, albeit with sufficiently weak definitions of "expensive operation" and "hard") and the Host Identity Protocol.

5.5.1 Memory-bound puzzling relations

While the protocol given in Section 5.4 uses a puzzling relation based on finding preimages in a hash function, other types of puzzling relations are possible, demonstrating the flexibility of our framework. Abadi *et al.* [ABMW03], for example, describe puzzles in which memory access time provides an expected lower bound on the time it takes to solve the puzzle, removing disparities in processor speed between large computers and small devices.

Care must be taken in choosing parameters for memory-bound puzzles. The cost incurred by a server in setting up one of these memory-bound puzzles, while much less expensive than the cost incurred by a client solving the puzzle, can still be significant. For example, in the memory-bound puzzles of Abadi *et al.*, it took a 2.4 GHz Pentium 4 server approximately 2^{-7} seconds to create a puzzle that takes a client approximately 2^2 seconds to solve. By comparison, the cost of doing one 1024-bit modular exponentiation on a computer of similar speed is only 2^{-9} seconds: setting up or verifying a memory bound puzzle can in fact be an expensive operation and thus care is required when using memory-bound puzzles for denial of service resilience.

Additionally, our construction includes the client's identity, server's identity, and client's ephemeral public key in the puzzle challenge. Since the challenges for memory-bound puzzles must be constructed in a special way, care must be taken in including these values in constructing the challenge of a memory-bound puzzle.

5.5.2 Stateless connections and cookies

One of the problems when comparing different denial of service countermeasures comes from the fact that they typically aim to guard against a specific attack at the expense of other properties. Stateless connections aim to reduce the amount of data the server has to store. However, servers are required to dedicate more computational resources, for example, by encrypting and decrypting session-specific secret data, and use more bandwidth to send augmented messages. In the case of stateless connections, the "puzzling relation" that the adversary has to solve is to present the server with a message that the server will accept as valid. As a result, the "proof of work" consists of the fact that the adversary stored information that was created by the server.

Cookies can be seen as a weaker form of stateless connections: the proof of work is the ability to replay a cookie, thus proving to the server that the client stored session-related information.

In general, denial of service countermeasures in which a server accepts only certain messages and as a result has some assurance that the message sender performed a minimum amount of work fit into the framework of "puzzling relation" and "proof of work". Note that in [AN97] the authors admit that the relation between a protocol and session keys may be broken. This fact is recognized in our model by

the fact that a single pre-session can theoretically lead to many different sessions if the same challenge is solved multiple times.

5.5.2.1 JFKi

In the JFKi protocol [ABB+04, §2.3], the denial of service resilience goal for the server is to avoid expensive operations unless the client echoes back the cookie.

There are two main ideas used: reuse of ephemeral public keys and use of a keyed hash function. The purpose of reusing ephemeral public keys is to distribute the cost of an expensive operation across multiple sessions. This allows the authors to argue the client must perform his share of the work first, in terms of bearing the cost of establishing a round communication trip. The keyed hash function is used by the server to verify that the client indeed executed the round. Note that the server does not need to dedicate any resources to verify the challenge was created by the server. This can be viewed as the pre-session stage of the protocol since the goal of the first round trip is to filter out bogus connections.

JFKi can be described in our model of denial of service resilience, but with weak definitions of "hard" in the puzzling relation. In the first stage of JFKi, the client $\hat{I}$ picks a nonce value N_I and sends $N_I' = H(N_I)$ to the server. The server $\hat{R}$ responds with its own nonce value N_R, its ephemeral public key g^r, and a keyed hash (as in a stateless connection) $H_{\text{HK}_R}(g^r, N_R, N_I', \hat{I})$ of these values under a private key HK_R. In order to satisfy the implied puzzling relation in JFKi, the client must echo back to the server all these values as well as the preimage N_I, and all appropriate hash relations must be satisfied. Additionally, the client sends its ephemeral public key g^i. If the puzzling relation test passes, then the server establishes a new session and computes $\mathsf{DH}(g^i, g^r)$.

The problem with JFKi's puzzling relation is that there is no binding between the client's ephemeral public key and the solution to the puzzling relation. A dishonest client can use the same solution to the puzzling relation with different ephemeral public keys (and it can generate these ephemeral public keys very cheaply, for example, by generating g^i, $g \cdot g^i = g^{i+1}, g \cdot g^{i+1} = g^{i+2}, \dots$) to cause the server to perform many exponentiations with little cost to the client. It is not very hard to produce many responses to the puzzling relation in Condition 2 of Definition 5.2.

This means that JFKi does not satisfy informal goal DoS-4, namely that the protocol should be resilient to flooding attacks. It is easy for a malicious client to prepare many valid connection requests and cause the server to establish many sessions. DoS-CMQV avoids this problem: producing a solution to the puzzling relation involves finding the preimage in the hash function and the values cannot be repeated if the server is to establish a new session.

One approach to fixing the denial of service resilience of JFKi is given by Smith *et al.* [SGNB06]. They note that JFKi is not denial-of-service-resilient when analyzed under Meadows' framework [Mea99]. They use a hash function preimage

puzzle as well to bind the puzzle solution to the key exchange session at hand. Their construction still preserves a fundamental design characteristic of JFKi: denial of service resilience is balanced against ephemeral key freshness. The responder must reuse its ephemeral private key in order to achieve denial of service resilience, thus preventing full freshness in the Canetti-Krawczyk model.

5.5.2.2 Host Identity Protocol

The Host Identity Protocol (HIP) [MNJH04, §4.1.1] is designed to offer protection against denial of service attacks. HIP uses a similar puzzling relation to that of Section 5.4: the client must find a preimage in SHA-1 such that the k lowest-order bits of the output are zero. HIP includes the identities of the initiator and responder in the hash function computation as we have done. Our model provides a justification of the design of HIP for denial of service resilience and the value of including the client and server identities in the hash function computation.

Chapter 6

Unified Point Addition Formulæ in Elliptic Curve Cryptography

Contents

6.1 Introduction

Elliptic curve cryptography (ECC) is one of the main forms of public key cryptography in use today. Compared with other public key cryptosystems such as RSA, ECC offers higher security per key bit given the known classical attacks, which can lead to more efficient implementations and smaller key sizes that are suitable for constrained devices such as smart cards and sensors.

The main operation in ECC is *scalar-point multiplication*. In order to make point multiplication efficient, implementations often use formulæ, such as point addition using projective coordinates, that decrease the number of expensive operations. A common method for implementing point multiplication is the *double-and-add* algorithm which makes use of repeated point doubling operations and point addition operations which depend on the key bits.

Implementations of ECC, however, are often at risk from side-channel attacks which make use of information observed during the execution of the algorithm, and these problems can be magnified when in constrained environments like smart cards. A **side channel** can reveal information about secret values as a result of the physical implementation of the procedure. For example, observing the power usage of a device while performing a cryptographic operation can reveal information about the sequence of low-level calculations performed, and this may then reveal information about the cryptographic key. Security against side-channel attacks needs to be addressed at three levels: the hardware level, the software level, and the algorithmic level.

In elliptic curve cryptography, unified point addition formulæ [BJ02, BDJ04] aim to reduce the amount of information leaked over side channels by using the same sequence of field operations for both point addition and point doubling. However, even these are not a complete solution: Walter [Wal04] gave a theoretical side-channel attack on an implementation of one unified point addition formula that, instead of exploiting any irregularity in the sequence of field operations performed, exploits an irregularity in the implementation of the field operations themselves in the context of the unified point addition formula.

Contributions. In this chapter, we give a projective version of the unified point addition formulæ of Brier, Déchène, and Joye. We extend Walter's attack to make use of a conditional addition which appears in many field subtraction implementations. This *conditional modular reduction attack* substantially decreases the amount of work necessary to recover the key: when used with projective coordinates and Montgomery field representation and combined with Walter's original technique, our attack is feasible on prime field elliptic curves up to 384 bits. We suggest some countermeasures which may help achieve constant run-time field operations. We also provide some performance results for the various unified point addition formulæ and discuss the applicability of timing attacks.

Future directions. In Section 6.3.3.3, we describe countermeasures that could help avoid our conditional modular reduction attack. It would be interesting to see if an experimental side-channel analysis of an implementation making use of these countermeasures does indeed unify the field operations or if further side-channel effects, such as cache timing, remain.

Our approach for analyzing the search space resulting from our side-channel attack is to concentrate on keys with a minimal number of unidentified additions, which may not correspond to keys for which the remaining keyspace is minimized. In general, binary keys where the zeros only appear in small groups in the key are much easier to break than keys with large groups of consecutive zeros. We note that, if we identify a point addition, then we have also identified its adjacent point doublings. For example, the 521-bit key $10101 \cdots 0101$ with 16 unidentified additions has a much smaller remaining keyspace than the 521-bit key consisting

of 261 ones followed by 260 zeros for which only 8 of the additions have not been identified, even though the number of additions remaining to be located is twice that of the second key. A more thorough cost analysis based on this observation could be pursued but would require a study of the distribution of strings of zeros in the binary representation of a given key, which is beyond the scope of the analysis in Sections 6.3.3.1 and 6.3.3.2.

Outline. The rest of this chapter is organized as follows. Section 6.2 provides a short review of the relevant issues in elliptic curve cryptography, including side-channel attacks and projective point addition formulæ. In Section 6.3, we discuss unified point addition formulæ over prime fields. In particular, we present the unified formula of Brier and Joye (Section 6.3.1), the unified formula of Brier, Déchène, and Joye (Section 6.3.2), our conditional modular reduction side-channel attack (Section 6.3.3), and report some timings (Section 6.3.4). Finally, we briefly discuss the (in)applicability of these techniques to binary fields in Section 6.4.

6.2 Background

Basic elliptic curve cryptography and affine coordinates. We recall from Section 2.1.3 that, for an elliptic curve E over a field F of prime characteristic q other than 2 or 3, the set of points $E(F)$ on the curve, joined with the point at infinity, forms an abelian group under the operation of point addition. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, $P \neq -Q$, be two points on the curve E. These two points can be added to obtain a third point $P+Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \text{ (addition)} \\ \frac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \text{ (doubling)} \end{cases} . \tag{6.1}$$

These points are represented using **affine coordinates**.

The most important operation in elliptic curve cryptography is **point multiplication**. For example, in the MFPAK protocol (Section 4.4.2) and the DoS-CMQV protocol (Section 5.4.2), the group operation, when the group is implemented using an elliptic curve, would be point multiplication. A standard algorithm for point multiplication is the **double-and-add algorithm**, given in Figure 6.1. The algorithm iterates over each key bit and performs a point doubling each time; if the key bit is 1, it also does a point addition.

Projective coordinates. The formula given in equation (6.1) uses affine coordinates. However, this formula for λ requires an inversion, and inversion operations, especially in prime fields, can be very expensive in practice: various software implementations of elliptic curve cryptography in prime fields report inversions taking around 60-80 times longer than field multiplications [BHHM01, HMV04].

Input:	Point P, integer $k = \sum_{i=0}^{n-1} k_i 2^i, k_{n-1} = 1$.
Output:	Point $Q = kP$.
1.	$Q \leftarrow P$
2.	for $i = n - 2$ down to 0 do
2.1.	$Q \leftarrow 2Q$
2.2.	if $k_i = 1$ then $Q \leftarrow P + Q$
3.	end for

Figure 6.1: Double-and-add point multiplication algorithm

This has motivated the development of formulæ using **projective coordinates** to reduce the number of field inversions required. In the ordinary projective case, a point is represented by three coordinates, $P = (X, Y, Z)$, with $x = X/Z$ and $y = Y/Z$. The denominator, Z, is updated through all of the point additions and point doublings in a point multiplication, and only at the end is the inversion Z^{-1} computed to return the final result to affine coordinates. Other variations can be used for improved efficiency, such as Jacobian coordinates (which have $x = X/Z^2$ and $y = Y/Z^3$) and Chudnovsky coordinates [CC86].

Side-channel attacks. Side-channel attacks were first proposed in the literature by Kocher [Koc96], although were used in espionage at least as early as 1956 [Wri87]. These types of attacks use information observed during the execution of the algorithm to help to determine the secret key. There are two main classes of side-channel attacks: *simple* side-channel attacks, which analyze data observed during a single execution of a cryptographic protocol, and *differential* side-channel attacks, which compare the traces of multiple executions of a protocol. The types of attacks we consider in this chapter relate to simple side channels. Side-channel attacks have been demonstrated experimentally using timing [Koc96], power analysis [KJJ99], and electromagnetic emissions [AARR02].

Because λ in equation (6.1) is defined differently for the cases when $P \neq Q$ and when $P = Q$, the formula for point addition is different from the formula for point doubling. This difference motivates the study of side-channel attacks in elliptic curve cryptography.

If a side-channel analysis allows us to distinguish a point addition from a point doubling, then we can determine secret key bits.

In the context of elliptic curve cryptography, techniques for counteracting simple side-channel attacks include: performing dummy operations, such as forcing a point addition at each iteration [Cor99]; using alternative point multiplication algorithms, such as Montgomery point multiplication [Mon87, LD99]; using alternative curve parameterizations, such as the Jacobi or Hessian forms; and unifying the algorithms for point addition and point doubling so that they use the same sequence of field

operations and hence are indistinguishable [BJ02, BDJ04]. It is this last technique that we address in this chapter. Care must be taken when using these techniques, however. If one countermeasure is used (for example, unified point addition formula from [BJ02]) but other aspects remain unprotected, attacks remain possible as was demonstrated by the attack of Walter [Wal04] on the unified point addition formula of Brier and Joye [BJ02] which takes advantage of a conditional operation in Montgomery modular reduction.

Point multiplication is the central operation when using elliptic curves for cryptography. Because the occurrences of point additions in the double-and-add algorithm (Figure 6.1) correspond precisely to the locations of 1 bits in the key, any difference in the implementation of the point doubling algorithm compared to the point addition algorithm could be exploited in a side-channel attack.

For example, the diagrams in Figure 6.2 (from [Osw05]) show power traces over time of point addition, point doubling, and point multiplication. The power trace for point doubling (second diagram) has a distinctive pattern at the end of the trace compared with point addition (first diagram). When a sequence of these operations is viewed together (third diagram), this distinctive pattern can be used to pick out where the point doublings occur, and thus where the point additions occur, leading to a full recovery of the key bits.

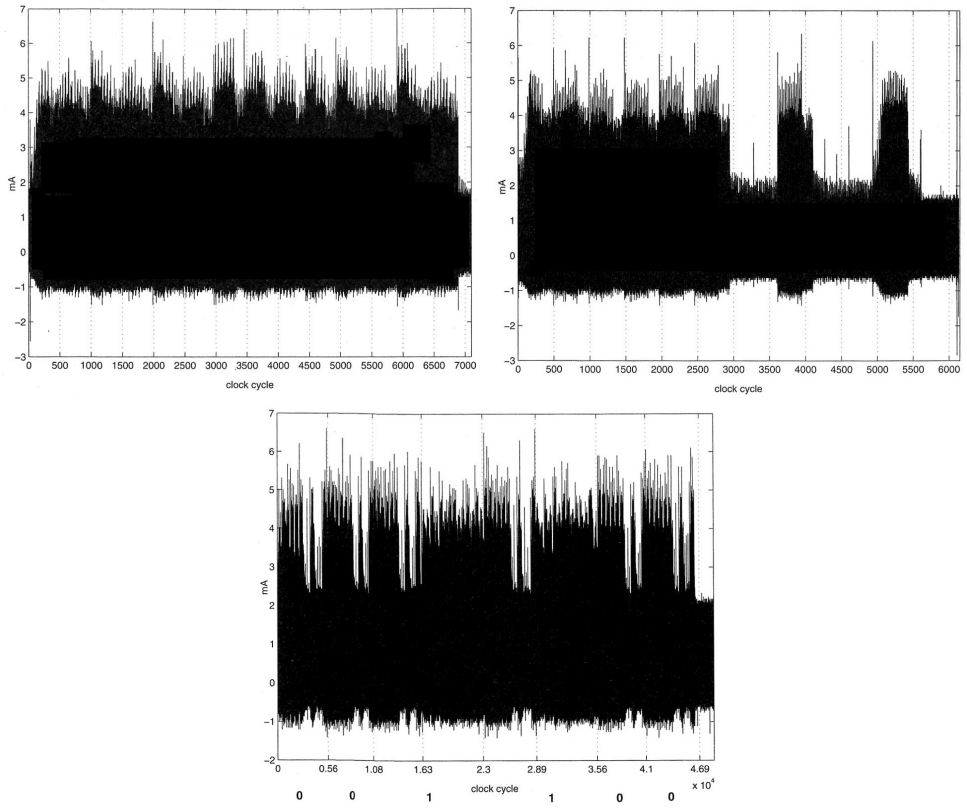

Figure 6.2: Point addition, double, and multiplication power traces from [Osw05].

Montgomery modular reduction. Montgomery's modular reduction technique [Mon85] replaces the modular reduction step after each field multiplication with a less expensive step, leaving a result that is not completely reduced. This partial result can be used again in further operations, with the expensive modular reduction step being left for the end. Though Montgomery's technique is not efficient for a single modular multiplication, it is effective when amortized over a long sequence of operations, such as a point multiplication.

Montgomery modular reduction is implemented as follows (*cf.* [HMV04]). Let q be an odd prime, and let $R > q$ with $\gcd(R, q) = 1$; it is often convenient to choose $R = 2^{Wt}$, where W is the word size of the computer architecture. For an input $z < Rq$, Montgomery reduction gives $zR^{-1} \mod q$. An algorithm for Montgomery modular reduction is given in Figure 6.3.

Input:	Integers q, R, z, q' with $q' = -q^{-1} \mod R$, $R > q$, $0 \leq z < Rq$.
Output:	Integer c such that $c = zR^{-1} \mod q$, $0 \leq c < q$.
1.	$c \leftarrow (z + (zq' \mod R)q)R^{-1}$
2.	if $c \geq q$ then $c \leftarrow c - q$ /* conditional subtraction */

Figure 6.3: Montgomery modular reduction algorithm

6.3 Unified point addition formulæ for prime fields

One approach to reducing the risk of side-channel attacks is to use a **unified point addition formula** in which the formulæ for point addition and point doubling are the same. Since there is no difference between the addition and doubling operations, a point multiplication trace should no longer act as a side channel.

The first unified point addition formulæ for elliptic curves in Weierstraß form was given by Brier and Joye [BJ02] in 2002. Walter [Wal04] proposed a side-channel attack that did not exploit the sequence of underlying field operations in the unified point addition formula directly but instead exploited an irregularity in the implementation of the underlying field operations in the context of the unified point addition formula. A subsequent paper of Brier, Déchène, and Joye [BDJ04] offered an infinite family of unified point addition formulæ in affine form.

6.3.1 Unified formula of Brier and Joye

Affine coordinates. The point addition formula for λ of equation (6.1) (when $P \neq Q$) is unsuitable for point doubling because $x_1 = x_2$ in the case of doubling, and hence the denominator is $x_2 - x_1 = 0$. However, starting with this λ in the

point addition formula, Brier and Joye [BJ02] obtain a different λ that can be used for both point addition and point doubling:

$$\lambda = \frac{(x_1 + x_2)^2 - x_1 x_2 + a}{y_1 + y_2}, \quad \text{if } y_1 + y_2 \neq 0 \ . \tag{6.2}$$

This alternative formula for λ works when $x_1 = x_2$, which is the case for doubling, but is not defined in the rare case when $y_1 + y_2 = 0$; the subsequent work of Brier, Déchène, and Joye removed this constraint (see Section 6.3.2).

The formula requires 2 field multiplications, 1 field squaring, and 1 field inversion.

Projective coordinates. In the same paper, Brier and Joye also derive a formula for unified point addition using projective coordinates, starting from the unified value of λ. By setting $x_i = X_i/Z_i$ and $y_i = Y_i/Z_i$, they obtain:

$$X_3 = 2FW \ , \quad Y_3 = R(G - 2W) - L^2 \ , \quad Z_3 = 2F^3 \ , \tag{6.3}$$

where $U_1 = X_1 Z_2, U_2 = X_2 Z_1, S_1 = Y_1 Z_2, S_2 = Y_2 Z_1, Z = Z_1 Z_2, T = U_1 + U_2, M = S_1 + S_2, F = ZM, L = MF, G = TL, R = T^2 - U_1 U_2 + aZ^2$, and $W = R^2 - G$.

This projective formula requires 13 field multiplications and 5 field squarings.

In Appendix A.1.1.1, we provide C code (using the NSS toolkit [Moz08]) that implements the projective coordinates formula in equation (6.3).

6.3.1.1 Walter's attack

In 2004, Walter [Wal04] described a side-channel attack on the unified formula of Brier and Joye under the additional assumption that the occurrence of a conditional subtraction in a Montgomery modular reduction operation can be detected. Such an attack should be viewed as successful if a non-negligible fraction of the keys can be computed significantly faster than attacking the whole keyspace. Walter's attack is successful since a high proportion of keys can be found using relatively few computations.

The attack. Walter's attack exploits the ability to detect a conditional subtraction in Montgomery modular reductions during double-and-add point multiplication with the unified point addition formula of Brier and Joye using projective coordinates.

For point doubling using projective coordinates, we have that $X_1 = X_2$, $Y_1 = Y_2$, and $Z_1 = Z_2$. In equation (6.3), the formulæ for $U_1 = X_1 Z_2$ and $U_2 = X_2 Z_1$ are identical, as are the computations of $S_1 = Y_1 Z_2$ and $S_2 = Y_2 Z_1$.

Consider the implementation of these various multiplication operations. During a point doubling, a conditional subtraction occurs in the Montgomery reduction for

$U_1 = X_1 Z_2$ if and only if a conditional subtraction occurs in the (same) Montgomery reduction for $U_2 = X_2 Z_1$. Thus, if a conditional subtraction is observed in the computation of either U_1 or U_2 but not the other, then a point doubling could not have occurred and the operation must be a point addition. By the same argument, the observation of a conditional subtraction in the computations of one of S_1 and S_2 but not the other can also distinguish a point addition.

Probabilistic analysis. Suppose that every pair of inputs to Montgomery multiplication is equally likely and independent. The probability that a conditional subtraction occurs in the computation of one of U_1 or U_2 but not the other (and similarly for S_1 and S_2) is

$$p_{\text{diff}} = 2p_{\text{sub}}(1 - p_{\text{sub}}) \approx \frac{3}{8} \tag{6.4}$$

where p_{sub} is the probability of a conditional subtraction occurring. Assuming random inputs to Montgomery modular multiplication, in practice $p_{\text{sub}} \approx 1/4$.

The probability that conditional subtractions give observable differences in the computations of U_1 and U_2 and S_1 and S_2 during point addition (and thus distinguish a point addition from a point doubling) is

$$p_{\text{dist}} = 1 - (1 - p_{\text{diff}})^2 \approx \frac{39}{64} \approx 0.61 \ . \tag{6.5}$$

During double-and-add point multiplication, the position of a point addition determines the point doublings on either side of it. The goal then will be to use detected point additions to determine as many operations as possible by noting adjacent point doublings, although a pair of subsequent point additions will both determine the intermediate point doubling. Let n be the size in bits of the prime field. For a given p_{dist}, the expected total number of determined operations is:

$$\frac{3}{2}(n-1)p_{\text{dist}} - (n-2)\left(\frac{1}{2}p_{\text{dist}}\right)^2 \ . \tag{6.6}$$

Once these positions have been determined, a search of the keys compatible with the remaining undetermined positions is undertaken until the correct key is found.

Walter's analysis provides additional methods of decreasing the key space that needs to be searched, including substring restrictions on the possible sequence of point additions and point doublings; furthermore, for the point addition operations which were not distinguished, some combinations of conditional subtractions are more likely than others, and this can also reduce the key space.

The probabilistic analysis resulting from combining equations (6.5) and (6.6) does not correspond to the best estimate in practice. Walter found in his experiments that, with a set of 512 sample traces, it was most efficient to just pick the sample that has the greatest number of distinguished point additions, and that this

sample would in fact have more distinguished point additions than the analysis above suggested. This greedy approach, combined with additional substring restrictions, could give effective keyspaces for a 192-bit prime curve of size just $2^{17.6}$, which can be easily searched. Our analysis in Section 6.3.3 gives a probabilistic argument that generalizes the experimental sampling that Walter used.

6.3.2 Unified formula of Brier, Déchène and Joye

The unified point addition formula in the previous section is defined for $y_1 + y_2 \neq 0$. Since the field is of odd prime characteristic q, this always holds in the case of point doubling (since $y_1 = y_2$ and, for all y, $2y \not\equiv 0 \mod q$), but it does not hold for all possible point additions. Izu and Tagaki [IT03] showed collisions for which $y_1 + y_2 \neq 0$ on many standardized elliptic curves. They also showed how these special cases could be used to reveal the key in some cryptosystems, with an example for the plain ElGamal encryption scheme.

As a result, Brier, Déchène, and Joye [BDJ04] developed an infinite family of unified point addition formulæ which are defined for all points.

Affine coordinates. We choose to work with the most efficient formula of the family in the rest of this section and thus fix a single formula for λ. Although Brier, Déchène, and Joye give an infinite family of unified point addition formulæ, which would allow a different λ value to be randomly chosen at each point addition, we assume that the most efficient formula, given in equation (6.7), is used each time.

The formula for λ is as follows:

$$\lambda = \frac{(x_1 + x_2)^2 - x_1 x_2 + a + (-1)^\delta (y_1 - y_2)}{y_1 + y_2 + (-1)^\delta (x_1 - x_2)} \ , \tag{6.7}$$

where δ is chosen such that $\delta = 0$ when $y_1 + y_2 + x_1 - x_2 \neq 0$ and $\delta = 1$ otherwise (or a randomized choice of δ when both choices give nonzero values).

The formula requires 2 field multiplications, 2 field squarings, and 1 field inversion.

In Appendix A.1.2.1, we provide C code (using the NSS toolkit [Moz08]) that implements the affine coordinates formula in equation (6.7).

Projective coordinates. Projective coordinates can be used to avoid the high cost of field inversion by trading expensive field inversions for cheaper field multiplications. In so doing, field inversion need only be done once per point multiplication (at the end) rather than during point addition or point doubling in the double-and-add algorithm.

We now give a projective form of the unified point addition formula corresponding to the projective form given with λ as defined in equation (6.7).

We begin by noting that, since $P + Q = Q + P$, the value for y_3 in point addition is symmetric and hence $2y_3 = \lambda(x_1 + x_2 - 2x_3) - (y_1 + y_2)$. Let $x_i = X_i/Z_i$, $y_i = Y_i/Z_i$. By completing the square in the numerator of λ, we obtain:

$$X_3 = 2FW \ , \quad Y_3 = R(G - 2W) - LFM \ , \quad Z_3 = 2F^3 \ , \tag{6.8}$$

where $U_1 = X_1Z_2, U_2 = X_2Z_1, S_1 = Y_1Z_2, S_2 = Y_2Z_1, Z = Z_1Z_2, T = U_1 + U_2, M = S_1 + S_2, V = (-1)^\delta(U_1 - U_2), N = (-1)^\delta(S_1 - S_2), E = M + V, F = ZE, L = FE, G = LT, R = T^2 - U_1U_2 + Z(aZ + N)$, and $W = R^2 - G$. Note that $\delta = 0$ when $S_1 + S_2 + U_1 - U_2 \neq 0$ and $\delta = 1$ otherwise; we can also make a randomized choice of δ when both choices give nonzero values.

This formula requires 16 field multiplications and 3 field squarings.[1]

In Appendix A.1.3.1, we provide Maple code that checks that the projective formula of equation (6.8) corresponds to the affine formula of equation (6.7).

In Appendix A.1.3.2, we provide C code (using the NSS toolkit [Moz08]) that implements the projective coordinates formula in equation (6.8).

6.3.3 Extending Walter's attack: conditional modular reduction attack

Walter's attack in Section 6.3.1.1 worked under the assumption that the conditional subtraction at the end of Montgomery reduction could be detected. We observe that there is a conditional subtraction (or addition) at the end of most implementations of field addition and subtraction. For example, in the field subtraction algorithm given in Figure 6.4, the conditional addition is step 2. This is a common practice in many implementations. For example, the OpenSSL [Ope08] library provides a function `BN_mod_sub_quick` which performs exactly the operations in Figure 6.4, and similarly for field addition. When reduction is done using the division, as in OpenSSL's `BN_mod_sub` function, and the value to be reduced is strictly between $-q$ and q, the operations performed can be mapped to those in Figure 6.4 and include a conditional addition.

Input:	Integers c, d, q such that $0 \leq c, d \leq q - 1$.
Output:	Integer e such that $e = c - d \mod q$ and $0 \leq e \leq q - 1$.
1.	$e \leftarrow c - d$
2.	if $e < 0$ then $e \leftarrow e + q$

Figure 6.4: Field subtraction algorithm

Thus, under the same assumption as in Walter's attack that a conditional subtraction (or addition) can be observed, we design an attack that makes use of

[1]The multiplication by $(-1)^\delta$ in the computation of V and N can be implemented with conditional branching (`if` statement). However, the use of conditional branching must be done with care to avoid a new vector for a side-channel attack.

detecting such an operation at the end of a field addition (or subtraction).

The key observation that enables the side-channel attack is that there are some modular subtractions in the unified point addition formulæ where, in the case of a point doubling, the terms in the subtraction are equal and hence the result of the subtraction is zero: when $c = d$, we compute $c - d \mod q$ as $c - d = 0$, and the conditional addition in field subtraction is never performed. However, if in the operation $c - d \mod q$ we observe the occurrence of a conditional addition, then it must be that $d > c$, and in particular that $c \neq d$, so the operation must be a point addition and not a point doubling.

We now consider how this observation can be used specifically against the affine and projective unified point addition formulæ of Section 6.3.2.

6.3.3.1 Effect on affine formula of Section 6.3.2

The attack. The affine unified point addition formulæ of Brier, Déchène, and Joye in equation (6.7) requires the computation of $y_1 - y_2 \mod q$ and $x_1 - x_2 \mod q$. Assuming these values are distributed uniformly at random, then the probability that a conditional addition occurs in the computation of $y_1 - y_2 \mod q$ is $1/2$, and similarly for $x_1 - x_2 \mod q$. In this case, the probability that a point addition can be identified is

$$p_{\text{dist}} = 1 - (1 - 1/2)^2 = 3/4 \ . \tag{6.9}$$

Probabilistic analysis. First, we note that, since a unified point addition and unified point doubling should have the same runtime, a side-channel attack on point multiplication will allow the number of point operations to be determined. Assuming the key length is known, we can then determine from the number of operations how many point additions occurred since the number of point doublings is fixed by the key length.

To simplify our analysis, we assume that the attack need only be successful for keys of the most common length. The attack can work for other key lengths, but requires a more careful analysis to bound the computational runtime of the attack.

Suppose the characteristic q of the prime field is between $3 \cdot 2^{r-1}$ and $3 \cdot 2^r$. Then among keys chosen uniformly at random between 0 and $q - 1$, the most common key length is r and this occurs for $1/3$ of the keys between 0 and $q - 1$. The closer q is to 2^{r+1}, the closer the fraction of keys having length r is to $1/2$.

If there are k point additions of which ℓ are not determined by the observation of a conditional addition, then the space we need to search is the set of sequences with $r - k$ zeroes and k ones combined with the constraints from the identified point additions in the double-and-add sequence. Furthermore, we can restrict some substrings based on the constraint that we cannot have two subsequent point additions, as each point addition must be preceded by a point doubling.

The number of possible keys is upper-bounded by $\binom{r-k+\ell}{\ell}$, which in turn is upper-bounded by $\binom{r}{\ell}$; this estimate is quite pessimistic but has the advantage of being independent of k, which we shall use in the analysis below.

Assume that all keys of length r are possible, which is true if $q \geq 2^{r+1} - 1$. Then the probability that point multiplication using a key of length r uses exactly k point additions is $\binom{r}{k}/2^r$. For a key with k point additions, the probability that exactly ℓ of them are not identified is

$$\binom{k}{\ell} p_{\text{dist}}^{k-\ell} (1 - p_{\text{dist}})^{\ell} \quad . \tag{6.10}$$

Since a key of length r corresponds to at most r point additions, the probability that exactly ℓ additions are not identified in a key of length r is

$$\begin{aligned}
p_\ell &= \sum_{k=\ell}^{r} (\text{prob. key of length } r \text{ uses exactly } k \text{ point additions}) \\
&\qquad \cdot (\text{prob. } \ell \text{ of these } k \text{ point additions are not identified}) && (6.11) \\
&= \sum_{k=\ell}^{r} \binom{r}{k} \frac{1}{2^r} \cdot \binom{k}{\ell} p_{\text{dist}}^{k-\ell} (1 - p_{\text{dist}})^{\ell} && (6.12) \\
&= \sum_{k=\ell}^{r} \frac{(1 - p_{\text{dist}})^{\ell}}{2^r} \binom{r}{\ell} \binom{r-\ell}{k-\ell} p_{\text{dist}}^{k-\ell} && (6.13) \\
&= \binom{r}{\ell} \frac{(1 - p_{\text{dist}})^{\ell}}{2^r} \sum_{i=0}^{r-\ell} \binom{r-\ell}{i} p_{\text{dist}}^{i} && (6.14) \\
&= \binom{r}{\ell} \frac{1}{2^r} (1 - p_{\text{dist}})^{\ell} (1 + p_{\text{dist}})^{r-\ell} \quad . && (6.15)
\end{aligned}$$

The average number of unidentified point additions is hence

$$\begin{aligned}
\sum_{k=0}^{r} k \cdot p_k &= \sum_{k=0}^{r} k \cdot \binom{r}{k} \frac{1}{2^r} (1-p_{\text{dist}})^k (1+p_{\text{dist}})^{r-k} && (6.16)\\
&= \frac{1}{2^r} \sum_{k=0}^{r} \frac{k \cdot r!}{(r-k)!k!} (1-p_{\text{dist}})^k (1+p_{\text{dist}})^{r-k} && (6.17)\\
&= \frac{1}{2^r} \sum_{k=1}^{r} \frac{k \cdot r!}{(r-k)!k!} (1-p_{\text{dist}})^k (1+p_{\text{dist}})^{r-k} && (6.18)\\
&= \frac{1}{2^r} \sum_{k=1}^{r} \frac{r!}{(r-k)!(k-1)!} (1-p_{\text{dist}})^k (1+p_{\text{dist}})^{r-k} && (6.19)\\
&= \frac{1}{2^r} \sum_{k=0}^{r-1} \frac{r \cdot (r-1)!}{(r-1-k)!k!} (1-p_{\text{dist}})^{k+1} (1+p_{\text{dist}})^{r-1-k} && (6.20)\\
&= \frac{(1-p_{\text{dist}})r}{2^r} \sum_{k=0}^{r-1} \binom{r-1}{k} (1-p_{\text{dist}})^k (1+p_{\text{dist}})^{r-1-k} && (6.21)\\
&= \frac{(1-p_{\text{dist}})r}{2^r} ((1-p_{\text{dist}}) + (1+p_{\text{dist}}))^{r-1} && (6.22)\\
&= \frac{(1-p_{\text{dist}})r}{2^r} 2^{r-1} && (6.23)\\
&= (1-p_{\text{dist}})r/2 \ . && (6.24)
\end{aligned}$$

Although the average number of unidentified point additions is $(1-p_{\text{dist}})r/2 = r/8$ when $p_{\text{dist}} = 3/4$ as in equation (6.9), some keys will have fewer additions remaining to be identified.

For example, consider the case of an elliptic curve over a 192-bit prime field. Here, we have $r = 191$ and, for $p_{\text{dist}} = 3/4$ as in equation (6.9), there are on average 23.9 point additions remaining to be identified, so the search space is still quite large. (This analysis assumes that the point additions in a point multiplication are independent which is not strictly true as x_1 and y_1, the x and y coordinates of the generator, are the same for all the point additions in a single point multiplication.)

A further refinement. We can apply a further refinement of the attack for special cases as suggested by Walter [Wal04, §7]. Suppose the generator P is selected uniformly at random. For $1/m$ of the cases, the x-coordinate of the base point P will take on a value between 0 and $\frac{1}{m}q$ and will have an average value of $\frac{1}{2m}q$. We will observe that we can observe even more conditional point additions in this case.

We adopt the notation that, in the double-and-add point multiplication algorithm, the fixed generator P is the first argument of the unified point addition formula. We assume that over a point multiplication, the values x_2 that arise

in each point addition in equation (6.7) will behave as if they are uniformly distributed. Thus, a $(1 - \frac{1}{2m})$-fraction of these values will be greater than x_1, so a $(1 - \frac{1}{2m})$-fraction of the point addition operations will have a conditional addition occurring. We do not make any restriction on the size of the y-coordinate of P and assume it remains uniformly distributed independent of the x-coordinate.

With this additional restriction on the x-coordinate of P, the probability that a point addition can be distinguished is the probability that a conditional addition occurs in either the computation of $x_1 - x_2$ or $y_1 - y_2$:

$$p_{\text{dist}} = 1 - \left(1 - \left(1 - \frac{1}{2m}\right)\right)\left(1 - \frac{1}{2}\right) = 1 - \frac{1}{4m} \ . \tag{6.25}$$

With this value of p_{dist} and with $1/m = 1/8$, the expected number of additions remaining to be identified decreases to $r/64$. For the example of a 192-bit curve with $r = 191$, this is a mere 2.99 unidentified point additions and occurs for 1/8 of the base points.

With this restriction on P, a significant proportion of keys of length r will be left with 3 or fewer unidentified additions. By contrast, without using this restriction on P and only making use of the distribution in equation (6.15), we would find that 1 in approximately 24.6 of all keys would leave 3 or fewer unidentified additions.

In this restricted case, the number of possible keys is now upper-bounded (loosely) by $\binom{191}{3} \approx 2^{20.1}$, for which an exhaustive search is quite feasible.

6.3.3.2 Effect on projective formula of Section 6.3.2

The attack. There are two operations in the projective formula of equation (6.8) where the ability to detect a conditional addition in a field subtraction can lead to a side-channel attack. Additionally, we can make use of Walter's original observation on Montgomery modular multiplication to enhance the attack further.

We will first describe where the detection of a conditional addition in field subtraction can be exploited. Suppose without loss of generality $\delta = 0$. Consider the operations $V = U_1 - U_2 \mod q$ and $N = S_1 - S_2 \mod q$. During a point doubling, $U_1 = U_2$ and $S_1 = S_2$, so there will be no conditional addition in either of these calculations. However, in the case of a point addition a conditional addition may occur.

Additionally, if the field multiplication is implemented using Montgomery modular multiplication, Walter's original attack that uses conditional subtractions in Montgomery reductions can still be applied to this projective formula. A point addition can be distinguished from a point doubling if a conditional subtraction occurs in the computation of one of $U_1 = X_1Z_2$ or $U_2 = X_2Z_1$ but not the other, and in one of $S_1 = Y_1Z_2$ or $S_2 = Y_2Z_1$ but not the other.

By combining these two sources of information, we can increase the number of determined point additions. We now have four different conditional events which

distinguish a point addition from a point doubling:

1. conditional subtraction in the computation of one of $U_1 = X_1Z_2$ or $U_2 = X_2Z_1$ but not the other,
2. conditional subtraction in the computation of one of $S_1 = Y_1Z_2$ or $S_2 = Y_2Z_1$ but not the other,
3. conditional addition in the computation of $V = U_1 - U_2 \mod q$, and
4. conditional addition in the computation of $N = S_1 - S_2 \mod q$.

Probabilistic analysis. As before, we will assume that over the course of a point multiplication U_1 and U_2 will behave as if they are independent and uniformly distributed over $0, \ldots, q-1$, and similarly for S_1 and S_2. Moreover, we will assume that the inputs to the multiplications $U_1 = X_1Z_2$, $U_2 = X_2Z_1$, $S_1 = Y_1Z_2$, and $S_2 = Y_2Z_1$ behave as if they are independent and uniformly distributed.

In each point addition, we have that $U_2 < U_1$ with probability $p_{\text{add}} = \frac{1}{2}$ in which case a conditional addition is needed in the computation of $V = U_1 - U_2 \mod q$, and similarly for $N = S_1 - S_2 \mod q$. If a side-channel trace reveals a conditional addition occurring in at least one of these computations, then the operation must be a point addition. The probability of distinguishing a point addition using these two events is $1 - (1 - 1/2)^2 = 3/4$. Unfortunately, we cannot assume that one of the arguments is small and take advantage of the further refinement we used in the affine case because the values in the subtractions $V = U_1 - U_2 \mod q$, and $N = S_1 - S_2 \mod q$ depend on both of the points of the point addition.

With the assumption that these four events occur independently, the probability of detecting a point addition given that the operation was a point addition is

$$p_{\text{dist}} = 1 - (1 - p_{\text{add}})^2(1 - p_{\text{diff}})^2 \ . \tag{6.26}$$

In practice, this assumption is justified as the coordinate values observed during a point multiplication do seem to behave as if they are sufficiently uniformly distributed and, with respect to the four conditional events above, are sufficiently uncorrelated.

Without making any assumption on the distribution of the base point, and using $p_{\text{add}} = 1/2$ (as above) and $p_{\text{diff}} \approx 3/8$ (as in equation (6.4)), we obtain from equation (6.26) that $p_{\text{dist}} \approx 231/256 \approx 0.902$. Based on the distribution of unidentified point additions in equation (6.15), the average number of unidentified point additions in a key of length r will be approximately $25r/512 \approx 0.049r$.

A further refinement. If we look for base points with conveniently sized coordinates along the lines of the further refinement of the previous section, the increase in the probability of success will be relatively small. For a projective point with

coordinates approximately $\left(\frac{1}{16}q, \frac{1}{16}q, \frac{15}{16}q\right)$, we will be able to improve p_{diff} to approximately 0.93, and then the expected number of unidentified point additions decreases to approximately $0.035r$. However, this decrease from $0.049r$ to $0.035r$ comes at the cost of restricting ourselves to 1 out of every 512 points that satisfies the constraints above.

Overall, it is much more practical to consider all base points and make use of the variance in the distribution of equation (6.15). For a 192-bit prime field, 15.4% of all point multiplications have will 6 or fewer unidentified point additions, whereas this refinement to 1 in 512 base points will have 6.7 unidentified point additions on average.

Unidentified substrings. Using the conditional modular reduction attack, we identify operations that are believed to be point additions, and then deduce that certain other operations, namely the adjacent operations, must be point doublings. To further refine the estimate of the number of unidentified point doublings, we introduce an approach that could also be used to detail the lengths of the unidentified substrings.

Our approach is asymptotic: we consider the frequencies that occur for a key of infinite length with the same proportion of zero bits and identified additions as our finite key, which then give us an estimate on the number of identified (and unidentified) doublings in our finite key.

The approach classifies doubling operations in three different ways, and uses a state diagram to describe how each operation can arise as we read through the bits of the key.

The state diagram, illustrated in Figure 6.5 has six states. There are three states corresponding to doublings that arise from moving from one bit of the key to the next: D_* (unidentified point doublings), D_1 (identified point doublings which precede an identified point addition), and D_2 (identified point doublings which follow but do not precede an identified point addition); There are three states that correspond to bit operations, that is, operations which depend on the value of the key bit: A (identified point addition, for the bit 1), A_* (unidentified point addition, for the bit 1), and V (absence of point addition, for the bit 0).

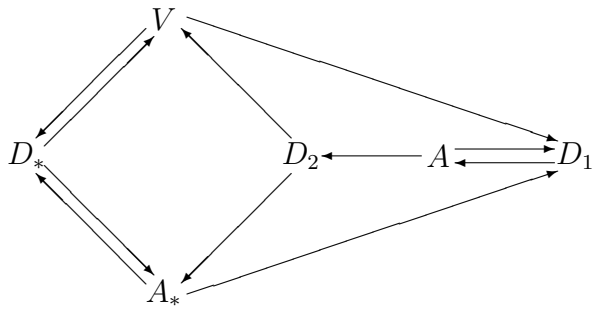

Figure 6.5: State diagram for analyzing point doublings

To estimate the number of identified and unidentified point doublings, we must find which proportion of the point doublings are in states D_1 and D_2. When in state D_1, the subsequent state must always be A, so the number of occurrences of D_1 must be the same as the number of occurrences of A, the number of identified point additions. State D_2 only arises following state A, and the probability of moving to D_2 when leaving A is the probability that the next bit does not correspond to an identified addition.

Recall that the finite key has length r. If the r-bit key has m identified point additions, then an estimate for the number of identified point doublings is

$$\left(1 + \frac{r-m}{r}\right) m \ . \tag{6.27}$$

In the expression above, the 1 (times m) indicates that a D_1 event arises for every A event, and $\frac{r-m}{r}$ is the probability of moving to D_2 when leaving A). Thus, the number of unidentified operations is upper-bounded by

$$\binom{r + \left(1 + \frac{r-m}{r}\right) m + k}{k} \ . \tag{6.28}$$

Attack analysis. Figure 6.6 describes the effect of these attacks at various field sizes with a number of different attack scenarios for a number of common field sizes.

In the first chunk of the table, we give for each field size $r + 1$ the average number of additions remaining to be identified based on the probabilistic analysis of equation (6.26) that yields $0.049r$ unidentified point additions.

In the second chunk of the table, we give an analysis based on what can be expected from the best of 512 sample traces. Here, we show the number k of unidentified point additions for which p_k from the distribution in equation (6.15) is greater than $1/512$, an upper-bound on this keyspace given by $\binom{r}{k}$, and a refined estimate on the keyspace using substring restrictions given by equation (6.28).

In the third chunk of the table, we look at what would be required if we wished to obtain a trace that had at most 3 unidentified point additions. Here, we show the number of traces that would be required to find a trace with at most 3 unidentified point additions (this is given by the expression $1/(p_0 + p_1 + p_2 + p_3)$ with p_k defined by equation (6.15)), and the upper-bound on this keyspace given by $\binom{r}{3}$.

In the fourth chunk of the table, we compare our results with the earlier attack of Walter [Wal04], showing the average number of unidentified point additions from that attack, the bound on the keyspace based on the average number of unidentified point operations, and a bound on the keyspace with an enhanced set of substring restrictions from Walter's experiments.

For our attacks, we make no restrictions on the size of the coordinates of the points, so $p_{\text{dist}} \approx 0.902$. We assume that half the keys have size r (that is, $q \approx 2^{r+1}$) and that an attack on a key of size different from r is always considered unsuccessful.

Field size in bits $(r+1)$	**160**	**192**	**224**	**256**	**384**	**521**
Average missing additions per point mult.	7.76	9.33	10.89	12.45	18.70	25.43
Sampling best trace from 512 samples						
k required for prob. of success $> 1/512$	2	2	3	4	8	13
Upper bound on keyspace for this k	$2^{13.6}$	$2^{14.1}$	$2^{20.8}$	$2^{27.4}$	$2^{53.2}$	$2^{84.5}$
Estimated keyspace (equation (6.28)):	$2^{9.87}$	$2^{10.4}$	$2^{15.2}$	$2^{20.0}$	$2^{38.8}$	$2^{61.5}$
To obtain at most 3 unidentified additions						
Expected number of keys required	22	67	217	746	$2^{17.1}$	$2^{25.2}$
Bound on keyspace	$2^{19.3}$	$2^{20.1}$	$2^{20.8}$	$2^{21.4}$	$2^{23.1}$	$2^{24.4}$
Walter's attack [Wal04]						
Average missing operations (τ)		36.2	45.9	55.7	96.3	141.6
Average missing additions (α)		19.2	23.0	26.6	41.5	57.9
Bound on keyspace (no restrictions) $(\binom{\tau}{\alpha})$		$2^{33.2}$	$2^{42.8}$	$2^{52.4}$	$2^{91.4}$	$2^{134.3}$
Bound using substring restrictions		$2^{17.6}$	$2^{24.0}$	$2^{30.4}$	$2^{56.0}$	$2^{84.2}$

Figure 6.6: Expected number of operations using conditional modular reduction attack, using $p_{\text{dist}} \approx 0.902$.

Our attacks compare favourably with Walter's, leading to improved keyspace bounds for all field sizes. In fact, our technique makes attacks on field sizes of up to 384 bits feasible as we note below, whereas Walter's attack is feasible only up to field sizes of 256 bits.

Based on the estimated keyspace for the best sample from among 512 samples, we note that even up to 384-bit curves our attack remains feasible given the cost of checking a key (a 384-bit elliptic curve point multiplication requires about $2^{24.3}$ operations on OpenSSL 0.9.8i [Ope08]) and the required search space ($2^{38.8}$).

Based on waiting to obtain at most 3 unidentified additions, the attack also remains feasible up to 384-bit curves with a total expected attack time of $2^{17.1} \cdot 2^{23.1} \cdot 2^{24.3} = 2^{64.5}$.

One interesting observation is that, for the 521-bit case, the expected number of traces required to obtain 3 unidentified additions approximately balances the bound on the keyspace: this gives, in some sense, an overall minimized complexity of attack.

It may be possible to use the substring restriction approach reported by Walter [Wal04, §8, 9] to decrease the keyspace needed to be searched. But this approach is likely to have only a limited impact in our cases, since the operations remaining to be identified consist of a large number of point doublings and a few point additions: sequences of zeros in the binary expansion will give rise to unavoidable unidentified point doublings, no matter how successful we are at identifying point additions.

6.3.3.3 Countermeasures

Our conditional modular reduction attack makes use of information leaked on the size of intermediate values based on the observation of the occurrence of a conditional addition in field subtraction. The obvious countermeasure is to ensure that field subtractions have a constant runtime so that no conditional addition can be observed. The standard technique of inserting dummy operations can create additional risks when differential side-channel attacks are considered.

Alternatively, we could implement all subtractions of the form $c - d \mod q$ as $(2q + c - d) - mq$, where $m \in \{1, 2\}$ is chosen based on the value of $2q + c - d$, so that the time required for a field subtraction is constant. Similarly, for field addition, one could implement $c + d \mod q$ as $(q + c + d) - mq$ with $m \in \{1, 2\}$. Finally Montgomery reduction can be adjusted similarly by replacing the conditional subtraction $c \leftarrow c - q$ on line 2 of Figure 6.3 with $c \leftarrow (c + q) - mq$ with $m \in \{1, 2\}$. Algorithms implemented with these countermeasures still have a branch in their implementation based on the appropriate value of m, but the branches have the same number of operations and thus may have less of a detectable difference in practice.

We could rewrite the unified point addition formula explicitly considering field reductions (both Montgomery reductions and addition/subtraction of a multiple

of q) as operations that are independent from the field multiplication, inversion, addition, and subtraction operations. We would have to accept that some of the intermediate values in the computation would not be always be fully reduced and thus could be greater than q or less than 0. In the case of field addition or subtraction, this is not very problematic as a single field addition could not get us too far outside of the desired range of 0 to $q-1$: the sum of two elements in this range is at most $2q-2$, requiring only a single bit more to record. When the field size is very close to a multiple of the word size, however, it may have a negative impact on efficiency if we need to use another whole word in the representation.

We note as well that correctly selecting δ in equation (6.8) requires the comparison of two field elements, so at least these two elements must be fully reduced (or we could do a field subtraction with reduction on them). To improve the situation, we compute δ such that

$$\frac{1}{2}(x_1 + x_2 + y_1 + y_2) \neq x_{2-\delta} \tag{6.29}$$

instead of $y_1 + y_2 + (-1)^\delta(x_1 - x_2) \neq 0$ as in equation (6.8): these expressions are equivalent but this new form is more unified.

This leads us to the following set of recommendations for implementing the projective unified point addition formula of equation (6.8). For simplicity, we will assume the field is in Montgomery representation.

- Products and squares are not reduced unless stated.
- Sums are not reduced unless stated.
- Subtractions never contain a conditional addition: a fixed multiple of q is always added to the first operand before doing the subtraction.
- If an integer is to be fully reduced, then it is guaranteed to be at least as large as q: $\mathsf{reduce}_q(x) = x - mq$, $m \in \{1, 2, \ldots, j\}$ for some predetermined j.
- For the affine formula, inversion accepts any integer between 1 and $6q-1$, coprime to q, and returns an integer between 1 and $q-1$.
- For the affine formula, Montgomery reduction accepts inputs between 0 and $6q^2$ with $R > 6q$.
- For the projective formula, Montgomery reduction accepts inputs between 0 and $16q^2$ with $R > 16q$.
- Montgomery reductions are allowed to return an output between 0 and $2q-1$.
- The multiples of q used in the formulæ are precomputed.
- For the projective formula, we let the X-, Y- and Z-coordinates be in the range $[0, 2q-1]$.

These countermeasures are not intended to be secure against differential side-channel attacks, for which standard countermeasures, such as blinding, should still be applied.

Applying these recommendations to the affine formula yields the algorithm in Figure 6.7. For the projective formula, we obtain the algorithm in Figure 6.8.

Note that by construction $(x_1+x_2)^2 \geq x_1x_2$, so the pre-reduction results in step 8 of the affine formula and step 14 of the projective formula are positive.

Input:	Points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$.
Output:	Point $(x_3, y_3) = P + Q$.
1.	$E \leftarrow x_1 + y_1 + y_2 + x_2$
2.	if E is odd, then $F \leftarrow E + 3q$, else $F \leftarrow E + 2q$
3.	$G \leftarrow \mathsf{reduce}_q(F/2)$
4.	if $G \neq x_2$, then $\delta \leftarrow 0$, else $\delta \leftarrow 1$
5.	$H \leftarrow F - 2x_{2-\delta}$
6.	$I \leftarrow H^{-1} \mod q$
7.	$J \leftarrow x_1 + x_2$
8.	$K \leftarrow \mathsf{MontRed}(J^2 - x_1 \cdot x_2)$
9.	$L \leftarrow \mathsf{reduce}_q(2q + K + a + y_{1+\delta} - y_{2-\delta})$
10.	$\lambda = \mathsf{MontRed}(L \cdot I)$
11.	$M \leftarrow \mathsf{MontRed}(\lambda^2)$
12.	$x_3 \leftarrow \mathsf{reduce}_q(3q + M - x_1 - x_2)$
13.	$N \leftarrow \mathsf{MontRed}(\lambda \cdot (q + x_1 - x_3))$
14.	$y_3 \leftarrow 2q + N - y_1$

Figure 6.7: Affine coordinates unified point addition formula with side-channel attack countermeasures.

6.3.4 Timing

In this section we report on timings that demonstrate the speed improvement that results from using projective coordinates instead of affine coordinates and show that using the unified point addition formula results in addition and doubling having much closer run times overall.

The timings reported in this section were performed on a 900 MHz UltraSPARC III. The software used the multi-precision integer and elliptic curve libraries from the Netscape Security Services (NSS) library version 3.9 [Moz08] with no optimized assembly code. To obtain high-resolution timings, we used the Solaris `hrtime` C library, which has a resolution of 100 ns. All timings reported are for the 160-bit prime field curve `secp160r2` [Cer00].

In Appendix A.1.4, we provide C code that we used to time various point multiplication formulæ.

Input:	Points $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$.
Output:	Point $(X_3, Y_3, Z_3) = P + Q$.
1.	$\hat{U}_1 \leftarrow X_1 \cdot Z_2$, $\hat{U}_2 \leftarrow X_2 \cdot Z_1$, $\hat{S}_1 \leftarrow Y_1 \cdot Z_2$, $\hat{S}_2 \leftarrow Y_2 \cdot Z_1$
2.	$\hat{T} = \hat{U}_1 + \hat{U}_2$
3.	$M \leftarrow \mathsf{MontRed}(\hat{S}_1 + \hat{S}_2 + \hat{T})$
4.	if M is odd, then $V \leftarrow M + 3q$, else $V \leftarrow M + 2q$
5.	$C \leftarrow \mathsf{reduce}_q(V/2)$
6.	$B \leftarrow \mathsf{reduce}_q(p + \mathsf{MontRed}(\hat{U}_2))$
7.	if $C \neq B$, then $\delta \leftarrow 0$, else $\delta \leftarrow 1$
8.	$E \leftarrow V - \mathsf{MontRed}(2\hat{U}_{2-\delta})$
9.	$Z \leftarrow \mathsf{MontRed}(Z_1 \cdot Z_2)$, $T \leftarrow \mathsf{MontRed}(\hat{T})$
10.	$F \leftarrow \mathsf{MontRed}(Z \cdot E)$, $U_1 \leftarrow \mathsf{MontRed}(\hat{U}_1)$, $U_2 \leftarrow \mathsf{MontRed}(\hat{U}_2)$
11.	$L \leftarrow \mathsf{MontRed}(F \cdot E)$
12.	$G \leftarrow \mathsf{MontRed}(L \cdot T)$
13.	$K \leftarrow 2q + \mathsf{MontRed}(a \cdot Z + \hat{S}_{1+\delta}) - \mathsf{MontRed}(\hat{S}_{2-\delta})$
14.	$R \leftarrow \mathsf{MontRed}(Z \cdot K + T^2 - U_1 \cdot U_2)$
15.	$H \leftarrow \mathsf{MontRed}(R^2)$
16.	$W \leftarrow \mathsf{reduce}_q(3q + H - G)$
17.	$X_3 \leftarrow \mathsf{MontRed}((2F) \cdot W)$
18.	$J \leftarrow \mathsf{MontRed}(F^2)$
19.	$Z_3 \leftarrow \mathsf{MontRed}((2F) \cdot J)$
20.	$N \leftarrow \mathsf{MontRed}(F \cdot M)$
21.	$Y_3 \leftarrow 2q - \mathsf{MontRed}(R \cdot (2q + 2W - G) + L \cdot N)$

Figure 6.8: Projective coordinates unified point addition formula with side-channel attack countermeasures.

First we looked at the difference in time between a modular subtraction with a conditional addition and without. The average time of a 160-bit prime field modular subtraction $a - b \mod q$ when $a > b$ was about 320 ns on our test system. By contrast, when $a < b$, and hence when a conditional addition is required, the average time was about 550 ns. This is a non-negligible difference which could potentially be detected in a side-channel analysis. Besides timing information, of course, other side-channels such as power analysis could leak information as well about this operation.

Next, we timed the point addition, doubling, and multiplication operations for various sets of point formulæ; these are reported in Figure 6.9. For the affine and unified formulæ, point multiplication uses the double-and-add technique. The timings in the table are the average of 10^5 operations.

Because the affine and projective formula of Brier, Déchène, and Joye in Figure 6.9 show some timing difference on average between the addition and doubling operations, we further analyzed these operations.

Formula	Addition	Doubling	Multiplication
BDJ affine	126.5 μs	126.2 μs	29.03 ms
Affine	115.7 μs	118.4 μs	27.89 ms
BDJ projective	58.9 μs	58.5 μs	13.99 ms
BJ projective	49.8 μs	49.5 μs	11.76 ms
Jacobian projective			7.95 ms
Modified Jacobian wNAF, $w = 5$			6.22 ms

Figure 6.9: Average point operation timings for `secp160r2` curve.

Figure 6.10 shows average timings and standard deviations for point additions and point doublings over the course of a single point multiplication. The results were obtained by recording the time of each addition or doubling in a single point multiplication using the double-and-add algorithm.

Formula	Operation	Average	Standard Deviation
BDJ affine	unified addition	126.528 μs	4.094 μs $\approx 3.2\%$
	unified doubling	126.155 μs	3.700 μs $\approx 2.9\%$
	difference	0.373 μs $\approx 0.3\%$	
BDJ projective	unified addition	58.992 μs	0.474 μs $\approx 0.8\%$
	unified doubling	59.307 μs	0.448 μs $\approx 0.75\%$
	difference	0.315 μs $\approx 0.53\%$	

Figure 6.10: Point operation timings from a single point multiplication for `secp160r2` curve.

The top half of Figure 6.10 reports timings for point addition and doubling using the affine formula of equation (6.7). Unified point doubling takes slightly less time than unified point addition on average, but difference between the two operations (0.3%) is one-tenth the size of the standard deviation of either operation, suggesting that the timings of the two operations cannot be reliably distinguished.

The bottom half of Figure 6.10 reports timings for point addition and doubling using the projective formula of equation (6.8). Here, unified point doubling takes slightly more time (0.53%) than unified point addition, but the difference is still less than the standard deviation of either of these operations (around 0.8%).

For both the affine and projective cases of these formulæ in Figure 6.10, the average difference in timing between a point addition and point doubling is small compared to the standard deviation. However, when combined with other side-channel information, this may still be useful to an attacker. Using the countermeasures we described in Section 6.3.3.3 may further eliminate these differences.

6.4 Unified point addition formulæ for binary fields

In this section we note the relevance of the work in this chapter to elliptic curves over binary fields. While unified projective formulæ can be derived from the unified affine formulæ, there is no speed improvement since binary field inversion is not as computationally expensive, comparatively, for binary fields as it is for prime fields. Additionally, because binary field subtraction does not have a conditional addition, our conditional modular reduction attack does not apply in the binary case.

Unified formula of Brier and Joye. The unified form of λ for point addition and doubling for curves over binary fields is:

$$\lambda = \frac{(x_1+x_2)^2 + x_1x_2 + a(x_1+x_2) + y_1}{y_1+y_2+x_2}, \quad \text{if } y_1+y_2+x_2 \neq 0 \ . \tag{6.30}$$

Point addition or doubling using the affine form requires 1 inversion, 4 multiplications, and 2 squarings; using a projective form requires 20 multiplications and 3 squarings. In practice, a binary field inversion has the same computational cost as about 10 field multiplications [HHM00], so the affine form is faster than the projective form.

Unified affine formulæ of Brier, Déchène, and Joye. Let

$$\lambda_m = \begin{cases} \frac{(x_1+x_2)^2+x_1x_2+y_2+a(x_1+x_2)+(y_1+y_2)m}{y_1+y_2+x_2+(x_1+x_2)m}, & \text{if } y_1+y_2+x_2+(x_1+x_2)m \neq 0 \\ \frac{(x_1+x_2)^2+x_1x_2+y_1+a(x_1+x_2)+(y_1+y_2)\tilde{m}}{y_1+y_2+x_2+(x_1+x_2)\tilde{m}}, & \text{if } y_1+y_2+x_1+(x_1+x_2)\tilde{m} \neq 0 \end{cases}, \tag{6.31}$$

for any polynomial $m = m(x_1, y_1; x_2, y_2)$, with $\tilde{m}(x_1, y_2; x_2, y_2) = m(x_2, y_2; x_1, y_1)$. These formulæ are defined for all points except those which satisfy

$$y_1+y_2+x_2+(x_1+x_2)m = 0 = y_1+y_2+x_1+(x_1+x_2)\tilde{m} \ . \tag{6.32}$$

For efficiency purposes, we can choose $m = m_0 = 0$. In this case, we get the following unified formula for λ:

$$\lambda = \lambda_0 = \begin{cases} \frac{(x_1+x_2)^2+x_1x_2+a(x_1+x_2)+y_2}{x_1+y_1+y_2}, & \text{if } x_1+y_1+y_2 \neq 0 \\ \frac{(x_1+x_2)^2+x_1x_2+a(x_1+x_2)+y_1}{x_2+y_1+y_2}, & \text{if } x_2+y_1+y_2 \neq 0 \end{cases} . \tag{6.33}$$

Unified point addition using $\lambda = \lambda_0$ requires 4 field multiplications, 2 field squarings, and 1 field inversion.

Unified projective formulæ of Brier, Déchène, and Joye. We now obtain a projective form of the unified point addition formula given by λ as defined in equation (6.33). Letting $x_i = X_i/Z_i$, $y_i = Y_i/Z_i$ and completing the square in the

numerator of λ, we obtain:

$$X_3 = FW \ , \quad Y_3 = R(LU + W) + X_3 + HES \ , \quad Z_3 = FH \ , \tag{6.34}$$

where $U_1 = X_1Z_2, U_2 = X_2Z_1, S_1 = Y_1Z_2, S_2 = Y_2Z_1, Z = Z_1Z_2, T = U_1 + U_2, M = S_1 + S_2, E = M + U_{1+\delta}, F = ZE, L = FE, G = LT, H = F^2, R = T^2 + U_1U_2 + Z(aT + S_{2-\delta}), K = FR + G + aH$, and $W = R^2 + K$. Note that $\delta = 0$ when $M \neq U_1$ and $\delta = 1$ otherwise. These terms were derived using $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$ but by symmetry of point addition could have equally been derived using $y_3 = \lambda(x_2 + x_3) + x_3 + y_2$.

This projective formula requires 19 field multiplications and 3 field squarings. In practice, a binary field inversion has the same computational cost as about 10 field multiplications [HHM00] so the affine form of the unified point addition formulæ is faster than the projective form.

In Appendix A.1.5.1, we provide Maple code that checks that the projective formula of equation (6.34) corresponds to the affine formula of equation (6.33).

Conditional modular reduction attack. For the projective formula for unified point addition for curves over binary fields as given in equation (6.34), Walter's attack does not apply nor does our extension in Section 6.3.3.2. Walter's original attack does not apply because field multiplication in binary fields is not implemented using Montgomery multiplication. Our extension does not apply because no modular reduction operations are necessary in field addition.

Part II

Quantum Cryptography

Chapter 7

The Case for Quantum Key Distribution

Contents

7.1 Introduction

Since its discovery, the field of quantum cryptography — and in particular, quantum key distribution (QKD) — has garnered widespread technical and popular interest. The promise of "unconditional security" has brought public interest, but the often unbridled optimism expressed for this field has also spawned criticism and analysis [Sch03, PPS04, Sch07, Sch08].

QKD is a new tool in the cryptographer's toolbox: it allows for secure key agreement where the output key is entirely independent from any input value, a task that is impossible using classical[1] cryptography. QKD does not eliminate the

[1] All computation must be viewed as taking place in a physical system described by particular laws of nature. By *classical cryptography*, we mean cryptography taking place in a computational system described by classical (Newtonian) Turing machines with classical communication.

need for other cryptographic primitives, such as authentication, but it can be used to build systems with new security properties.

Through the rest of this chapter, we restrict our discussion on quantum cryptography to quantum key distribution (QKD). Many other quantum cryptographic primitives exist — quantum private channels, quantum public key encryption, quantum coin tossing, blind quantum computation, quantum money — but almost all require a medium- to large- scale quantum computer for implementation. QKD, on the other hand, has already been implemented by many different groups, has seen attempts at commercialization, and thus its potential role in upcoming security infrastructures merits serious examination.

There are three phases (which are sometimes intertwined) to establishing secure communications:

1. *Key agreement:* Two parties to agree upon a secure, shared private key.
2. *Authentication:* Allows a party to be certain that a message comes from a particular party. In order for key agreement to avoid man-in-the-middle attacks, authentication of some form must be used.
3. *Key usage:* Once a secure key is established, it can be used for encryption (using a one-time pad or some other cipher), further authentication, or other cryptographic purposes.

QKD is just one part of this overall information security infrastructure: two parties can agree upon a private key, the security of which depends on no computational assumptions, and which is entirely independent of any input to the protocol.

If we live in a world where we can reasonably expect public key authentication to be secure in the short- to medium-term, then the combination of public key cryptography for authentication and QKD for key agreement can lead to very strong long-term security with all the convenience and benefits we have come to expect from distributed authentication in a public key infrastructure.

If we live in a world where public key authentication can no longer be employed safely, we must revert to shared secret key authentication or trusted third party authentication before we can use QKD. Here QKD still offers a benefit over an entirely classical solution because the key agreed upon by QKD is independent of the authentication keys, eliminating the ability of trusted third parties to later compromise information protected by QKD.

If we live in a world where there exist public key cryptography schemes that are believed to be secure indefinitely, then there is a reduced case for QKD, but it is still of interest for a variety of reasons. For example, if side-channel attacks become increasingly easier to mount against classical public key cryptography, then self-testable, entanglement-based QKD systems may be appealing. Other forms of quantum cryptography may also be of interest, especially for the secure communication of quantum information if quantum computing becomes widespread.

Experimental research on quantum key distribution continues to improve the usability, security, rate, and distance of QKD systems. As public key cryptography systems are retooled with new algorithms and standards over the coming years, there is an opportunity to incorporate QKD as a new tool offering fundamentally new security features.

Related work. This work is motivated as a response to other opinions about the role of QKD, especially the thoughtful note "Why quantum cryptography?" [PPS04]. A response by the SECOQC project [ABB+07] addresses related concerns as well, with special attention paid to the networks of QKD links. Our response emphasizes the role of authentication in QKD.

Outline. In the rest of this chapter, we argue that QKD has a valuable role to play in future security infrastructures. In Section 7.2, we give an overview of how QKD works, and give an example where its high security is needed in Section 7.3. We describe the conditions for the security of QKD in Section 7.4. We then discuss the other parts of the communication infrastructure: encryption in Section 7.5 and authentication in Section 7.6. In Section 7.7, we discuss some limitations to QKD as it stands and how they may be overcome, with special consideration to networks of QKD devices in Section 7.8. We offer a concluding statement in Section 7.9.

7.2 A brief introduction to QKD

In this section we provide a very brief overview of quantum key distribution. More detailed explanations are available from a variety of sources [NC00, ABB+07, SBPC+08].

In QKD, two parties, Alice and Bob, obtain some quantum states and measure them. They communicate (all communication from this point onwards is classical) to determine which of their measurement results could lead to secret key bits; some are discarded in a process called sifting because the measurement settings were incompatible. They perform error correction and then estimate a security parameter which describes how much information an eavesdropper might have about their key data. If this amount is above a certain threshold, then they abort as they cannot guarantee any secrecy whatsoever. If it is below the threshold, then they can apply privacy amplification to squeeze out any remaining information the eavesdropper might have, and arrive at a shared secret key. Some of this classical communication must be authenticated to avoid man-in-the-middle attacks. Some portions of the protocol can fail with negligible probability.

A flow chart describing the stages of quantum key distribution is given in Figure 7.1.

Once a secret key has been established by QKD, it can be used for a variety of purposes. The most common approach is to use it as the secret key in a one-time

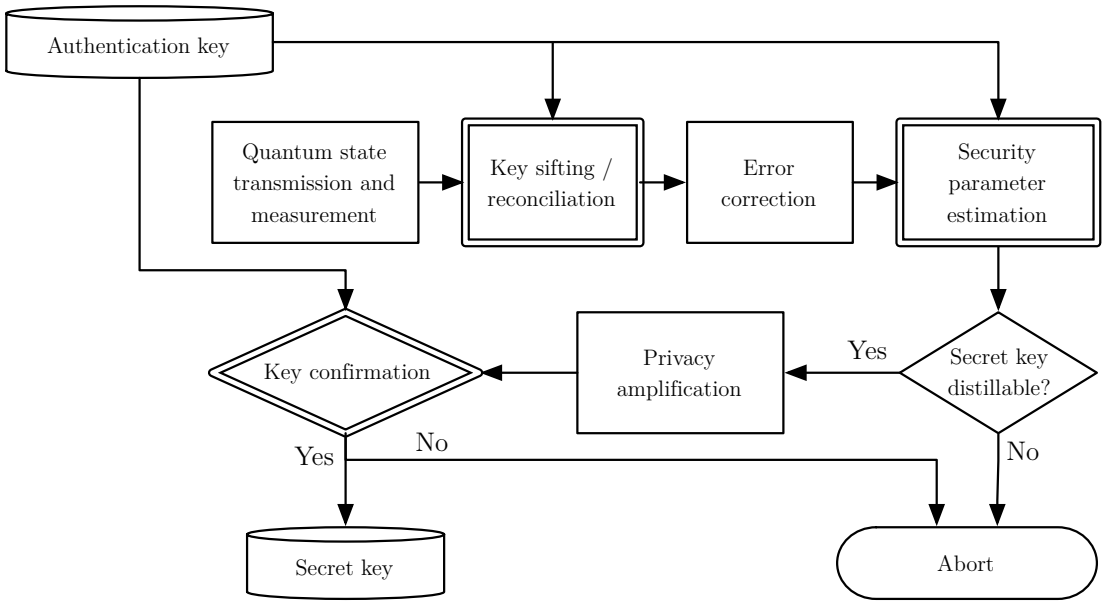

Figure 7.1: Flow chart of the stages of a quantum key distribution protocol. Stages with double lines require classical authentication.

pad to achieve unconditionally secure encryption. The key can also be used for classical authentication in subsequent rounds of QKD.

We can expect that as QKD research continues, QKD devices will become more robust, easier to configure, less expensive, and smaller, perhaps sufficiently miniaturized to fit on a single circuit board.

7.3 Who needs quantum key distribution?

It is widely understood that "security is a chain; it's as strong as the weakest link" [Sch03], and cryptography, even public key cryptography, is indeed one of the strongest links in the chain. We cannot trust that a particular computationally secure cryptographic scheme and parameter size will remain secure indefinitely, and many expert recommendations are unwilling to provide guidance for much more than 30 years in the future. While much of the information being encrypted today does not need 30 years of security, some does.

Moreover, it is important to plan well in advance for changes in security technology. Suppose, for example, that a particular application using RSA or elliptic curve cryptography (ECC) needs information to be secure for x years, and it takes y years to retool the infrastructure to a new cryptosystem. If large-scale quantum computers capable of breaking RSA or ECC are built within z years, with $z < x+y$, then we are already too late: we need to start planning to use new cryptosystems long before old ones are broken.

Government, military, and intelligence agencies need long-term security. For example, the UK government did not declassify the 1945 report on its efforts in breaking the Tunny cipher during World War II until 2000 [GMT45], and the US government's current classification regime keeps documents classified for up to 25 years [Bus03, §1.5(b)].

Businesses trying to protect long-term strategic trade secrets may also wish for long-term confidentiality. Situations with long-term deployments but well-specified communication requirements could also benefit from QKD: it is inconvenient and expensive to have to upgrade the 1.5 million automated teller machines (ATMs) worldwide whenever the latest cryptographic protocol is broken or deemed obsolete, but QKD could provide standards less likely to change due to cryptanalysis.

One particular industry likely to require long-term, future-proof security is health care. Health care systems are slowly but irreversibly becoming more electronic, and health care records need privacy for 100 years or more. Securing the storage of these records in data centers is essential, of course, and quantum key distribution does not aim to solve this difficult problem. Equally important, however, is the secure communication of health care records, which can be protected by the information-theoretic security offered by quantum key distribution.

Quantum key distribution is also not the only way to establish information theoretically secure keys. The physical transfer of long, randomly generated keys is also an information theoretically secure key distribution scheme. With hard drive prices approaching US $0.10 per gigabyte, one should not underestimate "the bandwidth of a truck filled with hard drives" (although increases in fuel prices may counteract the cost efficiency of such a communication system). This approach is not appropriate for all scenarios. In some cases, it may be impossible to rekey a system in this manner (for example, satellites and space probes). It requires assurances that the physical keys were transported securely. It also requires secure storage of large amounts of key until use. QKD requires only a small amount of key, the authentication key, to be securely stored until use, and then can generate fresh encryption keys on demand that need only be stored for the short time period between key generation and message encryption/decryption.

7.4 The security of QKD

Quantum key distribution is often described by its proponents as "unconditionally secure" to emphasize its difference with computationally secure classical cryptographic protocols. While there are still conditions that need to be satisfied for quantum key distribution to be secure, the phrase "unconditionally secure" is justified because, not only are the conditions reduced, they are in some sense minimal necessary conditions. Any secure key agreement protocol must make a few minimal assumptions, for security cannot come from nothing: we must be able to identify and authenticate the communicating parties, we must be able to have some private

location to perform local operations, and all parties must operate within the laws of physics.

The following statement describes the security of quantum key distribution, and there are many formal mathematical arguments for the security of QKD (for example, [May97, LC99, GLLP04]).

Theorem 7.1 (Security statement for quantum key distribution) *If*

A1) quantum mechanics is correct, and
A2) authentication is secure, and
A3) our devices are reasonably secure,

then with high probability the key established by quantum key distribution is a random secret key independent (up to a negligible difference) of input values.

Assumption 1: Quantum mechanics is correct. This assumption requires that any eavesdropper be bounded by the laws of quantum mechanics, although within this realm there are no further restrictions beyond the eavesdropper's inability to access the devices. In particular, we allow the eavesdropper to have arbitrarily large quantum computing technology, far more powerful than the current state of the art. Quantum mechanics has been tested experimentally for nearly a century, to very high precision. But even if quantum mechanics is superseded by a new physical theory, it is not necessarily true that quantum key distribution would be insecure: for example, secure key distribution can be achieved in a manner similar to QKD solely based on the assumption that no faster-than-light communication is possible [BHK05].

Assumption 2: Authentication is secure. This assumption is one of the main concerns of those evaluating quantum key distribution. In order to be protected against man-in-the-middle attacks, much of the classical communication in QKD must be authenticated. Authentication can be achieved with unconditional security using short shared keys, or with computational security using public key cryptography. We discuss the issue of authentication in greater detail in Section 7.6.

Assumption 3: Our devices are secure. Constructing a QKD implementation that is verifiably secure is a substantial engineering challenge that researchers are still working on. Although the first prototype QKD system leaked key information over a side channel (it made different noises depending on the photon polarization, and thus the "prototype was unconditionally secure against any eavesdropper who happened to be deaf" [Bra06]), experimental cryptanalysis leads to better theoretical and practical security. More sophisticated side-channel attacks continue to be proposed against particular implementations of existing systems (for example, [ZFQ+08]), but so too are better theoretical methods being proposed, such as the

decoy state method [Hwa03]. Device-independent security proofs [MY98, ABG+07] aim to minimize the security assumptions on physical devices. It seems reasonable to expect that further theoretical and engineering advances will eventually bring us devices which have strong arguments and few assumptions for their security.

7.5 Key usage: encryption

The most commonly discussed usage for the key generated by quantum key distribution is encryption. There are two ways [PPS04] this key can be used for encryption.

In an *unconditionally secure system*, the private key from QKD is used as the key in a one-time pad. Since the key is information theoretically secure, so too is the encryption of the message: no computer, quantum or classical, will ever be able to decipher the encrypted message. There are challenges to this system, however. First, the one-time pad keys must be carefully stored and managed, as the double-use of one-time keys can seriously compromise security. Second, as we discuss in Section 7.7, physical QKD systems cannot yet achieve sufficiently high key generation rates to be able to encrypt large messages with one-time pads in real time.

To deal with this second challenge of low QKD key rates, *hybrid systems* have been proposed, where the key from QKD is expanded with a classical stream cipher or block cipher such as the Advanced Encryption Standard (AES) to encrypt long messages. In this setting, the security of the encrypted messages is no longer information theoretic: it depends on the computational assumption that the cipher used is hard to break. While this is not ideal, it may not be too risky either. Historically, cryptographers have been very good at designing block ciphers with few weaknesses: for example, the Data Encryption Standard (DES), designed in the 1970s, is no longer considered secure due to its short key length, but DES has stood up well to over 30 years of cryptanalytic attacks. Under a known plaintext attack, the security of DES is reduced from 2^{56} to about 2^{41}, but, when rekeying is sufficiently frequent, the effect of known plaintext attacks is limited [ABB+07, §3.2]. Moreover, quantum computers do not seem to have too much impact on ciphers: while Grover's search algorithm implies that the key length needs to be doubled, the exponentially faster attacks promised by Shor's algorithm and others do not apply to most ciphers. However, when using QKD to generate short keys, care must be taken due to finite length effects (*c.f.* [CS08]).

Even when used in hybrid systems, QKD offers a substantial advantage over classical key agreement: the key from QKD is independent of any inputs to the key agreement protocol. Thus, QKD reduces the number of points of attack: once a key has been established, the only way to attack such a system is to cryptanalyze the encryption. By contrast, a system using classical key agreement could be attacked by trying to take the inputs to the classical key agreement protocol and

determining the generated private key (for example, by solving the Diffie-Hellman problem). QKD also provides an advantage compared to systems that are rekeyed using symmetric keying techniques, such as key transport using symmetric encryption or session key derivation from a master key: since QKD keys are independent between blocks, QKD has forward secrecy and the compromise of previous keys does not affect the confidentiality of future blocks, whereas the compromise of a master key in symmetric rekeying can be catastrophic.

Hybrid QKD systems offer enhanced security compared to ciphers used without QKD: the QKD subsystem provides fresh, independent keying material frequently, which can rekey the classical block or stream cipher; with frequent rekeying, we reduce the risk of attacks against the underlying cipher that make use of many plaintexts or ciphertexts encrypted under the same key and achieve greater forward secrecy in the face of key compromise.

7.6 Authentication

Quantum key distribution does not remove the need for authentication: indeed, authentication is *essential* to the security of QKD, for otherwise it is easy to perform a man-in-the-middle attack. There are two main ways to achieve authentication: public key authentication and symmetric key authentication. *Symmetric key authentication* can provide unconditionally secure authentication, but at the cost of needing to have pre-established pairs of symmetric keys. *Public key authentication*, on the other hand, is simpler to deploy, and provides extremely convenient distributed trust when combined with certificate authorities (CAs) in a public key infrastructure (PKI). Public key authentication cannot itself be achieved with information theoretic security. We argue, however, that the security situation is more subtle than this: the use of public key authentication can still lead to systems that have very strong long-term security.

A third method for authentication is to use trusted third parties which actively mediate authentication between two unauthenticated parties, but there has been little interest in adopting these in practice. Certificate authorities, which are used in public key authentication, are similar to trusted third party authentication but do not actively mediate the authentication: they distribute signed public keys in advance but then do not participate in the actual key authentication protocol. The difference in trust between trusted third parties and certificate authorities for authentication in QKD is smaller than in the purely classical case since the key from QKD is independent of the inputs.

7.6.1 Symmetric key authentication

Parties who already share a short private key can use an unconditionally secure message authentication code to authenticate their messages. The first such approach was described by Wegman and Carter [WC81] and has been refined for use

in QKD (for example, [HLM03, PNM+05]). It is for this reason that quantum key distribution is sometimes called *quantum key expansion*: it can take a short shared key and expand it to an information-theoretically secure long shared key.

Interestingly, the universal composability of quantum key distribution implies that we can use some of the key generated by QKD to authenticate the messages in the next round of QKD with a negligible decrease in security. Thus we can continue QKD (more or less) indefinitely having started only with a relatively short (on the order of a few kilobytes) authentication key.

7.6.2 Public key authentication

While symmetric key authentication promises unconditionally secure authentication, it is difficult to deploy because each pair of communicating parties must share a private key. Public key infrastructures allow for distributed trust and have been essential to the success of electronic commerce. While many advocates of quantum cryptography dismiss the role of computationally secure public key authentication in QKD, we argue that public key authentication will be vital in a quantum key distribution infrastructure and can still provide meaningful security statements.

Public key authentication schemes, being computationally secure, tend to be broken, and invariably sooner than we expect. In 1977, Rivest speculated [Gar77] that it could take 40 quadrillion years to solve the RSA-129 problem (factoring a 129-decimial-digital RSA modulus), but it was broken only 17 years later [AGLL96]. While the popular press still occasionally uses expressions such as "more than a quadrillion years" [Lys08] to describe the security of number-theoretic schemes, technical recommendations [NIS07, BCC+08] are more nuanced and tend not to speculate too far beyond 2030. Notably, these recommendations tend to "assume [...] (large) quantum computers do not become a reality in the near future" [BCC+08, p. 25].

Large scale quantum computers are widely believed to be some time off, but there appears to be no reason at present to doubt their eventual efficacy. Quantum computers, however, are not the only threat against public key authentication. Computers do become faster and new algorithms do help speed cryptanalysis. However, we are not so pessimistic to think that all public key authentication is doomed forever. In fact, we believe that public key authentication will continue to play a vital role in communication security indefinitely, even in the presence of quantum computing.

Although today's popular public key schemes — RSA, finite field discrete logarithm, and elliptic curve — would be broken by a large scale quantum computer, other "post-quantum" schemes do not immediately fall to quantum algorithms, and other schemes are sure to be developed (*cf.* [BBD09]). It seems to us, then, that public key schemes in the future are likely to go through a lifecycle in which a new primitive is proposed, it appears secure against current attack techniques, reasonable parameter sizes are proposed, adopted, and then computing technology and

cryptanalysis advances chip away at the security until a newer scheme provides better tradeoffs. It is not too hard to imagine a 20-year window in which a public key scheme, along with a particular set of parameter sizes, is considered viable.

It is in this scenario, where a particular public key authentication scheme is only deemed to be secure for a 20-year period, that quantum key distribution can thrive. A public key authentication infrastructure provides the large scale usability that we have come to expect from PKIs, and when combined with quantum key distribution can offer strong security promises. In quantum key distribution, the authentication — in the form of public key authentication — only needs to be secure up to and including the initial connection. Once the QKD protocol has output some secret key, a portion of this secret can subsequently be used for symmetric key authentication. In fact, even if the original authentication keys are revealed after the first QKD exchange, the key from QKD remains information theoretically secure. In other words, we have the following statement:

> If authentication is unbroken during the first round of QKD, *even if it is only computationally secure*, subsequent rounds of QKD will be information-theoretically secure.

By contrast, classical public key exchange schemes do not have this feature. Although one can employ a protocol in which a new key is transmitted encrypted under the old key, an eavesdropper who logs all communications and subsequently breaks the first key can read all future communications. With QKD, new session keys are completely independent of all prior keys and messages.

7.7 Limitations

Two undeniable limitations of present quantum key distribution schemes are distance and key rate. Because of the fragile nature of the quantum mechanical state that is transmitted during quantum key distribution, the longer the distance that the photons have to travel, the more photons that are lost to decoherence and noise and hence the lower the rate of secret key formation. Distance and key rate are a tradeoff, but progress is being made on improving the overall tradeoff.

Distance. The longest QKD experiments to date have acheived secure key generation over a 184.6km fiber optic link [HRP+06] and over a free-space link spanning a distance of 144km at a rate of 12.8 bits/second[SMWF+07] . This free-space distance is considered sufficient to communicate between any two points on the surface of the Earth via orbiting satellites, the feasibility of which is the subject of a proposed experiment [PAFdM+08].

Quantum repeaters [BDCZ98] would also overcome the distance limitation, allowing shared quantum states to be established between distant parties. While

these systems are not yet operational, they are easier to implement than full-scale quantum computers; theoretical and experimental work progresses on their development.

Key rate. While long distance experiments achieve very low key rates on the order of a few bits per second, shorter distance experiments have demonstrated very high key rates. NIST has achieved key rates of over 4 MB per second over 1km of fibre [Nat06] and 500 kb per second at 10km [XMM+07]. These key rates are an impressive accomplishment but still come short of the rates achieved in classical communication over long distances.

When a QKD key is used for encryption, current key rates may not be sufficient for a one-time pad and hybrid schemes need to be used, in which the QKD key is used as the private key in a symmetric encryption algorithm such as the block cipher AES. However, as we have argued in Section 7.5, even hybrid QKD systems offer enhanced security compared to classical key agreement since the keys generated by QKD are independent of any inputs to the key agreement procedure and since many symmetric encryption algorithms are resistant to known attacks by quantum computers. Key rate can always be negatively impacted by an adversary disturbing the quantum channel, but such an adversary can not impact the security of the key agreement.

7.8 QKD Networks

As QKD technology progresses, the structure of deployed QKD systems will progress in four stages to reduce distance limitations and increase commercial applicability:

1. *Point-to-point links:* Two QKD devices are directly connected over a relatively short distance.

2. *Networks with optical switches:* Multiple QKD devices are arranged in a network with optical switches to allow different pairs of interaction. Optical switches, however, do not increase communication distance. The switches need not be trusted. One example of such a network is the DARPA quantum network [ECP+05].

3. *Networks with trusted relays:* Multiple QKD devices are arranged in a network. Intermediate nodes in the network can act as classical relays which relay information between distant nodes. The relay nodes need to be trusted, although trust can be reduced by having the sender use a secret sharing scheme [BS08]. This type of QKD network would be suitable for scenarios where the operator of the network is also the user of the network, for example, a bank creating a network among its many branches, each of which is individually trusted. One example of such a network is the SECOQC quantum network [ABB+07].

4. *Fully quantum repeater network:* Multiple QKD devices are arranged in a network with quantum repeaters [BDCZ98]. Although individual links are still distance-limited, the quantum repeater nodes allow entanglement to be linked across longer distances, so QKD can be performed between distant parties. The quantum repeaters need not be trusted, and this type of QKD network corresponds to the service provider scenario.

7.9 Concluding remaks

Quantum key distribution makes use of the eavesdropper-detection power offered by quantum mechanics to establish a shared key that is verifiably secure and independent of any other data, provided the communicating parties share an authentic channel. The security of the system depends on no computational assumptions and thus has the potential to offer security against present or future attackers with unbounded classical or quantum computational power.

There are many scenarios, such as government, military, and health care, in which information needs to remain secure for 25, 50, or even 100 years. Using QKD reduces the assumptions about the cryptographic system and produces a shared secret key that, by the properties of quantum mechanics, is independent of any other data, including the input.

It is important to consider how QKD fits into the larger cryptographic infrastructure. When used with public key authentication, QKD provides strong security with the convenience of distributed authentication using public key infrastructures; the public key authentication scheme need only be secure up until QKD occurs, but the key from QKD will remain secure indefinitely. If public key authentication is not possible, shared secret authentication can still be used to give enhanced security compared to classical key expansion.

The present limitations of QKD — distance and key rate — will be further mitigated as experimental research in QKD continues, and quantum repeaters promise fully quantum long distance networks.

We believe that, as the technology continues to improve, QKD will be an increasingly valuable tool in the cryptographer's toolbox for building secure communication systems.

Chapter 8

Quantum Money

Contents

8.1 Introduction

The uncertainty principle and no-cloning theorem of quantum mechanics made quantum money one of the original interests of quantum information theory. The ability to create digital money which cannot be counterfeited because of the laws of physics is a compelling idea. Classical digital cash has been researched extensively, with ongoing improvements to its security tradeoffs, but remains fundamentally subject to the constraint that classical bits can be easily copied. With quantum money, we hope to use the inability to perfectly clone quantum states to prevent counterfeiting. Besides being non-counterfeitable, an effective digital cash scheme should also be efficiently verifiable, anonymous, transferable, and robust.

In this chapter, we describe a new form of quantum money called *quantum coins*, where all coins of the same denomination are represented by identical quantum states. We state formally what it means for them to be unforgeable and describe how to implement quantum coin schemes using black box operations and using blind quantum computing. We also describe *quantum bills* which capture a wide range of notions of quantum money.

Related work. Digital cash has been well-explored in classical cryptographic contexts, with the first schemes being proposed by Chaum [Cha85, Cha88] and Chaum, Fiat, and Naor [CFN88]. For classical digital cash schemes, one of the main problems to solve is the **multiple-spending problem**: since classical digital cash can easily be duplicated, there must be a way to prevent the same tokens from being redeemed more than once. An online scheme, in which each token is verified with the bank at the time it is meant to be spent, solves this problem immediately, but online verification requires an online communications channel between merchant and bank. The other general solution for preventing multiple spending is to embed some identity information in the money tokens such that, if the token is spent only once, the transaction remains anonymous, but if the token is spent multiple times, then the bank can combine these multiple transactions to recover the identity of the multiple spender. Moreover, classical digital cash is not transferable unless we allow the size of the token to grow linearly in the number of transfers [CP92].

Quantum money was one of the earliest applications of quantum information theory, and was introduced in the early papers of Wiesner [Wie83] and Bennett, Brassard, Breidbard, and Wiesner [BBBW82]. In both schemes, a bank constructs distinct quantum tokens and corresponding classical serial numbers. The tokens are the encoding of a random string in randomly chosen basis states of two non-orthogonal bases; the no-cloning theorem prevents perfect cloning of individual tokens. However, the tokens can only be verified by the bank: verification requires knowledge of the bases chosen for each token and the classical string that should be obtained upon measurement in the appropriate bases. This means that an online quantum channel is required between merchants and the bank. The tokens are non-transferable and are not anonymous.

Tokunaga, Okamoto, and Imoto [TOI03] give a scheme for non-transferable anonymous quantum cash with online verification. In their scheme, a user obtains a distinct token from the bank; tokens are generated using private parameters and random values stored by the bank. The user then alters the token with an appropriate randomly chosen unitary transformation to obtain anonymity. At payment time, the user presents the token to the merchant who transmits it (over a quantum channel) to the bank for verification. The scheme is secure against an attacker who can examine a single token, but has not been proven secure against an attacker who can obtain and examine all the quantum tokens.

Our work on quantum coins makes use of work by Aaronson [Aar05] that introduced a complexity-theoretic no-cloning theorem that allows us to argue for the unforgeability of quantum coins. We presented the basics of our quantum coin scheme at the Canadian Quantum Information Students' Conference in 2006 [MS06] and the Quantum Information Processing (QIP) conference in 2007 [MS07]. Subsequently Aarsonson expanded his work based on discussions with us to also include a presentation of quantum money [Aar09] similar to ours; we have noted in footnotes throughout this chapter where that he presents similar concepts.

Contributions. In this chapter, we present a new type of quantum money, which we call **quantum coins**: coins are transferable, locally verifiable, and unforgeable, and have some anonymity properties. Each coin generated by the bank should be a copy of the same quantum state, and hence coins should be indistinguishable from one another. Additionally, a circuit is provided to allow the coins to be verified locally and then transferred for later use.

We describe how to achieve quantum coins with black box quantum circuits and with blind quantum computation. The unforgeability of coins in our scheme comes from complexity theoretic assumptions on the adversary's running time.

Our work contrasts with previous quantum money schemes, which we call **quantum bills**: in a quantum bill scheme, the bank generates tokens that are classical/quantum pairs, which in general are distinct. The classical string may serve as a serial number or as some input value to be used in the verification procedure.

Future directions. Our quantum coin construction of Section 8.4 requires the use of a black-box oracle in the verification circuit, but it is not yet known how these can be implemented. An open question is to find a way to obfuscate the verification circuit so that it is effectively a black box, and in general to find a model for obfuscation of quantum circuits, possibly using computational assumptions. We provide an alternative mechanism for verifying quantum coins using blind quantum computation, at the expense of requiring quantum communication (or shared entanglement with classical communication). Reducing the communication and computational requirements of blind quantum computing is a problem that merits further study. Although our coins are inherently anonymous if the bank issues coins correctly, we do not yet have a mechanism to allow users of the system to verify that the coins are indeed issued correctly, so this remains an open question.

In Section 8.3.2, we briefly discuss a model for quantum bills. An open question related to quantum bills is to find an offline-verifiable quantum bill scheme; this may require using computational hardness assumptions.

Outline. The remainder of the chapter is organized as follows. In Section 8.2, we describe the goals for a quantum money scheme and analyze existing quantum money schemes, as well as our own, in relation to these goals. Section 8.3 introduces the two main types of quantum money, quantum coins and quantum bills, and describes their precise security properties. In Section 8.4, we describe how to implement quantum coins in the black box model and give bounds on unforgeability. In Section 8.5, we describe how to implement quantum coins using blind quantum computation.

8.2 Security goals

We now describe, informally, the properties that a good money scheme should have.

G1. *Anonymous*: it should be difficult for any party to trace the use of a token to determine who spent it or where they spent it.

G2. *Unforgeable*: given zero or more tokens and the verification circuit, it should be difficult for a forger to produce another token that passes the verification procedure with non-negligible probability.

G3. *Efficiently locally verifiable*: there should be an efficient algorithm that can determine with high accuracy whether a token is valid or not, without communicating with the bank.

G4. *Transferable*: a valid token should be unchanged by the verification procedure, and thus can be transferred and reused in a subsequent verification procedure.

We will formally define unforgeability for quantum coin schemes in Section 8.3.1.2.

Figure 8.1 shows which of the above goals are satisfied by various existing money schemes. The "type" column indicates whether the tokens for a given denomination are all identical ("coin") or different ("bill"). For classical digital cash schemes, we note that while unforgeability is impossible, it is possible to detect double spending of a token and trace it back to the offending party; such schemes, however, offer anonymity and offline double-spending detection only with computational assumptions. Our quantum coin schemes offer "partial" anonymity as we describe in Section 8.3.1.3. Additionally, the size of transferable digital cash must grow linearly in the number of transfers [CP92].

8.3 Types of quantum money

8.3.1 Quantum coins

In one type of quantum money, *quantum coins*, a bank issues many tokens for a particular denomination, and all these tokens are (supposed to be) copies of the same quantum state. The state for a 5-cent coin, for example, might be the pure state $|\psi_5\rangle$ and the bank produces many copies $|\psi_5\rangle^{\otimes 1000000}$, issuing one copy to each person who withdraws 5 cents from the bank. We use the term *quantum coin* because physical coins in the real world have the same property: there should be no discernible difference between different coins of the same denomination. The specification of a quantum coin scheme consists of the specification of the money state and the verification circuit.

Scheme	**Type**	**Anonymous**	**Unforgeable**	**Efficiently locally verifiable**	**Transferable**
Physical coins	coin	yes	physically	yes	yes
Physical bills	bill	no	physically	yes	yes
Classical digital cash	bill	yes	double-spending detection	yes	grows in size
[Wie83]	q. bill	no	yes	no	no
[BBBW82]	q. bill	no	yes	no	no
[TOI03]	q. bill	yes	yes	no	no
Quantum coins: black box	q. coin	partially	yes	yes	yes
Quantum coins: blind comp.	q. coin	partially	yes	no	yes

Figure 8.1: Summary of money schemes and their properties

Definition 8.1 *A* **quantum coin scheme** *is a pair $(V, |\psi\rangle)$, where $|\psi\rangle$ is an n-qubit pure state in a 2^n-dimensional Hilbert space $\mathcal{H}^{2^n}$, and V is a quantum circuit with a quantum n-qubit input register (denoted ρ), plus optional ancilla quantum registers, a classical output bit, and a quantum output register of n qubits.*

The basic scenario of how a quantum coin scheme would operate is as follows. A bank generates a large number of quantum coins and stores them. A user withdraws coins from the bank via a private quantum channel and stores the coins. When the user wishes to spend the coins, it transfers the coins to the merchant using a quantum channel. The merchant uses a quantum circuit to verify the coins; this procedure may or may not involve classical or quantum communication with the bank. Finally, the merchant stores the coins until redeeming them with the bank or issuing them as change to subsequent users.

8.3.1.1 Verification

In the most general setting, the verification circuit V operates on three registers: a 1-qubit data readout register, an n-qubit input register, and an arbitrary m-qubit ancilla. After applying V, the first register is measured, and the output is the decision on whether to accept the token as valid or not. If the input is a valid quantum coin $|\psi\rangle$, then, after the application of V and the measurement, the classical output should be 0 and the partial trace over the first and third registers should leave the second register in the same state $|\psi\rangle$. The circuit diagram is given in Figure 8.2.

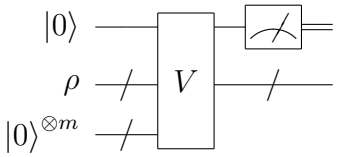

Figure 8.2: Generic verification circuit for a quantum coin scheme $(V, |\psi\rangle)$.

We cannot simply provide this circuit in an unprotected form to the public: it may be possible to decompose the circuit into component gates and find a way to forge money. In Section 8.4 we describe two techniques for implementing this circuit in a safe way: (1) black box verification, in which we assume the circuit is a black box and security rests on complexity-theoretic assumptions, and (2) blind quantum computation, which allows one party to implement an operation without gaining any information about the operation being performed, and security is information-theoretic. It could be possible to construct a scheme based on computational assumptions.

8.3.1.2 Unforgeability

We assume that a forger has the verification circuit V and many (or all) tokens issued, say k of them. The goal of a forger is to produce a state that passes more than k verification tests with good probability. Since the verification circuit projects the state into the subspace spanned by $|\psi\rangle$, this is equivalent to creating a state that has good overlap with the state $|\psi\rangle^{\otimes k+1}$.

Definition 8.2 *A quantum coin scheme $(V, |\psi\rangle)$, where $|\psi\rangle$ is an n-qubit state, is* **unforgeable** *if, given the verification circuit V and k copies of the state $|\psi\rangle$, for any $k \geq 0$, $k \in \text{poly}(n)$, it is not possible for a quantum adversary running in time $\text{poly}(n)$ to produce a state ρ such that $\langle\psi|^{\otimes k+1} \rho |\psi\rangle^{\otimes k+1}$ is non-negligible (in n).*[1]

In order to prevent a counterfeiter from performing quantum state tomography [AJK04] and precisely determining the state $|\psi\rangle$, the bank should avoid issuing more than a polynomial number (in n) of coins.

Information theoretically, no offline quantum coin scheme can be perfectly unforgeable (that is, with $\langle\psi|^{\otimes k+1} \rho |\psi\rangle^{\otimes k+1} = 0$ and no running time restriction in Definition 8.2). If a forger has a verification circuit and unbounded quantum computational resources, the forger can repeatedly generate test states until one such state passes; after verification, this state is projected into a valid money state and can subsequently be used as a money token. Thus, we must introduce computational assumptions on a forger and attempt to lower bound the amount of work required to forge.

Without any further specification of the quantum coin scheme and the verification circuit, we cannot say anything more about the unforgeability of such schemes. In Section 8.4.2, we show that a black box quantum coin scheme is unforgeable.

8.3.1.3 Anonymity

In our ideal formulation, all quantum coins (for a particular denomination) are minted as the same quantum state $|\psi\rangle$. However, the bank could create quantum coins from different quantum states, all of which can be verified by a particular verification circuit. Although we have no procedure for users to test the anonymity of the system, it would be possible for a regulator to regularly review the procedures of the bank and ensure that it is issuing identical tokens as the coins. If indeed all the coins issued are identical, then it is impossible for the use of a coin to be tracked. If quantum circuits can be obfuscated, then the verification circuit could be provided in an obfuscated form as a fixed public classical string which merchants then implement; since the circuit is fixed for all merchants, this would give anonymity to merchants as well. If an interactive protocol is required for verification (as in our

[1] In the language of Aaronson [Aar09], this is a single key public key quantum money scheme with completeness error 0 and soundness error negligible in n.

use of blind quantum computing in Section 8.5), then anonymous classical [BT07] and quantum [BBF+07] communication can be used to improve the anonymity of merchants.

8.3.2 Quantum bills

Whereas all quantum coins of the same denomination are identical states, with *quantum bills* we allow tokens of the same denomination to be different quantum states and additionally allow some classical information associated with each quantum state. So a bank might issue a set of states $\{(s_i, |\psi_i\rangle) : i \in \Gamma\}$ as the valid \$20 bills. This corresponds to physical bills which have a distinct serial number on each bill.

Definition 8.3 *A* **quantum bill scheme** *is a pair* $(V, \{(s_i, |\psi_i\rangle) : i \in \Gamma\})$*, where* Γ *is a finite set, and for each* $i \in \Gamma$*,* s_i *is a label in a set* $\mathcal{S}$*,* $|\psi_i\rangle$ *is an* n*-qubit pure state in a* 2^n*-dimensional Hilbert space* $\mathcal{H}^{2^n}$*. Moreover,* V *is a quantum circuit with a quantum input register (denoted* $|s\rangle$*), a quantum* n*-qubit input register (denoted* ρ*), plus optional ancilla quantum registers, a classical output bit, and a quantum output register of* n *qubits.*[2]

8.3.2.1 Verification

A generic verification circuit for a quantum bill scheme is given in Figure 8.3.

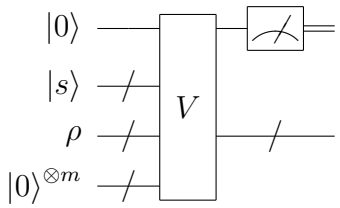

Figure 8.3: Generic verification circuit for a quantum bill scheme $(V, \{(s_i, |\psi_i\rangle) : i \in \Gamma\})$.

The use of the classical label s_i may vary according to the scheme. For example, in the schemes of Wiesner [Wie83] and Bennett *et al.* [BBBW82], s_i is a serial number that allows the issuer to retrieve the verification details, while in the scheme of Tokunga *et al.* [TOI03], s_i is effectively unused; in their scheme it is used to represent the denomination of the bill (e.g., \$5), but in our formulation the denomination is fixed for a particular scheme so the label is effectively the empty string for all $i \in \Gamma$. Schemes where s_i is non-trivial and unchanged by verification inherently limit the anonymity of the scheme, just as the serial number on physical bills places some limits on anonymity.

[2] In the language of Aaronson [Aar09], this is a public key quantum money scheme.

While all previous quantum money schemes discussed in Section 8.1 are classified as quantum bill schemes based on the above definition, none of them satisfy all of the security properties described in Section 8.2. In particular, no previous quantum money scheme is offline verifiable: all previous schemes require that the issuer verify a token via quantum communication, a requirement which we aim to remove for quantum coins. In the rest of this chapter, we are only concerned with quantum coin schemes, not quantum bill schemes.

8.4 Black box quantum coins

Our first implementation for quantum coins works in the black box circuit model. We assume the verification circuit provided to the public is a black box: "anything one can compute from it one could also compute from the input-output behavior of the program" [BGI+01a, p. 2]. With this assumption, we present a scheme in which coins are unforgeable. The scheme allows coins to be transferred an arbitrary number of times. The use of a black box circuit means that coins can be verified locally without any communication, classical or quantum, with the bank.

We note that it is not known at present whether a quantum circuit can be implemented as a true black box. There are pessimistic results about the ability to obfuscate classical circuits [BGI+01b], although loopholes do exist: for example, point functions can be obfuscated [Wee05]. However, no results are known about quantum circuits. Another classical technique for black box computation is physically tamper-proof hardware, but again the parallel in quantum computation is not clear.

In our black box construction, a coin is a randomly chosen secret state, and the verification circuit recognizes precisely that state using an oracle like the iterate in amplitude amplification [BBHT98].

Let $|\psi\rangle$ a pure state chosen randomly (according to the Haar measure) from among the pure states in $\mathcal{H}^{2^n}$. The verification oracle is $U_\psi = I - 2\,|\psi\rangle\,\langle\psi|$. Since this is a black-box oracle scheme, the unforgeability proof of Section 8.4.2 applies and the scheme is unforgeable in the black-box oracle model.

In practice, however, choosing a pure state $|\psi\rangle$ randomly according to the Haar measure with the additional constraints that we must be able to compute $I - 2\,|\psi\rangle\,\langle\psi|$ and that we must be able to produce many copies of $|\psi\rangle$ is problematic and it is not known how to do so in polynomial time. Recent work has focused on developing *approximate quantum t-designs* [AE07] where, roughly speaking, t copies of a state can be efficiently constructed such that tensor product state is sufficiently close to t copies of a state selected uniformly at random according to the Haar measure. Aaronson [Aar09, Theorem 8] gives a technique for constructing $t \in \text{poly}(n)$ copies of a pseudorandom state that are nearly indistinguishable (that is, negligibly different) from t copies of a truly random state by any measurement, even allowing the measurement procedure to make $\text{poly}(n)$ calls to an oracle U_ψ

recognizing the state. Aaronson's technique allows us to use pseudorandom states instead of truly random states with a negligible loss in security.

We note that, for quantum coins, it is not sufficient to choose a random binary string encoded randomly in a pair of non-orthogonal bases, such as the so-called "BB84" bases. An adversary with a small number of quantum coins, say $O(\log n)$, can measure each qubit of the $O(\log n)$ tokens in both bases, and will with good probability find the correct basis choices and thus the random binary string, allowing her to then create arbitrarily many forged coins.

8.4.1 Verification

Let U_ψ be an oracle that recognizes the state $|\psi\rangle$ by flipping the sign of the phase of the state $|\psi\rangle$. That is, $U_\psi |\psi\rangle = -|\psi\rangle$ and $U_\psi |\phi\rangle = |\phi\rangle$ for all $|\phi\rangle$ orthogonal to $|\psi\rangle$; in other words, $U_\psi = I - 2|\psi\rangle\langle\psi|$.

We can construct a verification circuit V from the oracle U_ψ as follows. On the data readout register, input the state $|0\rangle$, then perform a Hadamard transformation on the ancilla. Use the ancilla as the control bit of a controlled-U_ψ applied to the input state ρ. Then perform a Hadamard transformation again on the ancilla and measure it in the computational basis. The circuit diagram is given in Figure 8.4.

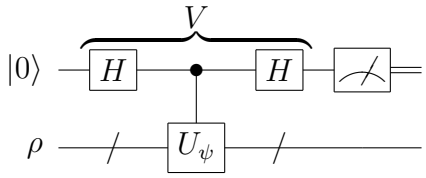

Figure 8.4: Verification circuit for quantum coins $|\psi\rangle$ recognized using the oracle U_ψ.

When a measurement in the computational basis is performed on the ancilla register, the result will be $|1\rangle$ when the input state ρ is $|\psi\rangle$ and $|0\rangle$ when the input state is $|\phi\rangle$ for $\langle\phi|\psi\rangle = 0$. Moreover, the state on the second register remains unchanged when its input is $|\psi\rangle$.

The fact that a valid token is unchanged by the verification process allows transferability of quantum coins. When a customer spends a quantum coin at a store, the merchant, after verifying and accepting the coin, can retain the coin until the merchant needs to make change. At that time, the merchant can give the coin to another user who, after optionally verifying the coin, can use that coin in another transaction. (In fact, the verification process not only enables transferability but also enhances the robustness of the quantum coins. Although over time a quantum state may decohere, at verification time the token may still be sufficiently close to the expected state $|\psi\rangle$ to pass the verification process with high probability. If it does pass, then the measurement process will project the coin back into the original state $|\psi\rangle$.)

Security. The verification procedure described in the previous section yields a correct quantum money scheme: valid money tokens are recognized. We now discuss the security of such a scheme. For unforgeability, we want that invalid tokens are recognized as being invalid and that it is difficult to forge new money.

8.4.2 Black-box unforgeability

To analyze the forgeability of the quantum coin scheme given in Figure 8.4, we suppose that the circuit for the unitary U_ψ is a black box, meaning that no information can be obtained from observing its inner workings; equivalently, we assume that U_ψ is given as an oracle. Having made this assumption, we proceed to obtain a lower bound on the number of queries to the oracle that must be made in order to produce a state that has a particular overlap p with $|\psi\rangle^{\otimes k+1}$, when the adversary is only given k coins. We show this result in the next section.

Definition 8.4 *A quantum coin scheme $(V, |\psi\rangle)$, where $|\psi\rangle$ is an n-qubit state, is* **black-box unforgeable** *if, given an oracle U_ψ recognizing the state $|\psi\rangle$ and k copies of the state $|\psi\rangle$, for any $k \geq 0$, $k \in \mathrm{poly}(n)$, it is not possible for a quantum adversary using $\mathrm{poly}(n)$ queries to U_ψ to produce a state ρ such that $\langle\psi|^{\otimes k+1} \rho |\psi\rangle^{\otimes k+1}$ is non-negligible.*[3]

We note that our definition of unforgeability has the adversary producing a $(k+1)$-register state, each register of which should overlap well with $|\psi\rangle$. An alternative formulation could be that the adversary needs to produce a multi-register state such that some $k+1$ of its registers, but not necessarily all of its registers, overlap well with $|\psi\rangle$. These definitions are equivalent. The adversary has access to a verification oracle and, for each of the many registers it constructs, could simply apply the verification oracle to each register and then trace out any registers that do not pass verification. This requires additional calls to the verification oracle, but still only $\mathrm{poly}(n)$ calls to the oracle (since a polynomial-time adversary can only construct $\mathrm{poly}(n)$ registers), and hence remains within the constraints of the security argument above.

We note as well that it is not necessary to extend this definition to $k+\ell$ copies of $|\psi\rangle$: any adversary who can construct $k+\ell$ copies of $|\psi\rangle$ with non-negligible probability can in particular construct $k+1$ copies of $|\psi\rangle$ with non-negligible probability. In other words, there are no "long shots" that pay off in expected value: the definition precludes being able to generate a very large number of coins with a very small probability but with non-negligible expected number of coins.

We now aim to show that a generic quantum coin scheme implemented with black-box oracles as in Figure 8.4 is black-box unforgeable. However, we cannot use the basic no-cloning theorem [WZ82, Die82] or the result on approximate cloning

[3] In the language of Aaronson [Aar09], this is a single key private key quantum money scheme with completeness error 0 and soundness error negligible in n.

[BM07] because not only does a forger have copies of the state $|\psi\rangle$, the forger also has an oracle U_ψ that will indicate whether the attempted cloning was successful. Similarly, we cannot directly apply the $\Omega(\sqrt{N})$ lower bound on quantum search [BBBV97] because the forger has not only an oracle U_ψ recognizing the desired state but also some copies of the state itself. Rather, we need a hybrid of these two results.

Aaronson [Aar05] gives the following complexity-theoretic version of the no-cloning theorem that combines the lower bound for quantum search with the no-cloning theorem.

Theorem 8.5 (Theorem 5, [Aar05]) *Let $|\psi\rangle$ be an n-qubit pure state. Suppose we are given the initial state $|\psi\rangle^{\otimes k}$ for some $k \geq 1$ as well as an oracle U_ψ such that $U_\psi |\psi\rangle = -|\psi\rangle$ and $U_\psi |\phi\rangle = |\phi\rangle$ whenever $\langle\phi|\psi\rangle = 0$. Then to prepare a state ρ such that*

$$\langle\psi|^{k+1} \rho |\psi\rangle^{k+1} \geq p \tag{8.1}$$

we need

$$\Omega\left(\frac{\sqrt{2^n p}}{k \log k} - k\right) \tag{8.2}$$

queries to U_ψ.

This allows us to show that a quantum coin scheme is unforgeable in the black-box oracle model.

Theorem 8.6 *Let $(V, |\psi\rangle)$ be a quantum coin scheme, where V is as in Figure 8.4 with U_ψ given as a black-box oracle, and $|\psi\rangle$ is an n-qubit pure state. If not more than $\mathrm{poly}(n)$ coins are issued, then $(V, |\psi\rangle)$ is black-box unforgeable.*

Proof. Suppose otherwise. Then there exists an adversary who, upon receiving k copies of $|\psi\rangle$ and using $q = \mathrm{poly}(n)$ queries to U_ψ, can produce a state ρ such that $\langle\psi|^{\otimes k+1} \rho |\psi\rangle^{\otimes k+1} = p \in 1/\mathrm{poly}(n)$. By Theorem 8.5, we need

$$q = \Omega\left(\frac{\sqrt{2^n p}}{k \log k} - k\right) = \Omega\left(\frac{\sqrt{2^n/\mathrm{poly}(n)}}{\mathrm{poly}(n) \log \mathrm{poly}(n)} - \mathrm{poly}(n)\right) = \Omega\left(\frac{\sqrt{2^n}}{\mathrm{poly}(n)}\right) \tag{8.3}$$

queries to U_ψ. But since the adversary is allowed only a polynomial number q of queries to U_ψ, we have that $q \in \mathrm{poly}(n)$ and hence $\mathrm{poly}(n) = \Omega\left(\frac{\sqrt{2^n}}{\mathrm{poly}(n)}\right)$, which is a contradiction. Thus the quantum coin scheme must be black-box unforgeable. □

8.5 Quantum coins using blind quantum computation

Blind quantum computation allows one party, Alice, to have another party, Bob, perform computations on her behalf without Bob learning any information about the input state, output state, or the operation performed.

Blind quantum computation was first introduced by Childs [Chi05] under the name "secure assisted quantum communication". In Childs' approach, an Alice who has limited quantum computational abilities (quantum communication, quantum storage, and controlled-X and controlled-Z gates) can have Bob securely perform arbitrary quantum computation, with quantum input and quantum output. In order to implement the protocol, Alice and Bob must perform large amounts of quantum communication, though this could be replaced by quantum teleportation (shared entanglement with Bell measurements and classical communication).

Other work on blind quantum computation exists but is not fully suitable for use in the context of quantum money. Aharonov, Ben-Or, and Eban [ABOE08] present, via the language of computational complexity, a mechanism for blind quantum circuit evaluation. Since they are interested in complexity theory, their work is stated only for classical input and classical output, although it may be possible to modify their work for quantum input and quantum output as would be required for quantum coins. Broadbent, Fitzsimons, and Kashefi [BFK08] present a protocol for blind quantum computation with quantum input and output using measurement-based quantum computation that needs only two rounds of quantum communication: one at the beginning and one at the end. However the security argument protects only against individual, not coherent attacks (though no attack is known). We have a procedure to eliminate the last round of quantum communication, but again this only has security shown in the case of individual, not coherent, attacks.

Thus at present, barring an implementation of black box quantum circuits, we must either use teleportation or Childs' blind quantum computation to implement quantum coin verification.

Childs' blind quantum computation scheme could be used as follows. The merchant, playing the role of Bob, implements the verification circuit blindly for the bank, playing the role of Alice. The merchant receives the coin as the input to the circuit, and interacts quantumly with the bank who helps it implement the circuit. In the end, the output state along with the accept/reject information is with the merchant.

Because Childs' protocol requires large amounts of quantum communication between the bank and the merchant, this is little better than having the merchant teleport the coin to the bank who processes the coin and teleports it back.

The approach taken by Broadbent, Fitzsimons, and Kashefi, and our extension to it, is more promising from a resource perspective: the only quantum communication is the distribution of qubits for the formation of a measurement-based circuit at the beginning of the protocol; all later communication can be classical. This eliminates the need for an online quantum communications channel. It also means the bank need not store quantum states; while in such a setting the bank would have the ability to store quantum states (since coins themselves are quantum states), the bank having to store a large number of qubits for each verification circuit could be quite expensive, whereas in this setting it would only need to store classical data. This scheme does not yet have arguments for security against coherent attacks.

Appendix A

Sample Code

Contents

A.1 Unified point addition formulæ in elliptic curve cryptography (Chapter 6)

A.1.1 Projective unified formula of Brier and Joye (Section 6.3.1)

A.1.1.1 `ecp_unif_bj02.c`

The following block of C code implements the projective version of the unified point addition formula of Brier and Joye as in equation (6.3).

This code uses the big integer library and elliptic curve library that is part of the Netscape Security Services (NSS) toolkit [Moz08] found in the `mpi` and `ecl` subdirectories in of `mozilla/security/nss/lib/freebl` in that software package.

```
1 #include "ecp.h"
  #include "mplogic.h"
  #include <stdlib.h>
```

```
#ifdef ECL_DEBUG
#include <assert.h>
#endif

/* Converts a point P(px, py) from affine coordinates to projective
   coordinates R(rx, ry, rz). Assumes input is already field-encoded
   using field_enc, and returns output that is still field-encoded. */
mp_err ec_GFp_pt_aff2proj(const mp_int *px, const mp_int *py, mp_int *
   rx, mp_int *ry, mp_int *rz, const ECGroup *group) {
   mp_err res = MP_OKAY;

   if (ec_GFp_pt_is_inf_aff(px, py) == MP_YES) {
      MP_CHECKOK(ec_GFp_pt_set_inf_proj(rx, ry, rz));
   } else {
      MP_CHECKOK(mp_copy(px, rx));
      MP_CHECKOK(mp_copy(py, ry));
      MP_CHECKOK(mp_set_int(rz, 1));
      if (group->meth->field_enc) {
         MP_CHECKOK(group->meth->field_enc(rz, rz, group->meth));
      }
   }
  CLEANUP:
   return res;
}

/* Converts a point P(px, py, pz) from projective coordinates to
   affine coordinates R(rx, ry). P and R can share x and y
   coordinates. Assumes input is already field-encoded using field_enc
   , and returns output that is still field-encoded. */
mp_err ec_GFp_pt_proj2aff(const mp_int *px, const mp_int *py, const
   mp_int *pz, mp_int *rx, mp_int *ry, const ECGroup *group) {
   mp_err res = MP_OKAY;
   mp_int z1;

   MP_DIGITS(&z1) = 0;
   MP_CHECKOK(mp_init(&z1));

   /* if point at infinity, then set point at infinity and exit */
   if (ec_GFp_pt_is_inf_proj(px, py, pz) == MP_YES) {
      MP_CHECKOK(ec_GFp_pt_set_inf_aff(rx, ry));
      goto CLEANUP;
   }

   /* transform (px, py, pz) into (px / pz, py / pz) */
   if (mp_cmp_d(pz, 1) == 0) {
      MP_CHECKOK(mp_copy(px, rx));
      MP_CHECKOK(mp_copy(py, ry));
   } else {
      MP_CHECKOK(group->meth->field_div(NULL, pz, &z1, group->meth));
      MP_CHECKOK(group->meth->field_mul(px, &z1, rx, group->meth));
      MP_CHECKOK(group->meth->field_mul(py, &z1, ry, group->meth));
   }

  CLEANUP:
   mp_clear(&z1);
   return res;
}

mp_err ec_GFp_pt_add_unif_bj02_proj(const mp_int *px, const mp_int *py
   , const mp_int *pz, const mp_int *qx, const mp_int *qy, const
```

```
    mp_int *qz, mp_int *rx, mp_int *ry, mp_int *rz, const ECGroup *
    group) {
56  mp_err res = MP_OKAY;
    mp_int u1, u2, t, s1, s2, m, z, f, l, g, r, w;
    char s[1000];

    MP_DIGITS(&u1) = 0; MP_DIGITS(&u2) = 0; MP_DIGITS(&t) = 0;
61  MP_DIGITS(&s1) = 0; MP_DIGITS(&s2) = 0; MP_DIGITS(&m) = 0;
    MP_DIGITS(&z) = 0; MP_DIGITS(&f) = 0; MP_DIGITS(&l) = 0;
    MP_DIGITS(&g) = 0; MP_DIGITS(&r) = 0; MP_DIGITS(&w) = 0;
    MP_CHECKOK(mp_init(&u1)); MP_CHECKOK(mp_init(&u2));
    MP_CHECKOK(mp_init(&t)); MP_CHECKOK(mp_init(&s1));
66  MP_CHECKOK(mp_init(&s2)); MP_CHECKOK(mp_init(&m));
    MP_CHECKOK(mp_init(&z)); MP_CHECKOK(mp_init(&f));
    MP_CHECKOK(mp_init(&l)); MP_CHECKOK(mp_init(&g));
    MP_CHECKOK(mp_init(&r)); MP_CHECKOK(mp_init(&w));

71  /* If either P or Q is the point at infinity, then return the other
       point */
    if (ec_GFp_pt_is_inf_proj(px, py, pz) == MP_YES) {
       MP_CHECKOK(mp_copy(qx, rx));
       MP_CHECKOK(mp_copy(qy, ry));
       MP_CHECKOK(mp_copy(qz, rz));
76     goto CLEANUP;
    }
    if (ec_GFp_pt_is_inf_proj(qx, qy, qz) == MP_YES) {
       MP_CHECKOK(mp_copy(px, rx));
       MP_CHECKOK(mp_copy(py, ry));
81     MP_CHECKOK(mp_copy(pz, rz));
       goto CLEANUP;
    }

    /* u1 = x1 * z2, u2 = x2 * z1, t = u1 + u2 */
86  MP_CHECKOK(group->meth->field_mul(px, qz, &u1, group->meth));
    MP_CHECKOK(group->meth->field_mul(qx, pz, &u2, group->meth));
    MP_CHECKOK(group->meth->field_add(&u1, &u2, &t, group->meth));
    /* s1 = y1 * z2, s2 = y2 * z1, m = s1 + s2 */
    MP_CHECKOK(group->meth->field_mul(py, qz, &s1, group->meth));
91  MP_CHECKOK(group->meth->field_mul(qy, pz, &s2, group->meth));
    MP_CHECKOK(group->meth->field_add(&s1, &s2, &m, group->meth));
    /* z = z1 * z2, f = z * m, l = m * f, g = t * l */
    MP_CHECKOK(group->meth->field_mul(pz, qz, &z, group->meth));
    MP_CHECKOK(group->meth->field_mul(&z, &m, &f, group->meth));
96  MP_CHECKOK(group->meth->field_mul(&m, &f, &l, group->meth));
    MP_CHECKOK(group->meth->field_mul(&t, &l, &g, group->meth));
    /* r = t^2 - u1 * u2 + a * z^2 */
    MP_CHECKOK(group->meth->field_sqr(&t, &r, group->meth));
    MP_CHECKOK(group->meth->field_mul(&u1, &u2, &w, group->meth));
101 MP_CHECKOK(group->meth->field_sub(&r, &w, &r, group->meth));
    MP_CHECKOK(group->meth->field_sqr(&z, &w, group->meth));
    MP_CHECKOK(group->meth->field_mul(&group->curvea, &w, &w, group->
       meth));
    MP_CHECKOK(group->meth->field_add(&r, &w, &r, group->meth));
    /* w = r^2 - g */
106 MP_CHECKOK(group->meth->field_sqr(&r, &w, group->meth));
    MP_CHECKOK(group->meth->field_sub(&w, &g, &w, group->meth));
    /* x3 = 2 * f * w */
    MP_CHECKOK(group->meth->field_mul(&f, &w, rx, group->meth));
    MP_CHECKOK(group->meth->field_add(rx, rx, rx, group->meth));
111 /* y3 = r * (g - 2 * w) - l^2 */
```

```
    MP_CHECKOK(group->meth->field_sub(&g, &w, ry, group->meth));
    MP_CHECKOK(group->meth->field_sub(ry, &w, ry, group->meth));
    MP_CHECKOK(group->meth->field_mul(&r, ry, ry, group->meth));
    MP_CHECKOK(group->meth->field_sqr(&l, rz, group->meth));
    MP_CHECKOK(group->meth->field_sub(ry, rz, ry, group->meth));
    /* z3 = 2 * f^3 */
    MP_CHECKOK(group->meth->field_sqr(&f, rz, group->meth));
    MP_CHECKOK(group->meth->field_mul(&f, rz, rz, group->meth));
    MP_CHECKOK(group->meth->field_add(rz, rz, rz, group->meth));

  CLEANUP:
    mp_clear(&u1); mp_clear(&u2); mp_clear(&t);
    mp_clear(&s1); mp_clear(&s2); mp_clear(&m);
    mp_clear(&z); mp_clear(&f); mp_clear(&l);
    mp_clear(&g); mp_clear(&r); mp_clear(&w);
    return res;
}

mp_err ec_GFp_pt_mul_unif_bj02_proj(const mp_int *n, const mp_int *px,
    const mp_int *py, mp_int *rx, mp_int *ry, const ECGroup *group) {
    mp_err res = MP_OKAY;
    mp_int k, qx, qy, qz, sx, sy, sz;
    int i, l;

    MP_DIGITS(&k) = 0;
    MP_DIGITS(&qx) = 0; MP_DIGITS(&qy) = 0; MP_DIGITS(&qz) = 0;
    MP_DIGITS(&sx) = 0; MP_DIGITS(&sy) = 0; MP_DIGITS(&sz) = 0;
    MP_CHECKOK(mp_init(&k)); MP_CHECKOK(mp_init(&qx));
    MP_CHECKOK(mp_init(&qy)); MP_CHECKOK(mp_init(&qz));
    MP_CHECKOK(mp_init(&sx)); MP_CHECKOK(mp_init(&sy));
    MP_CHECKOK(mp_init(&sz));

    /* if n = 0 then r = inf */
    if (mp_cmp_z(n) == 0) {
        mp_zero(rx);
        mp_zero(ry);
        res = MP_OKAY;
        goto CLEANUP;
    }
    /* Q = P, k = n */
    MP_CHECKOK(ec_GFp_pt_aff2proj(px, py, &qx, &qy, &qz, group));
    MP_CHECKOK(mp_copy(n, &k));
    /* if n < 0 then Q = -Q, k = -k */
    if (mp_cmp_z(n) < 0) {
        MP_CHECKOK(group->meth->field_neg(py, &qy, group->meth));
        MP_CHECKOK(ec_GFp_pt_aff2proj(px, &qy, &qx, &qy, &qz, group));
        MP_CHECKOK(mp_neg(&k, &k));
    }
    l = mpl_significant_bits(&k) - 1;
    MP_CHECKOK(mp_copy(&qx, &sx));
    MP_CHECKOK(mp_copy(&qy, &sy));
    MP_CHECKOK(mp_copy(&qz, &sz));
    for (i = l - 1; i >= 0; i--) {
        /* S = 2S */
        MP_CHECKOK(ec_GFp_pt_add_unif_bj02_proj(&sx, &sy, &sz, &sx, &sy,
            &sz, &sx, &sy, &sz, group));
        /* if k_i = 1, then S = S + Q */
        if (mpl_get_bit(&k, i) != 0) {
            MP_CHECKOK(ec_GFp_pt_add_unif_bj02_proj(&sx, &sy, &sz, &qx, &
                qy, &qz, &sx, &sy, &sz, group));
```

```
        }
    }
    /* output S */
    MP_CHECKOK(ec_GFp_pt_proj2aff(&sx, &sy, &sz, rx, ry, group));

  CLEANUP:
    mp_clear(&k);
    mp_clear(&qx); mp_clear(&qy); mp_clear(&qz);
    mp_clear(&sx); mp_clear(&sy); mp_clear(&sz);
    return res;
}
```

A.1.2 Affine unified formula of Brier, Déchène, and Joye (Section 6.3.2)

A.1.2.1 ecp_unif_bdj04_aff.c

The following block of C code implements the affine version of the unified point addition formula of Brier, Déchène, and Joye as in equation (6.7). This code uses the big integer library and elliptic curve library that is part of the Netscape Security Services (NSS) toolkit [Moz08].

```
#include "ecp.h"
#include "mplogic.h"
#include <stdlib.h>
#include "hrtime.h"

mp_err ec_GFp_pt_add_unif_bdj04(const mp_int *px, const mp_int *py,
    const mp_int *qx, const mp_int *qy, mp_int *rx, mp_int *ry, const
    ECGroup *group) {
    mp_err res = MP_OKAY;
    mp_int t1, t2, t3, t4;

    MP_DIGITS(&t1) = 0; MP_DIGITS(&t2) = 0;
    MP_DIGITS(&t3) = 0; MP_DIGITS(&t4) = 0;
    MP_CHECKOK(mp_init(&t1)); MP_CHECKOK(mp_init(&t2));
    MP_CHECKOK(mp_init(&t3)); MP_CHECKOK(mp_init(&t4));

    /* if P = inf, then R = Q */
    if (ec_GFp_pt_is_inf_aff(px, py) == 0) {
        MP_CHECKOK(mp_copy(qx, rx));
        MP_CHECKOK(mp_copy(qy, ry));
        res = MP_OKAY;
        goto CLEANUP;
    }
    /* if Q = inf, then R = P */
    if (ec_GFp_pt_is_inf_aff(qx, qy) == 0) {
        MP_CHECKOK(mp_copy(px, rx));
        MP_CHECKOK(mp_copy(py, ry));
        res = MP_OKAY;
        goto CLEANUP;
    }
    /* if P = -Q, then R = inf */
    MP_CHECKOK(group->meth->field_neg(qy, &t1, group->meth));
    if ((mp_cmp(px, qx) == 0) && (mp_cmp(py, &t1) == 0)) {
        mp_zero(rx);
```

```
      mp_zero(ry);
      res = MP_OKAY;
      goto CLEANUP;
   }

   MP_CHECKOK(group->meth->field_add(px, qx, &t1, group->meth));
   MP_CHECKOK(group->meth->field_sqr(&t1, &t2, group->meth));
   MP_CHECKOK(group->meth->field_mul(px, qx, &t3, group->meth));
   MP_CHECKOK(group->meth->field_sub(&t2, &t3, &t2, group->meth));
   MP_CHECKOK(group->meth->field_add(&t2, &group->curvea, &t2, group->
      meth));

   MP_CHECKOK(group->meth->field_add(py, qy, &t4, group->meth));

   MP_CHECKOK(group->meth->field_sub(qx, px, &t3, group->meth));

   if (mp_cmp(&t3, &t3) == 0) {
      MP_CHECKOK(group->meth->field_add(&t2, qy, &t2, group->meth));
      MP_CHECKOK(group->meth->field_sub(&t2, py, &t2, group->meth));
      MP_CHECKOK(group->meth->field_add(&t4, qx, &t4, group->meth));
      MP_CHECKOK(group->meth->field_sub(&t4, px, &t4, group->meth));
   } else {
      MP_CHECKOK(group->meth->field_add(&t2, py, &t2, group->meth));
      MP_CHECKOK(group->meth->field_sub(&t2, qy, &t2, group->meth));
      MP_CHECKOK(group->meth->field_add(&t4, px, &t4, group->meth));
      MP_CHECKOK(group->meth->field_sub(&t4, qx, &t4, group->meth));
   }

   MP_CHECKOK(group->meth->field_div(&t2, &t4, &t3, group->meth));

   MP_CHECKOK(group->meth->field_sqr(&t3, &t2, group->meth));
   MP_CHECKOK(group->meth->field_sub(&t2, &t1, &t2, group->meth));

   MP_CHECKOK(group->meth->field_sub(px, &t2, &t4, group->meth));
   MP_CHECKOK(group->meth->field_mul(&t4, &t3, &t4, group->meth));
   MP_CHECKOK(group->meth->field_sub(&t4, py, &t4, group->meth));

   MP_CHECKOK(mp_copy(&t2, rx));
   MP_CHECKOK(mp_copy(&t4, ry));

  CLEANUP:
   mp_clear(&t1);
   mp_clear(&t2);
   mp_clear(&t3);
   mp_clear(&t4);
   return res;
}

mp_err ec_GFp_pt_mul_unif_bdj04(const mp_int *n, const mp_int *px,
   const mp_int *py, mp_int *rx, mp_int *ry, const ECGroup *group) {
   mp_err res = MP_OKAY;
   mp_int k, k3, qx, qy, qny, sx, sy;
   int b1, b3, i, l;
#ifdef ECL_HRTIME
   M_Time_DeclareVariables
#endif

   MP_DIGITS(&k) = 0; MP_DIGITS(&k3) = 0; MP_DIGITS(&qx) = 0;
   MP_DIGITS(&qy) = 0; MP_DIGITS(&qny) = 0; MP_DIGITS(&sx) = 0;
   MP_DIGITS(&sy) = 0;
```

```
91      MP_CHECKOK(mp_init(&k)); MP_CHECKOK(mp_init(&k3));
        MP_CHECKOK(mp_init(&qx)); MP_CHECKOK(mp_init(&qy));
        MP_CHECKOK(mp_init(&qny)); MP_CHECKOK(mp_init(&sx));
        MP_CHECKOK(mp_init(&sy));

96      /* if n = 0 then r = inf */
        if (mp_cmp_z(n) == 0) {
            mp_zero(rx);
            mp_zero(ry);
            res = MP_OKAY;
101         goto CLEANUP;
        }
        /* Q = P, k = n */
        MP_CHECKOK(mp_copy(px, &qx));
        MP_CHECKOK(mp_copy(py, &qy));
106     MP_CHECKOK(mp_copy(n, &k));
        /* if n < 0 then Q = -Q, k = -k */
        if (mp_cmp_z(n) < 0) {
            MP_CHECKOK(group->meth->field_neg(&qy, &qy, group->meth));
            MP_CHECKOK(mp_neg(&k, &k));
111     }

        l = mpl_significant_bits(&k) - 1;
        MP_CHECKOK(mp_copy(&qx, &sx));
        MP_CHECKOK(mp_copy(&qy, &sy));
116     for (i = l - 1; i >= 0; i--) {
            /* S = 2S */
    #ifdef ECL_HRTIME
            M_Time_Start
    #endif
121         MP_CHECKOK(ec_GFp_pt_add_unif_bdj04(&sx, &sy, &sx, &sy, &sx, &sy
                , group));
    #ifdef ECL_HRTIME
            M_Time_Stop
            printf("d: %llu\n", M_Time_Difference);
    #endif
126         /* if k_i = 1, then S = S + Q */
            if (mpl_get_bit(&k, i) != 0) {
    #ifdef ECL_HRTIME
                M_Time_Start
    #endif
131             MP_CHECKOK(ec_GFp_pt_add_unif_bdj04(&sx, &sy, &qx, &qy, &sx,
                    &sy, group));
    #ifdef ECL_HRTIME
                M_Time_Stop
                printf("a: %llu\n", M_Time_Difference);
    #endif
136         }
        }
        /* output S */
        MP_CHECKOK(mp_copy(&sx, rx));
        MP_CHECKOK(mp_copy(&sy, ry));
141
    CLEANUP:
        mp_clear(&k); mp_clear(&k3); mp_clear(&qx);
        mp_clear(&qy); mp_clear(&sx); mp_clear(&sy);
        return res;
146 }
```

A.1.3 Projective unified formula of Brier, Déchène, and Joye (Section 6.3.2)

A.1.3.1 Formula verification in Maple

The following block of Maple code confirms that the projective unified point addition formula given in equation (6.8) mathematically corresponds to the affine formula of Brier, Déchène, and Joye [BDJ04] in equation (6.7).

```
restart;
U1 := X1*Z2; U2 := X2*Z1; S1 := Y1*Z2; S2 := Y2*Z1; Z := Z1*Z2;
T := U1+U2; V := U1-U2; M := S1+S2; N := S1-S2;
E := M+V; F := Z*E; L := F*E; G := L*T;
R := T^2-U1*U2+a*Z^2+Z*N; W := R^2-G;
Z3 := 2*F^3; X3 := 2*F*W; Y3 := R*(G-2*W)-L*F*M;
X1 := x1*Z1; Y1 := y1*Z1; X2 := x2*Z2; Y2 := y2*Z2;
lambda := ((x1+x2)^2-x1*x2+a+y1-y2)/(y1+y2+x1-x2);
x3 := lambda^2-x1-x2; y3 := 1/2*(lambda*(x1+x2-2*x3)-(y1+y2));
simplify(X3-x3*Z3)=0;
simplify(Y3-y3*Z3)=0;
```

A.1.3.2 `ecp_unif_bdj04_proj.c`

The following block of C code implements the projective version of the unified point addition formula of Brier, Déchène, and Joye as in equation (6.8). This code uses the big integer library and elliptic curve library that is part of the Netscape Security Services (NSS) toolkit [Moz08].

```
#include "ecp.h"
#include "mplogic.h"
#include <stdlib.h>
#include "hrtime.h"

mp_err ec_GFp_pt_add_unif_bdj04_proj(const mp_int *px, const mp_int *
    py, const mp_int *pz, const mp_int *qx, const mp_int *qy, const
    mp_int *qz, mp_int *rx, mp_int *ry, mp_int *rz, const ECGroup *
    group) {
    mp_err res = MP_OKAY;
    mp_int u1, u2, t, v, s1, s2, m, n, z, f, e, l, g, r, w;
    char s[1000];

    MP_CHECKOK(mp_init(&u1)); MP_CHECKOK(mp_init(&u2));
    MP_CHECKOK(mp_init(&t)); MP_CHECKOK(mp_init(&v));
    MP_CHECKOK(mp_init(&s1)); MP_CHECKOK(mp_init(&s2));
    MP_CHECKOK(mp_init(&m)); MP_CHECKOK(mp_init(&n));
    MP_CHECKOK(mp_init(&z)); MP_CHECKOK(mp_init(&f));
    MP_CHECKOK(mp_init(&e)); MP_CHECKOK(mp_init(&l));
    MP_CHECKOK(mp_init(&g)); MP_CHECKOK(mp_init(&r));
    MP_CHECKOK(mp_init(&w));

    /* If either P or Q is the point at infinity, then return the other
        point */
    if (ec_GFp_pt_is_inf_proj(px, py, pz) == MP_YES) {
        MP_CHECKOK(mp_copy(qx, rx));
        MP_CHECKOK(mp_copy(qy, ry));
```

```
24          MP_CHECKOK(mp_copy(qz, rz));
            goto CLEANUP;
        }
        if (ec_GFp_pt_is_inf_proj(qx, qy, qz) == MP_YES) {
            MP_CHECKOK(mp_copy(px, rx));
29          MP_CHECKOK(mp_copy(py, ry));
            MP_CHECKOK(mp_copy(pz, rz));
            goto CLEANUP;
        }

34      /* u1 = x1 * z2, u2 = x2 * z1, t = u1 + u2, v = u1 - u2 */
        MP_CHECKOK(group->meth->field_mul(px, qz, &u1, group->meth));
        MP_CHECKOK(group->meth->field_mul(qx, pz, &u2, group->meth));
        MP_CHECKOK(group->meth->field_sub(&u1, &u2, &v, group->meth));
        MP_CHECKOK(group->meth->field_add(&u1, &u2, &t, group->meth));
39      /* s1 = y1 * z2, s2 = y2 * z1, m = s1 + s2, n = s1 - s2 */
        MP_CHECKOK(group->meth->field_mul(py, qz, &s1, group->meth));
        MP_CHECKOK(group->meth->field_mul(qy, pz, &s2, group->meth));
        MP_CHECKOK(group->meth->field_sub(&s1, &s2, &n, group->meth));
        MP_CHECKOK(group->meth->field_add(&s1, &s2, &m, group->meth));
44      /* z = z1 * z2, e = m + v, f = z * e, l = f * e, g = l * t */
        MP_CHECKOK(group->meth->field_mul(pz, qz, &z, group->meth));
        MP_CHECKOK(group->meth->field_add(&m, &v, &e, group->meth));
        MP_CHECKOK(group->meth->field_mul(&z, &e, &f, group->meth));
        MP_CHECKOK(group->meth->field_mul(&f, &e, &l, group->meth));
49      MP_CHECKOK(group->meth->field_mul(&l, &t, &g, group->meth));
        /* r = t^2 - u1 * u2 + a * z^2 + z * n */
        MP_CHECKOK(group->meth->field_sqr(&t, &r, group->meth));
        MP_CHECKOK(group->meth->field_mul(&u1, &u2, &w, group->meth));
        MP_CHECKOK(group->meth->field_sub(&r, &w, &r, group->meth));
54      MP_CHECKOK(group->meth->field_sqr(&z, &w, group->meth));
        MP_CHECKOK(group->meth->field_mul(&group->curvea, &w, &w, group->
            meth));
        MP_CHECKOK(group->meth->field_add(&r, &w, &r, group->meth));
        MP_CHECKOK(group->meth->field_mul(&z, &n, &w, group->meth));
        MP_CHECKOK(group->meth->field_add(&r, &w, &r, group->meth));
59      /* w = r^2 - g */
        MP_CHECKOK(group->meth->field_sqr(&r, &w, group->meth));
        MP_CHECKOK(group->meth->field_sub(&w, &g, &w, group->meth));
        /* x3 = 2 * f * w */
        MP_CHECKOK(group->meth->field_mul(&f, &w, rx, group->meth));
64      MP_CHECKOK(group->meth->field_add(rx, rx, rx, group->meth));
        /* y3 = r * (g - 2 * w) - l * f * m */
        MP_CHECKOK(group->meth->field_sub(&g, &w, ry, group->meth));
        MP_CHECKOK(group->meth->field_sub(ry, &w, ry, group->meth));
        MP_CHECKOK(group->meth->field_mul(&r, ry, ry, group->meth));
69      MP_CHECKOK(group->meth->field_mul(&l, &f, rz, group->meth));
        MP_CHECKOK(group->meth->field_mul(rz, &m, rz, group->meth));
        MP_CHECKOK(group->meth->field_sub(ry, rz, ry, group->meth));
        /* z3 = 2 * f^3 */
        MP_CHECKOK(group->meth->field_sqr(&f, rz, group->meth));
74      MP_CHECKOK(group->meth->field_mul(&f, rz, rz, group->meth));
        MP_CHECKOK(group->meth->field_add(rz, rz, rz, group->meth));

    CLEANUP:
        mp_clear(&u1); mp_clear(&u2); mp_clear(&t);
79      mp_clear(&v); mp_clear(&s1); mp_clear(&s2);
        mp_clear(&m); mp_clear(&n); mp_clear(&z);
        mp_clear(&f); mp_clear(&e); mp_clear(&l);
        mp_clear(&g); mp_clear(&r); mp_clear(&w);
```

```
    return res;
}

mp_err ec_GFp_pt_mul_unif_bdj04_proj(const mp_int *n, const mp_int *px
    , const mp_int *py, mp_int *rx, mp_int *ry, const ECGroup *group) {
    mp_err res = MP_OKAY;
    mp_int k, qx, qy, qz, sx, sy, sz;
    int i, l;
#ifdef ECL_HRTIME
    M_Time_DeclareVariables
#endif

    MP_DIGITS(&k) = 0; MP_DIGITS(&qx) = 0; MP_DIGITS(&qy) = 0;
    MP_DIGITS(&qz) = 0; MP_DIGITS(&sx) = 0; MP_DIGITS(&sy) = 0;
    MP_DIGITS(&sz) = 0;
    MP_CHECKOK(mp_init(&k)); MP_CHECKOK(mp_init(&qx));
    MP_CHECKOK(mp_init(&qy)); MP_CHECKOK(mp_init(&qz));
    MP_CHECKOK(mp_init(&sx)); MP_CHECKOK(mp_init(&sy));
    MP_CHECKOK(mp_init(&sz));

    /* if n = 0 then r = inf */
    if (mp_cmp_z(n) == 0) {
        mp_zero(rx);
        mp_zero(ry);
        res = MP_OKAY;
        goto CLEANUP;
    }
    /* Q = P, k = n */
    MP_CHECKOK(ec_GFp_pt_aff2proj(px, py, &qx, &qy, &qz, group));
    MP_CHECKOK(mp_copy(n, &k));
    /* if n < 0 then Q = -Q, k = -k */
    if (mp_cmp_z(n) < 0) {
        MP_CHECKOK(group->meth->field_neg(py, &qy, group->meth));
        MP_CHECKOK(ec_GFp_pt_aff2proj(px, &qy, &qx, &qy, &qz, group));
        MP_CHECKOK(mp_neg(&k, &k));
    }
    l = mpl_significant_bits(&k) - 1;
    MP_CHECKOK(mp_copy(&qx, &sx));
    MP_CHECKOK(mp_copy(&qy, &sy));
    MP_CHECKOK(mp_copy(&qz, &sz));
    for (i = l - 1; i >= 0; i--) {
        /* S = 2S */
#ifdef ECL_HRTIME
        M_Time_Start
#endif
        MP_CHECKOK(ec_GFp_pt_add_unif_bdj04_proj(&sx, &sy, &sz, &sx, &sy
            , &sz, &sx, &sy, &sz, group));
#ifdef ECL_HRTIME
        M_Time_Stop
        printf("d: %llu\n", M_Time_Difference);
#endif
        /* if k_i = 1, then S = S + Q */
        if (mpl_get_bit(&k, i) != 0) {
#ifdef ECL_HRTIME
            M_Time_Start
#endif
            MP_CHECKOK(ec_GFp_pt_add_unif_bdj04_proj(&sx, &sy, &sz, &qx,
                &qy, &qz, &sx, &sy, &sz, group));
#ifdef ECL_HRTIME
            M_Time_Stop
```

```
            printf("a: %llu\n", M_Time_Difference);
#endif
        }
    }
    /* output S */
    MP_CHECKOK(ec_GFp_pt_proj2aff(&sx, &sy, &sz, rx, ry, group));

  CLEANUP:
    mp_clear(&k); mp_clear(&qx); mp_clear(&qy);
    mp_clear(&qz); mp_clear(&sx); mp_clear(&sy);
    mp_clear(&sz);
    return res;
}
```

A.1.4 Timing (Section 6.3.4)

A.1.4.1 `hrtime.h`

The following block of C preprocessor code provides operations that can be used to time point operations. When used on Solaris, the code provides high-resolution timing (up to 100 ns).

```
#include <time.h>
#include <sys/time.h>
#include <sys/resource.h>

#define M_Time_UseHR
#define ECL_HRTIME

#ifdef M_Time_UseHR
#define M_Time_DeclareVariables M_Time_DeclareVariablesHR
#define M_Time_Start M_Time_StartHR
#define M_Time_Stop M_Time_StopHR
#define M_Time_Print M_Time_PrintHR
#define M_Time_Difference M_Time_DifferenceHR
#else
#define M_Time_DeclareVariables M_Time_DeclareVariablesBasic
#define M_Time_Start M_Time_StartBasic
#define M_Time_Stop M_Time_StopBasic
#define M_Time_Print M_Time_PrintBasic
#define M_Time_Difference M_Time_DifferenceBasic
#endif

#define M_Time_DeclareVariablesBasic \
    double M_Time_dStart, M_Time_dNow; \
    struct rusage M_Time_ru; \
    int M_Time_i;
#define M_Time_StartBasic \
    getrusage(RUSAGE_SELF, &M_Time_ru); \
    M_Time_dStart = (double)M_Time_ru.ru_utime.tv_sec+(double)M_Time_ru
        .ru_utime.tv_usec*0.000001;
#define M_Time_StopBasic \
    getrusage(RUSAGE_SELF, &M_Time_ru); \
    M_Time_dNow = (double)M_Time_ru.ru_utime.tv_sec+(double)M_Time_ru.
        ru_utime.tv_usec*0.000001;
#define M_Time_DifferenceBasic M_Time_dNow - M_Time_dStart
#define M_Time_PrintBasic \
```

```
    printf("t: %6.2f sec\n", M_Time_DifferenceBasic);

#define M_Time_DeclareVariablesHR \
    hrtime_t M_Time_hrStart, M_Time_hrStop; \
    int M_Time_i;
#define M_Time_StartHR \
    M_Time_hrStart = gethrtime();
#define M_Time_StopHR \
    M_Time_hrStop = gethrtime();
#define M_Time_DifferenceHR M_Time_hrStop - M_Time_hrStart
#define M_Time_PrintHR \
    printf("t: %llu nsec\n", M_Time_DifferenceHR);
```

A.1.4.2 ecp_test.c

The following block of C code times various point multiplication formulæ.

```
#include "mpi.h"
#include "mplogic.h"
#include "mpprime.h"
#include "ecl.h"
#include "ecl-curve.h"
#include "ecp.h"
#include <stdio.h>
#include <strings.h>
#include <assert.h>

#include <time.h>
#include <sys/time.h>
#include <sys/resource.h>

#define M_TimeOperation(op, k) { \
    double dStart, dNow, dUserTime; \
    struct rusage ru; \
    int i; \
    getrusage(RUSAGE_SELF, &ru); \
    dStart = (double)ru.ru_utime.tv_sec+(double)ru.ru_utime.tv_usec
        *0.000001; \
    for (i = 0; i < k; i++) { \
       { op; } \
    }; \
    getrusage(RUSAGE_SELF, &ru); \
    dNow = (double)ru.ru_utime.tv_sec+(double)ru.ru_utime.tv_usec
        *0.000001; \
    dUserTime = dNow-dStart; \
    if (dUserTime) printf("    %-45s k: %6i, t: %6.2f sec\n", #op, k,
        dUserTime); \
}

int ectest_curve_GFp(ECGroup *group) {

    mp_int gx, gy, rx, ry, n;
    int size;
    mp_err res;

    /* initialize values */
    MP_CHECKOK(mp_init(&gx)); MP_CHECKOK(mp_init(&gy));
    MP_CHECKOK(mp_init(&rx)); MP_CHECKOK(mp_init(&ry));
```

```
    MP_CHECKOK(mp_init(&n));

    MP_CHECKOK(mp_set_int(&one, 1));
    MP_CHECKOK(mp_sub(&group->order, &one, &order_1));

    /* encode base point */
    if (group->meth->field_dec) {
        MP_CHECKOK(group->meth->field_dec(&group->genx, &gx, group->meth
            ));
        MP_CHECKOK(group->meth->field_dec(&group->geny, &gy, group->meth
            ));
    } else {
        MP_CHECKOK(mp_copy(&group->genx, &gx));
        MP_CHECKOK(mp_copy(&group->geny, &gy));
    }

    /* compute random scalar */
    size = mpl_significant_bits(&group->meth->irr);
    if (size < MP_OKAY) {
        goto CLEANUP;
    }
    MP_CHECKOK(mpp_random_size(&n, (size + ECL_BITS - 1) / ECL_BITS));
    MP_CHECKOK(group->meth->field_mod(&n, &n, group->meth));
    /* timed test */
    M_TimeOperation(MP_CHECKOK (ec_GFp_pt_mul_aff(&n, &group->genx, &
        group->geny, &rx, &ry, group)), 100);
    M_TimeOperation(MP_CHECKOK (ec_GFp_pt_mul_unif_bj02_proj(&n, &group
        ->genx, &group->geny, &rx, &ry, group)), 100);
    M_TimeOperation(MP_CHECKOK (ec_GFp_pt_mul_unif_bdj04_aff(&n, &group
        ->genx, &group->geny, &rx, &ry, group)), 100);
    M_TimeOperation(MP_CHECKOK (ec_GFp_pt_mul_unif_bdj04_proj(&n, &
        group->genx, &group->geny, &rx, &ry, group)), 100);
    M_TimeOperation(MP_CHECKOK (ECPoint_mul(group, &n, NULL, NULL, &rx,
         &ry)), 100);
    M_TimeOperation(MP_CHECKOK (ECPoints_mul (group, &n, &n, &gx, &gy,
        &rx, &ry)), 100);

  CLEANUP:
    mp_clear(&gx); mp_clear(&gy);
    mp_clear(&rx); mp_clear(&ry); mp_clear(&n);
    if (res != MP_OKAY) {
        printf("  Error: exiting with error value %i\n", res);
    }
    return res;
}

int main(int argv, char **argc) {

    int i;
    ECGroup *group = NULL;
    ECCurveParams *params = NULL;
    mp_err res = MP_OKAY;

    params = EC_GetNamedCurveParams(ECCurve_SECG_PRIME_160R1);
    if (params == NULL) {
          res = MP_NO;
          goto CLEANUP;
    }
    ECGroup_free(group);
    group = ECGroup_fromHex(params);
```

```
    if (group == NULL) {
        res = MP_NO;
        goto CLEANUP;
    }
    MP_CHECKOK( ectest_curve_GFp(group) );
    printf("... okay.\n");

  CLEANUP:
    EC_FreeCurveParams(params);
    ECGroup_free(group);
    if (res != MP_OKAY) {
        printf("Error: exiting with error value %i\n", res);
    }
    return res;
}
```

A.1.5 Binary projective unified formula of Brier, Déchène, and Joye (Section 6.4)

A.1.5.1 Formula verification in Maple

The following block of Maple code confirms that the projective unified point addition formula for binary curves given in equation (6.34) mathematically corresponds to the affine formula of Brier, Déchène, and Joye [BDJ04] in equation (6.33).

```
restart;
U1:=X1*Z2; U2:=X2*Z1; S1:=Y1*Z2; S2:=Y2*Z1; Z:=Z1*Z2;
T:=U1+U2; M:=S1+S2;
E1:=U1+M; F1:=Z*E1; L1:=F1*E1; G1:=L1*T; H1:=F1^2;
R:=T^2+U1*U2+a*Z*T+Z*S2;
K1:=F1*R+G1+a*H1; W1:=R^2+K1;
Z3:=F1*H1; X3:=F1*W1; Y3:=R*(L1*U1+W1)+X3+H1*E1*S1;
lambda:=((x1+x2)^2+x1*x2+a*(x1+x2)+y2)/(x1+y1+y2);
x3:=lambda^2+lambda+x1+x2+a;
y3:=lambda*(x1+x3)+x3+y1;
X1:=x1*Z1; Y1:=y1*Z1; X2:=x2*Z2; Y2:=y2*Z2;
simplify(X3-x3*Z3)=0;
simplify(Y3-y3*Z3)=0;
```

Colophon

This thesis was typeset using LaTeX. The bibliography was typeset from a BibTeX database using a custom bibliography stylesheet called `halphads`, based on the `halpha` stylesheet and modified to included links to electronic resources identified by Digital Object Identifiers (DOIs) and e-prints.

References

[Aar05] Scott Aaronson. Quantum copy-protection. Private correspondence, 2005.

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In preparation, 2009.

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula Rao, and Pankaj Rohatgi. The EM side-channel(s). In B. S. Kaliski, Jr., Ç. K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES) 2002*, *LNCS*, volume 2523, pp. 29–45. Springer, 2002. DOI:10.1007/3-540-36400-5_4.

[ABB+04] William Aiello, Steven M. Bellovin, Matt Blaze, Ran Canetti, John Ioannidis, Angelos D. Keromytis, and Omer Reingold. Just Fast Keying: Key agreement in a hostile Internet. *ACM Transactions on Information and System Security*, **7**(2):1–30, May 2004. DOI:10.1145/996943.996946.

[ABB+07] Romain Alléaume, Jan Bouda, Cyril Branciard, Thierry Debuisschert, Mehrdad Dianati, Nicolas Gisin, Mark Godfrey, Philippe Grangier, Thomas Länger, Anthony Leverrier, Norbert Lütkenhaus, Philippe Painchault, Momtchil Peev, Andreas Poppe, Thomas Pornin, John Rarity, Renato Renner, Grégoire Ribordy, Michel Riguidel, Louis Salvail, Andrew Shields, Harald Weinfurter, and Anton Zeilinger. SECOQC white paper on quantum key distribution and cryptography, January 2007. EPRINT arXiv:quant-ph/0701168.

[ABC+06] Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, Bodo Möller, and David Pointcheval. Provably secure password-based authentication in TLS. In Shiuhpyng Shieh and Sushil Jajodia, editors, *Proc. 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS'06)*, pp. 35–45. ACM Press, 2006. DOI:10.1145/1128817.1128827.

[ABG+07] Antonio Acin, Nicolas Brunner, Nicolas Gisin, Serge Massar, Stefano Pironio, and Valerio Scarani. Device-independent security of quantum cryptography against collective attacks. *Physical Review Letters*, **98**(23):230501, 2007. DOI:10.1103/PhysRevLett.98.230501. EPRINT arXiv:quant-ph/0702152.

[ABMW03] Martín Abadi, Michael Burrows, Mark Manasse, and Ted Wobber. Moderately hard, memory-bound functions. In *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 2003*. Internet

Society, 2003. URL http://www.isoc.org/isoc/conferences/ndss/03/proceedings/.

[ABOE08] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations, October 2008. EPRINT arXiv:0810.5375.

[ACP05a] Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange. In Vaudenay [Vau05], pp. 47–64. DOI:10.1007/b105124. Full version available as [ACP05b].

[ACP05b] Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange, 2005. URL http://www.di.ens.fr/~mabdalla/papers/ACP05-letter.pdf. Extended abstract published as [ACP05a].

[AE07] Andris Ambainis and Joseph Emerson. Quantum t-designs: t-wise independence in the quantum world. In *Proc. 22nd Ann. IEEE Conference on Computational Complexity (CCC) 2007*, pp. 129–140. IEEE, June 2007. DOI:10.1109/CCC.2007.26. EPRINT arXiv:quant-ph/0701126.

[AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Vaudenay [Vau05], pp. 65–84. DOI:10.1007/b105124.

[AGLL96] Derek Atkins, Michael Graff, Arjen K. Lenstra, and Paul C. Leyland. The magic words are squeamish ossifrage (extended abstract). In Kim and Matsumoto [KM96], pp. 265–277. DOI:10.1007/BFb0034829.

[AJK04] Joseph B. Altepeter, Daniel F. V. James, and Paul G. Kwiat. Qubit quantum state tomography. In Matteo Paris and Jaroslav Řeháček, editors, *Quantum State Estimation*, *Lecture Notes in Physics*, volume 649, pp. 113–145. Springer, 2004. DOI:10.1007/b98673.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, **22**(1):6–15, January 1996. DOI:10.1109/32.481513.

[AN97] Tuomas Aura and Pekka Nikander. Stateless connections. In Yongei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Proc. 1st International Conference on Information and Communications Security*, *LNCS*, volume 1334, pp. 87–97. Springer, November 1997. DOI:10.1007/BFb0028465.

[ANL00] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DOS-resistant authentication with client puzzles. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *SECPROT*, *LNCS*, volume 2133, pp. 170–177. Springer, 2000. DOI:10.1007/3-540-44810-1_22. URL http://research.microsoft.com/en-us/um/people/tuomaura/Publications/aura-nikander-leiwo-protocols00.pdf.

[Art91] Michael Artin. *Algebra*. Prentice-Hall, 1991.

[Atk95] Randall Atkinson. Security architecture for the Internet Protocol, August 1995. URL http://www.ietf.org/rfc/rfc1825.txt. RFC 1825.

[Bac97] Adam Back. A partial hash collision based postage scheme, 1997. URL http://www.hashcash.org/papers/announce.txt.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Computing*, **26**(5):1510–1523, 1997. DOI:10.1137/S0097539796300933. EPRINT arXiv:quant-ph/9701001.

[BBBW82] Charles H. Bennett, Gilles Brassard, Seth Breidbard, and Stephen Wiesner. Quantum cryptography, or unforgeable subway tokens. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – Proc. CRYPTO '82*. Plenum Press, 1982.

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors. *Post Quantum Cryptography*. Springer, 2009.

[BBF+07] Gilles Brassard, Anne Broadbent, Joseph Fitzsimons, Sébastien Gambs, and Alain Tapp. Anonymous quantum communication. In Kurosawa [Kur07], pp. 460–473. DOI:10.1007/978-3-540-76900-2_28. EPRINT arXiv:0706.2356.

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, **46**(4–5):493–505, 1998. DOI:10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. EPRINT arXiv:quant-ph/9605034.

[BCC+06] Steve Babbage, Dario Catalano, Carlos Cid, Louis Granboulan, Tanja Lange, Arjen Lenstra, Phong Nguyen, Christof Paar, Jan Pelzl, Thomas Pornin, Bart Preneel, Matt Robshaw, Andy Rupp, Nigel Smart, and Michael Ward. ECRYPT yearly report on algorithms and keysizes (2005), January 2006. URL http://www.ecrypt.eu.org/documents/D.SPA.16-1.0.pdf.

[BCC+08] Steve Babbage, Dario Catalano, Carlos Cid, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tanja Lange, Arjen Lenstra, Phong Nguyen, Christof Paar, Jan Pelzl, Thomas Pornin, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Andy Rupp, Nigel Smart, and Michael Ward. ECRYPT yearly report on algorithms and keysizes (2007–2008), July 2008. URL http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf.

[BDCZ98] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett.*, **81**(26):5932–5935, December 1998. DOI:10.1103/PhysRevLett.81.5932. EPRINT arXiv:quant-ph/9803056.

[BDJ04] Éric Brier, Isabelle Déchène, and Marc Joye. Unified point addition formulæ for elliptic curve cryptosystems. In N. Nedjah and L. de Macedo Mourelle, editors, *Embedded Cryptographic Hardware: Methodologies and Architectures*, pp. 247–256. Nova Science Publishers, 2004.

[BFK08] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation, 2008. eprint arXiv:0807.4154.

[BGI+01a] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs, 2001. url `http://www.wisdom.weizmann.ac.il/~oded/p_obfuscate.html`. Published as [BGI+01b].

[BGI+01b] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – Proc. CRYPTO 2001*, *LNCS*, volume 2139, pp. 1–18. Springer, 2001. doi:10.1007/3-540-44647-8_1. eprint `http://eprint.iacr.org/2001/069`. Full version available as [BGI+01a].

[BHHM01] Michael Brown, Darrel Hankerson, Julio López Hernandez, and Alfred J. Menezes. Software implementation of the NIST elliptic curves over prime fields. In David Naccache, editor, *Progress in Cryptography – CT-RSA 2001: The Cryptographers' Track at RSA Conference 2001*, *LNCS*, volume 2020, pp. 250–265. Springer, 2001. doi:10.1007/3-540-45353-9_19.

[BHK05] Jonathan Barrett, Lucien Hardy, and Adrian Kent. No signaling and quantum key distribution. *Physical Review Letters*, **95**(1):010503, 2005. doi:10.1103/PhysRevLett.95.010503. eprint arXiv:quant-ph/0405101.

[BJ02] Éric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In Y. G. Desmedt, editor, *Public Key Cryptography (PKC) 2002*, *LNCS*, volume 2274, pp. 87–100. Springer, 2002. doi:10.1007/3-540-45664-3_24.

[BK07] Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators (revised), March 2007. url `http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf`. NIST Special Publication 800-90.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *Advances in Cryptology – Proc. ASIACRYPT 2001*, *LNCS*, volume 2248, pp. 514–532. Springer, 2001. doi:10.1007/3-540-45682-1_30.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the 1992 IEEE Computer Society Conference on Research in Security and Privacy*, May 1992. doi:10.1109/RISP.1992.213269. url `http://www.alw.nih.gov/Security/FIRST/papers/crypto/neke.ps`.

[BM94] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise. Technical report, AT&T Bell Laboratories, c. 1994. url `http://www.alw.nih.gov/Security/FIRST/papers/crypto/aeke.ps`.

[BM03] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[BM07] Dagmar Bruß and Chiara Macchiavello. Approximate quantum cloning. In Dagmar Bruß and Gerd Leuchs, editors, *Lectures on Quantum Information*. Wiley-VCH, 2007.

[BMP00a] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure Password-Authenticated Key exchange using Diffie-Hellman. In Preneel [Pre00], pp. 156–171. DOI:10.1007/3-540-45539-6_12. Full version available as [BMP00b].

[BMP00b] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure Password-Authenticated Key exchange using Diffie-Hellman, 2000. EPRINT `http://eprint.iacr.org/2000/044`. Short version published as [BMP00a].

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In J. P. Buhler, editor, *Proc. Analytic Number Theory 3rd International Symposium (ANTS) '98*, *LNCS*, volume 1423, pp. 48–63. Springer, 1998. DOI:10.1007/BFb0054851. EPRINT `http://crypto.stanford.edu/~dabo/abstracts/DDH.html`.

[Bon03] Dan Boneh, editor. *Advances in Cryptology – Proc. CRYPTO 2003*, *LNCS*, volume 2729. Springer, 2003. DOI:10.1007/b11817.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matt Franklin, editor, *Advances in Cryptology – Proc. CRYPTO 2004*, *LNCS*, volume 3152, pp. 273–289, 2004. DOI:10.1007/b99099.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Preneel [Pre00], pp. 139–155. DOI:10.1007/3-540-45539-6_11.

[BR93a] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution, 1993. URL `http://www-cse.ucsd.edu/~mihir/papers/key-distribution.html`. Extended abstract published as [BR93b].

[BR93b] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – Proc. CRYPTO '93*, *LNCS*, volume 773, pp. 232–249. Springer, 1993. DOI:10.1007/3-540-48329-2_21. Full version available as [BR93a].

[BR95] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *Proc. 27th Ann. ACM Symp. on the Theory of Computing (STOC)*, pp. 57–66. ACM Press, 1995. DOI:10.1145/225058.225084. URL `http://www-cse.ucsd.edu/~mihir/papers/3pkd.pdf`.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures – how to sign with RSA and Rabin. In Maurer [Mau96], pp. 399–416. DOI:10.1007/3-540-68339-9_34.

[BR04] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In Pfitzmann and Liu [PL04], pp. 62–73. DOI:10.1145/168588.168596.

[Bra06] Gilles Brassard. Brief history of quantum cryptography: A personal perspective, 2006. EPRINT arXiv:quant-ph/0604072.

[Bro04] Daniel R. L. Brown. Generic groups, collision resistance, and ECDSA. *Designs, Codes and Cryptography*, **35**(1):119–152, April 2004. DOI:10.1007/s10623-003-6154-z. EPRINT `http://eprint.iacr.org/2002/026`.

[BS08] Travis R. Beals and Barry C. Sanders. Distributed relay protocol for probabilistic information-theoretic security in a randomly-compromised network. In Reihaneh Safavi-Naini, editor, *Third International Conference on Information Theoretic Security (ICITS) 2008*, *LNCS*, volume 5155, pp. 29–39. Springer, 2008. DOI:10.1007/978-3-540-85093-9_4. EPRINT arXiv:0803.2919.

[BT07] Anne Broadbent and Alain Tapp. Information-theoretic security without an honest majority. In Kurosawa [Kur07], pp. 410–426. DOI:10.1007/978-3-540-76900-2_25. EPRINT arXiv:0706.2010.

[Bus03] George W. Bush. Executive Order 13292. further amendment to Executive Order 12958, as amended, Classified National Security Information, March 2003. URL `http://www.archives.gov/isoo/policy-documents/eo-12958-amendment.pdf`.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *Cryptography and Coding – 6th IMA International Conference*, *LNCS*, volume 1355. Springer, 1997. DOI:10.1007/BFb0024447.

[CC86] D. V. Chudnovsky and G. V. Chudnovsky. Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics*, **7**(4):385–434, December 1986. DOI:10.1016/0196-8858(86)90023-0.

[CEMM98] Richard Cleve, Artur Ekert, Chiara Macchiavello, and Michele Mosca. Quantum algorithms revisited. *Proc. Royal Society London A*, **454**(1969):339–354, January 1998. DOI:10.1098/rspa.1998.0164. EPRINT arXiv:quant-ph/9708016.

[Cer00] Certicom Research. SEC 2: Recommended elliptic curve domain parameters, 2000. URL `http://www.secg.org/`.

[CFN88] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash (extended abstract). In Goldwasser [Gol88], pp. 319–327. DOI:10.1007/0-387-34799-2_25.

[Cha85] David Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, **28**(10):1030–1044, October 1985. DOI:10.1145/4372.4373.

[Cha88] David Chaum. Privacy protected payments: Unconditional payer and/or payee untracability. In *Smartcard 2000*. North Holland, 1988.

[Chi05] Andrew Childs. Secure assisted quantum computation. *Quantum Information and Computation*, **5**(6):456–466, September 2005. EPRINT arXiv:quant-ph/0111046, URL `http://www.rinton.net/xqic5/qic-5-6/456-466.pdf`.

[CHK+05] Ron Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – Proc. EUROCRYPT 2005*, *LNCS*, volume 3494, pp. 404–421. Springer, 2005. DOI:10.1007/11426639_24.

[CK01a] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – Proc. EUROCRYPT 2001*, *LNCS*, volume 2045, pp. 453–474. Springer, 2001. DOI:10.1007/3-540-44987-6_28. Full version available as [CK01b].

[CK01b] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels, 2001. EPRINT `http://eprint.iacr.org/2001/040`. Extended abstract published as [CK01a].

[CMMP08] Donny Cheung, Dmitri Maslov, Jimson Mathew, and Dhiraj K. Pradhan. On the design and optimization of a quantum polynomial-time attack on elliptic curve cryptography. In Yasuhito Kawano and Michele Mosca, editors, *Theory of Quantum Computation, Communication, and Cryptography (TQC) 2008*, *LNCS*, volume 5106, pp. 96–104. Springer, 2008. DOI:10.1007/978-3-540-89304-2_9. EPRINT arXiv:0710.1093.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Koç and Paar [KP99], pp. 292–302. DOI:10.1007/3-540-48059-5.

[CP92] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In Rainer A. Rueppel, editor, *Advances in Cryptology – Proc. EUROCRYPT '92*, *LNCS*, volume 658, pp. 390–407. Springer-Verlag, 1992. DOI:10.1007/3-540-47555-9_32.

[CPS08a] Jean-Sebastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *Advances in Cryptology – Proc. CRYPTO 2008*, *LNCS*, volume 5157, pp. 1–20. Springer, 2008. DOI:10.1007/978-3-540-85174-5_1. Full version available as [CPS08b].

[CPS08b] Jean-Sebastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent, 2008. EPRINT `http://eprint.iacr.org/2008/246`. Extended abstract published as [CPS08a].

[CS08] Raymond Y. Q. Cai and Valerio Scarani. Finite-key analysis for practical implementations of quantum key distribution, November 2008. EPRINT arXiv:0811.2628.

[dB90] Bert den Boer. Diffie-Hellman is as strong as discrete log for certain primes. In Goldwasser [Gol88], pp. 530–539. DOI:10.1007/0-387-34799-2_38.

[DGN03] Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In Boneh [Bon03], pp. 426–444. DOI:10.1007/b11817.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, **22**(6):644–654, November 1976.

[Die82] D. Dieks. Communication by EPR devices. *Phys. Lett. A*, **92**(6):271–272, November 1982. DOI:10.1016/0375-9601(82)90084-6.

[DMR06] Sujata Doshi, Fabian Monrose, and Aviel D. Rubin. Efficient memory bound puzzles using pattern databases. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security (ACNS) 2006*, *LNCS*, volume 3989, pp. 98–113. Springer, 2006. DOI:10.1007/11767480_7.

[DN92] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *Advances in Cryptology – Proc. CRYPTO '92*, *LNCS*, volume 740, pp. 139–147. Springer, 1992. DOI:10.1007/3-540-48071-4_10.

[DR06] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) protocol version 1.1, April 2006. URL `http://www.ietf.org/rfc/rfc4346.txt`. RFC 4346.

[DvOW92] Whitfield Diffie, Paul van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, **2**(2):107–125, June 1992. DOI:10.1007/BF00124891.

[ECP+05] Chip Elliott, Alexander Colvin, David Pearson, Oleksiy Pikalo, John Schlafer, and Henry Yeh. Current status of the DARPA quantum network, 2005. EPRINT arXiv:quant-ph/0503058.

[Edd07] Wesley M. Eddy. TCP SYN flooding attacks and common mitigations, August 2007. URL `http://www.ietf.org/rfc/rfc4987.txt`. RFC 4987.

[FMR99] Gerhard Frey, M. Muller, and H. Ruck. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Information Theory*, **45**(5):1717–1719, 1999. DOI:10.1109/18.771254.

[Gar77] Martin Gardner. Mathematical games: A new kind of cipher that would take millions of years to break. *Scientific American*, pp. 120–124, August 1977.

[GB08] Damien Giry and Philippe Bulens. Keylength.com – cryptographyic key length recommendation, 2008. URL `http://www.keylength.com/`.

[GLLP04] Daniel Gottesman, Hoi-Kwong Lo, Norbert Lütkenhaus, and John Preskill. Security of quantum key distribution with imperfect devices. *Quantum Information and Computation*, **4**(5):325–360, September 2004. EPRINT arXiv:quant-ph/0212066, URL `http://www.rinton.net/xqic4/qic-4-5/325-360.pdf`.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, **17**(2):281–308, April 1988. DOI:10.1137/0217017.

[GMR05] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. PAK-Z+, August 2005. URL `http://grouper.ieee.org/groups/1363/WorkingGroup/presentations/pakzplusv2.pdf`. Contribution to the IEEE P1363-2000 study group for Future PKC Standards.

[GMR06] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, *Advances in Cryptology – Proc. CRYPTO 2006*, *LNCS*, volume 4117, pp. 142–159. Springer, 2006. DOI:10.1007/11818175_9.

[GMT45] Jack Good, Donald Michie, and Geoffrey Timms. General report on tunny. Technical report, Government Code and Cypher School, 1945. URL `http://www.alanturing.net/turing_archive/archive/index/tunnyreportindex.html`. Declassified September 28, 2000, by Pulic Records Office, UK, documents HW 25/4 and HW 25/5.

[Gol88] Shafi Goldwasser, editor. *Advances in Cryptology – Proc. CRYPTO '88*, *LNCS*, volume 403. Springer, 1988. DOI:10.1007/0-387-34799-2.

[Gol06] Oded Goldreich. On post-modern cryptography, 2006. EPRINT `http://eprint.iacr.org/2006/461`.

[Gor93] Daniel M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Mathematics*, **6**(1):124–138, February 1993. URL `http://www.ccrwest.org/gordon/log.pdf`.

[GS95] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pp. 44–55, September 1995. URL `http://citeseer.ist.psu.edu/article/gong94failstop.html`.

[HC98] Dan Harkins and Dave Carrel. The Internet Key Exchange (IKE), November 1998. URL `http://www.ietf.org/rfc/rfc2409.txt`. RFC 2409.

[HHM00] Darrel Hankerson, Julio López Hernandez, and Alfred J. Menezes. Software implementation of elliptic curve cryptography over binary fields. In Ç. K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES) 2000*, *LNCS*, volume 1965, pp. 1–24. Springer, 2000. DOI:10.1007/3-540-44499-8_1.

[HLM03] Patrick Hayden, Debbie Leung, and Dominic Mayers. Composability of authentication in QKD. Private communication, 2003.

[HMV04] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[HRP+06] P. A. Hiskett, D. Rosenberg, C. G. Peterson, R. J. Hughes, S. Nam, A. E. Lita, A. J. Miller, and J. E. Nordholt. Long-distance quantum key distribution in optical fibre. *New Journal of Physics*, **8**(9):193, 2006. DOI:10.1088/1367-2630/8/9/193. EPRINT arXiv:quant-ph/0607177.

[Hwa03] Won-Young Hwang. Quantum key distribution with high loss: Toward global secure communication. *Phys. Rev. Lett.*, **91**(5):057901, 2003. DOI:10.1103/PhysRevLett.91.057901. EPRINT arXiv:quant-ph/0211153.

[IT03] Tetsuya Izu and Tsuyoshi Takagi. Exceptional procedure attack on elliptic curve cryptosystems. In Y. G. Desmedt, editor, *Public Key Cryptography (PKC) 2003*, *LNCS*, volume 2567, pp. 224–239. Springer, 2003. DOI:10.1007/3-540-36288-6_17.

[Jab08] David P. Jablon. Research papers on password-based cryptography, 2008. URL `http://www.jablon.org/passwordlinks.html`.

[JB99] Ari Juels and John Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 1999* [NDS99], pp. 151–165.

[JJ99] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In Bart Preneel, editor, *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*, *IFIP Conference Proceedings*, volume 152, pp. 258–272. Kluwer, 1999. URL `http://www.rsa.com/rsalabs/node.asp?id=2049`.

[JN01] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups, January 2001. EPRINT `http://eprint.iacr.org/2001/003`.

[JV96] Mike Just and Serge Vaudenay. Authenticated multi-party key agreement. In Kim and Matsumoto [KM96], pp. 36–49. DOI:10.1007/BFb0034833.

[Kau05] Charlie Kaufman. Internet Key Exchange (IKEv2) protocol. Online, December 2005. URL `http://www.ietf.org/rfc/rfc4306.txt`. RFC 4306.

[Kit95] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem, 1995. EPRINT arXiv:quant-ph/9511026.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – Proc. CRYPTO '99*, *LNCS*, volume 1666, pp. 388–397. Springer, 1999. DOI:10.1007/3-540-48405-1_25.

[KM96] Kwangjo Kim and Tsutomu Matsumoto, editors. *Advances in Cryptology – Proc. ASIACRYPT '96*, *LNCS*, volume 1163. Springer, 1996. DOI:10.1007/BFb0034829.

[KM04] Neal Koblitz and Alfred J. Menezes. Another look at "provable security", 2004. EPRINT `http://eprint.iacr.org/2004/152`. Published as [KM07].

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, **20**(1):3–37, 2007. DOI:10.1007/s00145-005-0432-z. Earlier version appeared as [KM04].

[Koc96] P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – Proc. CRYPTO '96*, *LNCS*, volume 1109, pp. 104–113. Springer, 1996. DOI:10.1007/3-540-68697-5_9.

[KOY] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient and secure authenticated key exchange using weak passwords. URL `http://www.cs.umd.edu/~jkatz/papers/password.pdf`. Undated.

[KP99] Ç. K. Koç and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems (CHES) '99*, *LNCS*, volume 1717. Springer, 1999. DOI:10.1007/3-540-48059-5.

[Kra03a] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Boneh [Bon03], pp. 400–425. DOI:10.1007/b11817. Full version available as [Kra03b].

[Kra03b] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols, 2003. URL `http://www.ee.technion.ac.il/~hugo/sigma.ps`. Short version published as [Kra03a].

[Kra05a] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – Proc. CRYPTO 2005*, *LNCS*, volume 3621, pp. 546–566. Springer, 2005. DOI:10.1007/11535218_33. Full version available as [Kra05b].

[Kra05b] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol, 2005. EPRINT `http://eprint.iacr.org/2005/176.pdf`. Extended abstract published as [Kra05a].

[KS99] Phil Karn and William Allen Simpson. Photuris: Session-key management protocol. Online, March 1999. URL `http://www.ietf.org/rfc/rfc2522.txt`. RFC 2522.

[Kur07] Kaoru Kurosawa, editor. *Advances in Cryptology – Proc. ASIACRYPT 2007*, *LNCS*, volume 4833. Springer, 2007. DOI:10.1007/978-3-540-76900-2.

[KZ04] Phillip Kaye and Christof Zalka. Optimized quantum implementation of elliptic curve arithmetic over binary fields, July 2004. EPRINT arXiv:quant-ph/0407095.

[LC99] Hoi-Kwong Lo and H. F. Chau. Unconditional security of quantum key distribution over arbitrarily long distances. *Science*, **283**(5410):2050–2056, 1999. DOI:10.1126/science.283.5410.2050. EPRINT arXiv:quant-ph/9803006.

[LD99] Julio López and Ricardo Dahab. Fast multiplication on elliptic curves over $gf(2^m)$ without precomputation. In Koç and Paar [KP99], pp. 316–327. DOI:10.1007/3-540-48059-5_27.

[LLM07] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *First International Conference on Provable Security (ProvSec) 2007*, *LNCS*, volume 4784, pp. 1–16. Springer, 2007. DOI:10.1007/978-3-540-75670-5_1. EPRINT `http://eprint.iacr.org/2006/073`.

[LP08] Jooyoung Lee and Je Hong Park. Authenticated key exchange secure under the computational Diffie-Hellman assumption, August 2008. EPRINT `http://eprint.iacr.org/2008/344`.

[Lys08] Anna Lysyanskaya. Cryptography: How to keep your secrets safe. *Scientific American*, pp. 89–94, September 2008. URL `http://www.sciam.com/article.cfm?id=cryptography-how-to-keep-your-secrets-safe`.

[Mac01] Philip MacKenzie. On the security of the SPEKE password-authenticated key exchange protocol. Technical report, Bell Laboratories, Lucent Technologies, 2001. EPRINT `http://eprint.iacr.org/2001/057`.

[Mac02] Philip MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Technical Report 2002-46, DIMACS Center, Rutgers University, 2002. URL `http://dimacs.rutgers.edu/TechnicalReports/abstracts/2002/2002-46.html`.

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology – Proc. EUROCRYPT '96*, *LNCS*, volume 1070. Springer, 1996. DOI:10.1007/3-540-68339-9.

[May97] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, **78**(17):3414–3417, April 1997. EPRINT arXiv:quant-ph/9605044.

[Mea99] Catherine Meadows. A formal framework and evaluation method for network denial of service. In *Proc. 1999 IEEE Computer Security Foundations Workshop (CSFW)*, p. 4. IEEE Computer Society Press, 1999. DOI:10.1109/CSFW.1999.779758.

[MNJH04] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas R. Henderson. Host identity protocol. Online, February 2004. URL `http://tools.ietf.org/html/draft-moskowitz-hip-09`. Internet-Draft.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, **44**(170):519–521, 1985. URL `http://www.jstor.org/stable/2007970`.

[Mon87] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, **48**(177):243–264, January 1987.

[Moz08] Mozilla Foundation, The. Netscape Security Services (NSS) v3.9, 2008. URL `http://www.mozilla.org/projects/security/pki/nss/`.

[MP02] Wenbo Mao and Kenneth G. Paterson. On the plausible deniability feature of Internet protocols. Manuscript, 2002. URL `http://citeseer.ist.psu.edu/678290.html`.

[MS06] Michele Mosca and Douglas Stebila. Uncloneable quantum money. In *Canadian Quantum Information Students' Conference (CQISC) 2006*, Calgary, Alberta, August 2006. URL `http://www.iqis.org/events/cqisc06/papers/Mon-1130-Stebila.pdf`.

[MS07] Michele Mosca and Douglas Stebila. A framework for quantum money. In *Quantum Information Processing (QIP) 2007*, Brisbane, Australia, January 2007.

[MvOV01] Alfred J. Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, 5th edition, 2001. URL `http://www.cacr.math.uwaterloo.ca/hac/`.

[MW99] Ueli M. Maurer and Stefan Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, **28**(5):1689–1721, 1999. DOI:10.1137/S0097539796302749.

[MY98] Dominic Mayers and Andrew C. Yao. Quantum cryptography with imperfect apparatus. In *Proc. 39th Ann. IEEE Symp. Foundations of Comp. Sci.*, pp. 503–509. IEEE Press, 1998. DOI:10.1109/SFCS.1998.743501. EPRINT arXiv:quant-ph/9809039.

[Nat06] National Institute of Standards and Technology. Quantum information networks, 2006. URL `http://www.antd.nist.gov/qin/`.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[NDS99] *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 1999*. Internet Society, 1999. URL `http://www.isoc.org/isoc/conferences/ndss/99/proceedings/`.

[NIS07] NIST. Recommendations for key management – Part 1: General (revised), March 2007. URL `http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1_3-8-07.pdf`.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *Public Key Cryptography (PKC) 2000*, *LNCS*, volume 1992, pp. 104–118. Springer, 2001. DOI:10.1007/3-540-44586-2_8.

[Ope08] OpenSSL Project, The. OpenSSL v0.9.8, 2008. URL `http://www.openssl.org/`.

[Osw05] Elisabeth Oswald. Side-channel analysis. In Ian F. Blake, Gadiel Seroussi, and Nigel P. Smart, editors, *Advances in Elliptic Curve Cryptography*, pp. 69–86. Cambridge University Press, 2005.

[PAFdM+08] Josep Maria Perdigues Armengol, Bernhard Furch, Clovis Jacinto de Matos, Olivier Minster, Luigi Cacciapuoti, Martin Pfennigbauer, Markus Aspelmeyer, Thomas Jennewein, Rupert Ursin, Tobias Schmitt-Manderbach, Guy Baister, John Rarity, Walter Leeb, Cesare Barbieri, Harald Weinfurter, and Anton Zeilinger. Quantum communications at ESA: Towards a space experiment on the ISS. *Acta Astronautica*, **63**(1-4):165–178, 2008. DOI:10.1016/j.actaastro.2007.12.039.

[PL04] Birgit Pfitzmann and Peng Liu, editors. *Proc. 11th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2004.

[PNM+05] M. Peev, M. Nölle, O. Maurhardt, T. Lorünser, M. Suda, A. Poppe, R. Ursin, A. Fedrizzi, and A. Zeilinger. A novel protocol-authentication algorithm ruling out a man-in-the-middle attack in quantum cryptography. *International Journal of Quantum Information*, **3**(1):225–231, March 2005. DOI:10.1142/S0219749905000797. EPRINT arXiv:quant-ph/0407131.

[Pol78] J. M. Pollard. Monte carlo methods for index computation (mod p). *Mathematics of Computation*, **32**(143):918–924, 1978.

[PP04] Young Man Park and Sang Gyu Park. Two factor authenticated key exchange (TAKE) protocol in public wireless LANs. *IEICE Transactions on Communications*, **E87-B**(5):1382–1385, May 2004.

[PPS04] Kenneth G. Paterson, Fred Piper, and Rüdiger Schack. Why quantum cryptography?, June 2004. EPRINT arXiv:quant-ph/0406147.

[Pre00] Bart Preneel, editor. *Advances in Cryptology – Proc. EUROCRYPT 2000*, *LNCS*, volume 1807. Springer, 2000. DOI:10.1007/3-540-45539-6.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Maurer [Mau96], pp. 387–398. DOI:10.1007/3-540-68339-9_33.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, **13**(3):361–396, December 2000. DOI:10.1007/s001450010003.

[PZ03] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information and Computation*, **3**(4):317–344, July 2003. EPRINT arXiv:quant-ph/0301141, URL `http://www.rinton.net/xqic3/qic-3-4/317-344.ps`.

[PZ07] David Pointcheval and Sébastien Zimmer. Multi-factor authenticated key exchange. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security (ACNS) 2007*, *LNCS*, volume 4521, pp.

277–295. Springer, 2007. DOI:10.1007/978-3-540-72738-5. EPRINT ftp://ftp.di.ens.fr/pub/users/pointche/Papers/2008_acns.pdf.

[RSA] RSA Security Inc. RSA SecurID. URL http://www.rsa.com/node.aspx?id=1156.

[SBPC+08] Valerio Scarani, Helle Bechmann-Pasquinucci, Nicolas J. Cerf, Miloslav Dusek, Norbert Lütkenhaus, and Momtchil Peev. The security of practical quantum key distribution, 2008. EPRINT arXiv:0802.4155.

[Sch03] Bruce Schneier. Crypto-Gram: Quantum cryptography, December 2003. URL http://www.schneier.com/crypto-gram-0312.html#6.

[Sch07] Bruce Schneier. Schneier on Security: Switzerland protects its vote with quantum cryptography, October 2007. URL http://www.schneier.com/blog/archives/2007/10/switzerland_pro.html.

[Sch08] Bruce Schneier. Quantum cryptography: As awesome as it is pointless. *Wired*, October 2008. URL http://www.wired.com/politics/security/commentary/securitymatters/2008/10/securitymatters_1016.

[SDOF07] Stuart Schecter, Rachna Dhamija, Andy Ozment, and Ian Fischer. The emperor's new security indicators: An evaluation of website authentication and the effect of role playing on usability studies. In *Proc. IEEE Symposium on Security and Privacy (S&P) 2007*, pp. 51–65. IEEE Press, 2007. DOI:10.1109/SP.2007.35. EPRINT http://usablesecurity.org/emperor/.

[SGNB06] Jason Smith, Juan Gonzalez-Nieto, and Colin Boyd. Modelling denial of service attacks on JFK with Meadows's cost-based framework. In Rajkumar Buyya, Tianchi Ma, Reihaneh Safavi-Naini, Chris Steketee, and Willy Susilo, editors, *Proc. 4th Australasian Information Security Workshop – Network Security (AISW-NetSec) 2006*, *CRPIT*, volume 54, pp. 125–134. Australian Computer Society, 2006. URL http://crpit.com/confpapers/CRPITV54Smith.pdf.

[Sho94] Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proc. 35th Ann. IEEE Symp. Foundations of Comp. Sci.*, pp. 124–134. IEEE Press, 1994. DOI:10.1109/SFCS.1994.365700. EPRINT arXiv:quant-ph/9508027.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing*, **26**(5):1484–1509, 1997. DOI:10.1137/S0097539795293172.

[Sho99a] Victor Shoup. On formal models for secure key exchange. Report RZ 3120, IBM Research, April 1999.

[Sho99b] Victor Shoup. On formal models for secure key exchange (version 4), November 1999. URL http://shoup.net/papers/skey.pdf. Earlier version appeared as [Sho99a].

[Sho06] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs, 2006. URL http://www.shoup.net/papers/games.pdf. First version appear in 2004.

[Sho08] Shop.org/Forrester Research. Online sales to climb despite struggling economy, April 2008. URL http://www.shop.org/c/journal_articles/view_article_content?groupId=1&articleId=702&version=1.0.

[SMWF+07] Tobias Schmitt-Manderbach, Henning Weier, Martin Furst, Rupert Ursin, Felix Tiefenbacher, Thomas Scheidl, Josep Perdigues, Zoran Sodnik, Christian Kurtsiefer, John G. Rarity, Anton Zeilinger, and Harald Weinfurter. Experimental demonstration of free-space decoy-state quantum key distribution over 144 km. *Physical Review Letters*, **98**(1):010504, 2007. DOI:10.1103/PhysRevLett.98.010504. EPRINT arXiv:quant-ph/0607182.

[Sti02] Douglas R. Stinson. *Cryptography: Theory and Practice*. Chapman & Hall, 2nd edition, 2002.

[SU08] Douglas Stebila and Berkant Ustaoğlu. Towards denial of service-resistant key agreement protocols. Preprint, 2008. EPRINT http://www.douglas.stebila.ca/research/papers/SU08.

[SW03] Nigel P. Smart and Edward John Westwood. Point multiplication on ordinary elliptic curves over fields of characteristic three. *Applicable Algebra in Engineering, Communication and Computing*, **13**(6):485–497, April 2003. DOI:10.1007/s00200-002-0114-0.

[TOI03] Yuuki Tokunaga, Taisuaki Okamoto, and Nobuyuki Imoto. Anonymous quantum cash. In *ERATO Conference on Quantum Information Science (EQIS) 2003*, September 2003. URL http://www.qci.jst.go.jp/eqis03/program/papers/009-Tokunaga.ps.gz.

[TWMP07] David Taylor, Thomas Wu, Nikos Mavrogiannopoulos, and Trevor Perrin. Using the Secure Remote Password (SRP) protocol for TLS authentication, November 2007. URL http://www.ietf.org/rfc/rfc5054.txt. RFC 5054.

[Ust08a] Berkant Ustaoglu. *Key establishment – security models, protocols and usage*. PhD thesis, University of Waterloo, July 2008. URL http://hdl.handle.net/10012/3827.

[Ust08b] Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, **46**(3):329–342, March 2008. DOI:10.1007/s10623-007-9159-1. EPRINT http://eprint.iacr.org/2007/123.pdf.

[Vau05] Serge Vaudenay, editor. *Public Key Cryptography (PKC) 2005*, *LNCS*, volume 3386. Springer, 2005. DOI:10.1007/b105124.

[Ver07] Eric Verheul. Selecting secure passwords. In *Verification and Validation of Software Systems (VVVS) 2007*, March 2007. URL http://www.laquso.com/VVSS2007/presentations/track15b.pdf.

[Wal04] Colin D. Walter. Simple power analysis of unified code for ECC double and add. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems (CHES) 2004*, *LNCS*, volume 3156, pp. 191–204. Springer, 2004. DOI:10.1007/b99451.

[Wan01] Yongge Wang. IEEE P1363.2 submission / D2001-06-29 – standard specifications for public key cryptography: Password-based techniques – ECSRP, 2001. URL `http://www.sis.uncc.edu/~yonwang/papers/ecsrp.pdf`.

[WC81] Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *J. Computer and System Sciences*, **22**(3):265–279, 1981. DOI:10.1016/0022-0000(81)90033-7.

[Wee05] Hoeteck Wee. On obfuscating point functions. In *Proc. 37th Ann. ACM Symp. on the Theory of Computing (STOC)*, pp. 523–532. ACM Press, 2005. DOI:10.1145/1060590.1060669. EPRINT `http://eprint.iacr.org/2005/001`.

[WG00] David Wagner and Ian Goldberg. Proofs of security for the unix password hashing algorithm. In Tatsuaki Okamoto, editor, *Advances in Cryptology – Proc. ASIACRYPT 2000*, *LNCS*, volume 1976, pp. 560–572. Springer, 2000. DOI:10.1007/3-540-44448-3_43.

[Wie83] Stephen Wiesner. Conjugate coding. *ACM SIGACT News*, **15**(1):78–88, 1983. DOI:10.1145/1008908.1008920.

[WJHF04] Brent Waters, Ari Juels, J. Alex Halderman, and Edward W. Felten. New client puzzle outsourcing techniques for dos resistance. In Pfitzmann and Liu [PL04], pp. 246–256. DOI:10.1145/1030083.1030117.

[WR03] Xiaofeng Wang and M.K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Proc. 2003 IEEE Symposium on Security and Privacy (SP'03)*, pp. 78–92. IEEE Press, 2003. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=27002&arnumber=1199329`.

[Wri87] Peter Wright. *Spy Catcher: The Candid Autobiography of a Senior Intelligence Officer*. Penguin Viking, 1987.

[Wu98] Thomas Wu. The Secure Remote Password protocol. In *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 1998*, pp. 97–111. Internet Society, March 1998. URL `http://www.isoc.org/isoc/conferences/ndss/98/ndss98.htm`.

[Wu99] Thomas Wu. A real-world analysis of kerberos password security. In *Proc. Internet Society Network and Distributed System Security Symposium (NDSS) 1999* [NDS99].

[Wu02] Thomas Wu. SRP-6: Improvements and refinements to the Secure Remote Password protocol, October 2002. URL `http://srp.stanford.edu/srp6.ps`.

[Wu08] Thomas Wu. The Stanford SRP authentication project. Online, 2008. URL `http://srp.stanford.edu/`.

[WZ82] William K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, **299**:802–803, October 1982. DOI:10.1038/299802a0.

[XMM+07] Hai Xu, Lijun Ma, Alan Mink, Barry Hershman, and Xiao Tang. 1310-nm quantum key distribution system with up-conversion pump wavelength at 1550 nm. *Optics Express*, **15**(12):7247–7260, 2007. DOI:10.1364/OE.15.007247.

[YWWD06a] Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. Formal analysis and systematic construction of two-factor authentication scheme (short paper). In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security*, *LNCS*, volume 4307, pp. 82–91. Springer, 2006. DOI:10.1007/11935308_7. Full version available as [YWWD06b].

[YWWD06b] Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. Formal analysis and systematic construction of two-factor authentication scheme (short paper), 2006. EPRINT `http://eprint.iacr.org/2006/270`. Short version published as [YWWD06a].

[YY06] Eun-Jun Yoon and Kee-Young Yoo. An optimized two factor authenticated key exchange protocol in PWLANs. In Vassil N. Alexandrov, Geert Dick van Albada, Peter M.A. Sloot, and Jack Dongarra, editors, *Computational Science – ICCS 2006*, *LNCS*, volume 3992, pp. 1000–1007. Springer, 2006. DOI:10.1007/11758525_133.

[ZFQ+08] Yi Zhao, Chi-Hang Fred Fung, Bing Qi, Christine Chen, and Hoi-Kwong Lo. Experimental demonstration of time-shift attack against practical quantum key distribution systems, March 2008. EPRINT arXiv:0704.3253v2.

Notes

DOI denotes a Digital Object Identifier. An article labeled DOI:xxx can be accessed electronically at the URL `http://dx.doi.org/xxx`.

Eprints labeled arXiv:xxx can be accessed electronically at the URL `http://arxiv.org/abs/xxx`.

Index

List of Symbols

Part I: Classical Authenticated Key Exchange

$\mathsf{Adv}_P^{\text{c2s}}(\mathcal{A})$	The c2s (client-to-server authentication) advantage of $\mathcal{A}$ attacking protocol P.
$\mathsf{Adv}_P^{\text{ma}}(\mathcal{A})$	The ma (mutual authentication) advantage of $\mathcal{A}$ attacking protocol P.
$\mathsf{Adv}_P^{\text{s2c}}(\mathcal{A})$	The s2c (server-to-client authentication) advantage of $\mathcal{A}$ attacking protocol P.
$\mathsf{Adv}_{G,g}^{\text{CDH}}(\mathcal{A})$	The CDH advantage of $\mathcal{A}$ on group G generated by g.
$\mathsf{Adv}_{G,g}^{\text{DL}}(\mathcal{A})$	The DL advantage of $\mathcal{A}$ on group G generated by g.
$\mathsf{Adv}_{G,g}^{\text{GDH}}(\mathcal{A})$	The GDH advantage of $\mathcal{A}$ on group G generated by g.
$\mathsf{Adv}_{G,g}^{\text{LCDH}}(\mathcal{A})$	The List-CDH advantage of $\mathcal{A}$ on group G generated by g.
$\mathsf{Adv}_{G,g}^{\text{LCDH}}(\mathcal{A})$	The List-CDH advantage of $\mathcal{A}$ on group G generated by g.
$\mathsf{Adv}_P^{\text{ake}}(\mathcal{A})$	The ake advantange of $\mathcal{A}$ attacking protocol P.
$\mathsf{Adv}_P^{\text{p-ke}}(\mathcal{A})$	The passive-ke advantange of $\mathcal{A}$ attacking protocol P.
$\mathcal{A}$	The adversary; a probabilistic algorithm.
$\mathsf{Challenges}$	The set of challenges for a puzzling relation.
$\mathsf{Clients}$	The set of clients.
$\mathsf{DH}(g^x, g^y)$	The Diffie-Hellman value g^{xy} of g^x and g^y.
$\text{dlog}_g(g^x)$	The discrete logarithm x of g^x to the base g.
$\mathsf{RevealEphemeralKey}(\mathsf{sid})$	Returns the ephemeral private key of the owner in the session sid.
$\mathsf{EstablishParty}(\hat{M}, M)$	Establishes a party with identifier $\hat{M}$ and static public key M.
$\mathsf{Execute}_P(\hat{C}, i, \hat{S}, j)$	Causes client instance $\Pi_i^{\hat{C}}$ and server instance $\Pi_j^{\hat{S}}$ to execute protocol P and return the transcript of their messages.
Γ	The verifier corresponding to γ
γ	A long-term private value corresponding to a password.
$\gcd(a, b)$	The greatest common divisor of a and b.
λ	(Chapter 5:) A security parameter.
λ	(Chapters 2 and 6:) The slope in the elliptic curve point addition formula.

$\langle x \rangle$	The group generated by x.
$\mathbb{Z}_n$	The set of integers modulo n.
$\mathcal{O}$	The point at infinity.
$\mathcal{R}$	A puzzling relation.
$\mathrm{GF}(n)$	A Galois field.
$\mathrm{ord}_G(x)$	The order of x.
p_{add}	
$\hat{C}$	A client.
$\hat{S}$	A server.
$\mathsf{Passwords}$	The set of password strings.
p_{diff}	The probability of a conditional subtraction occurring in only one of two Montgomery multiplications.
p_{dist}	The probability of observing conditional subtractions and thus distinguishing point addition from point doubling.
$\Pi_i^{\hat{U}}$	The oracle representing protocol instance i of party $\hat{U}$.
pid	A partner identifier.
p_{sub}	The probability of a conditional subtraction occurring in a random Montgomery multiplication.
$\mathsf{pw}_{\hat{S}}[\hat{C}]$	The transformed password of client $\hat{C}$ with server $\hat{S}$ as held by server $\hat{S}$.
$\mathsf{pw}_{\hat{C},\hat{S}}$	The password of client $\hat{C}$ with server $\hat{S}$.
q_{ex}	Number of queries of type $\mathsf{Execute}$.
q_{ro}	Number of queries to random oracles.
q_{se}	Number of queries of type Send.
$\mathsf{re}_{\hat{S}}[\hat{C}]$	The transformed short-term password of client $\hat{C}$ with server $\hat{S}$ as held by server $\hat{S}$.
$\mathsf{re}_{\hat{C},\hat{S}}$	The short-term password of client $\hat{C}$ with server $\hat{S}$.
$\mathsf{Responses}$	The set of possible solutions to a puzzling relation (in Chapter 5).
$\mathsf{Responses}$	The set of short-term password strings.
$\mathsf{RevealPWC}_P(\hat{C},\hat{S})$	Returns the password $\mathsf{pw}_{\hat{C},\hat{S}}$ of client $\hat{C}$ with server $\hat{S}$.
$\mathsf{RevealPWS}_P(\hat{S},\hat{C})$	Returns the transformed password $\mathsf{pw}_{\hat{S}}[\hat{C}]$ of client $\hat{C}$ on server $\hat{S}$.
$\mathsf{RevealRe}_P(\hat{C},\hat{S})$	Returns the short-term password $\mathsf{re}_{\hat{C},\hat{S}}$ of client $\hat{C}$ with server $\hat{S}$.
$\mathsf{RevealSessionKey}_P(\hat{U},i)$	Returns the session key held by $\Pi_i^{\hat{U}}$.
Salts	The set of salts.
$\mathsf{Send}_P(\hat{U},i,M)$	Sends message M to instance $\Pi_i^{\hat{U}}$ which executes protocol P based on M, updates its state, and returns any outgoing messages.
$\mathsf{Servers}$	The set of servers.

sid	A session identifier.
sk	A session key.
Ψ	Session string identifier in the eCK model.
$\mathsf{RevealStaticKey}(\hat{A})$	Returns the static private key of party $\hat{A}$.
$\mathsf{Succ}_P^{\text{ake}}(\mathcal{A})$	The event that $\mathcal{A}$ guesses the Test bit b in a fresh session.
$\mathsf{Succ}_P^{\text{ake-f1}}(\mathcal{A})$	The event that $\mathcal{A}$ guesses the Test bit b in a fresh-in-the-first-factor session.
$\mathsf{Succ}_P^{\text{ake-f2}}(\mathcal{A})$	The event that $\mathcal{A}$ guesses the Test bit b in a fresh-in-the-second-factor session.
$\mathsf{Succ}_P^{\text{p-ke}}(\mathcal{A})$	The event that $\mathcal{A}$ guesses the Test bit b in a fresh-against-passive-adversaries session.
$\mathsf{Test}_P(\hat{U}, i)$	Choose $b \in_R \{0,1\}$; if $b = 1$, return the session key held by $\Pi_i^{\hat{U}}$, otherwise return a random string.
t_{exp}	The time required to perform an exponentiation in the group G.
φ	The Euler phi function.
E	An elliptic curve.
$E(F)$	The group of points on the elliptic curve E over the field F.
F	A finite field.
G	A finite cyclic group of order q generated by g.
g	The generator of the group G.
q	(Chapters 2-5:) The order of the group G generated by g.
q	(Chapter 6:) An odd prime which is the characteristic of the finite field F.
r	The length of an elliptic curve private key.
s	A salt value.
t	Running time.

Part II: Quantum Cryptography

$(V, \lvert\psi\rangle)$	A quantum coin scheme.
$(V, \{(s_i, \lvert\psi_i\rangle) : i \in \Gamma\})$	A quantum bill scheme.
$\lvert\psi\rangle$	An n-qubit pure state representing a quantum coin.
$\mathcal{H}^{2^n}$	A Hilbert space on n qubits.
k	The number of coins provided to an adversary.
U_ψ	A verification oracle for quantum coins: $U_\psi = I - 2\lvert\psi\rangle\langle\psi\rvert$
V	A quantum circuit for verifying a quantum coin.