Kristian Gjøsteen

Subgroup membership problems and public key cryptosystems

Dr. ing. thesis

Department of Mathematical Sciences
Norwegian University of Science and Technology
2004

Preface

This thesis contains my research work done between January 2000 and June 2004, while employed by the Department of Mathematical Sciences at the Norwegian University of Science and Technology. My advisors have been Idar Hansen and Alexei Rudakov. I am grateful for their continuous help, advice and encouragement.

In January 2004 I visited Ronald Cramer in Aarhus. I would like to thank him and the cryptology group in Aarhus for their hospitality and for the many enlightening discussions. The visit was very helpful for me and for the completion of my work.

Finally, I would like to thank my wife and my daughter for being there for me.

Trondheim, May 2004

Kristian Gjøsteen

Contents

List of Figures

Chapter 1

Introduction

Public key encryption was first proposed by Diffie and Hellman [16], and widely popularised with the RSA cryptosystem [37]. Over the years, the security goals of public key encryption have been studied [17, 22], as have adversary models [30, 36], and many public key cryptosystems have been proposed and analysed.

It turns out that the security of many of those cryptosystems [16, 18, 22, 29, 34, 35] are based on a common class of mathematical problems, called subgroup membership problems. Cramer and Shoup [10] designed a chosen-ciphertext-secure cryptosystem based on a general subgroup membership problem (generalising their previous work [9]), and provided two new instances. Yamamura and Saito [41] defined a general subgroup membership problem, catalogued several known subgroup membership problems, and designed a private information retrieval system based on a subgroup membership problem. Nieto, Boyd and Dawson [31] designed a cryptosystem based on essentially a symmetric subgroup membership problem (see Section 4.4 and Section 6.1).

Chapter 2 and 3 contain certain preliminary discussions necessary for the later work.

In Chapter 4, we discuss subgroup membership problems, both abstractly and concrete families. For all of the concrete examples, there is a related problem called the splitting problem. We discuss various elementary reductions, both abstract and for concrete families. In cryptographic applications, a third related problem, called the subgroup discrete logarithm problem, is also interesting, and we discuss this in some detail.

We also discuss a variant of the subgroup membership problem where there are two subgroups that are simultaneously hard to distinguish. We prove a useful reduction (Theorem 4.11) for this case. The technique used in the proof is reused throughout the thesis.

In Chapter 5, we discuss two homomorphic cryptosystems, based on trapdoor splitting problems. This gives us a uniform description of a number of homomorphic cryptosystems, and allows us to apply the theory and results of Chapter 4 to the security of those cryptosystems.

Using the technique of Theorem 4.11, we develop a homomorphic cryptosystem that is not based on a trapdoor problem. This gives us a fairly efficient cryptosystem, with potentially useful properties.

We also discuss the security of a homomorphic cryptosystem under a non-standard assumption. While these results are very weak, they are stronger than results obtained in the generic model.

In Chapter 6, we develop two key encapsulation methods. The first can be proven secure against passive attacks, using the same technique as in the proof of Theorem 4.11. The second method can be proven secure against active attacks in the random oracle model, but to do this, we need a certain non-standard assumption.

Finally, in Chapter 7 we discuss a small extension to the framework developed by Cramer and Shoup [10], again by essentially reusing the technique used to prove Theorem 4.11. This gives us a cryptosystem that is secure against chosen ciphertext attacks, without recourse to the random oracle model or non-standard assumptions. The cryptosystem is quite practical, and performs quite well compared to other variants of the Cramer-Shoup cryptosystem.

Chapter 2

Preliminaries

We discuss the notation and computational model used throughout this text. We shall also discuss certain computational problems common in cryptography. To simplify the main exposition, we have included a number of elementary results in this chapter.

2.1 Notation

The symbols $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ represent the natural numbers, the integers, the rational numbers and the real numbers, respectively. A *k-bit integer* is an integer n such that $2^{k-1} \leq n < 2^k$. For an integer $n > 1$, we use the notation $x \bmod n$ to denote the remainder of x divided by n. We denote the ring of integers modulo n by $\mathbb{Z}_n$, and the field with p elements by $\mathbb{F}_p$. We use the notation $[a]$ and $[a]_n$ to denote the residue class in $\mathbb{Z}_n$ represented by the integer a. The map $a \mapsto [a]_n$ is a ring homomorphism from $\mathbb{Z}$ onto $\mathbb{Z}_n$.

A *smooth* integer is an integer whose prime factors are small. Sometimes, we say B-smooth to mean that all the prime factors are below the integer bound B, or S-smooth to mean that all the prime factors are in the set S.

Let $n > 0$ be an integer. The *Euler phi-function* $\phi(n)$ is the number of non-negative integers less than n that are relatively prime to n.

If G is a group and $g \in G$, $\langle g \rangle$ denotes the cyclic subgroup of G generated by g.

We say that a function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if for any polynomial function $p(\tau)$, there exists a constant $\tau_0 \geq 0$ such that

$$|f(\tau)| \leq \frac{1}{p(\tau)} \text{ for } \tau > \tau_0.$$

Any function f that is not negligible is *significant*. We also say that $f(\tau)$ is *negligible in* τ.

A *distribution* on a countable set S is a function $\mu : S \to [0, 1]$ such that $\sum_{s \in S} \mu(s) = 1$. A pair (S, μ) is called a *probability space*. Let $X = (S, \mu)$ be a

probability space. For a subset S' of S use the notation

$$\Pr[X \in S'] = \sum_{s \in S'} \mu(s).$$

We also use the notation $\Pr[X = s] = \Pr[X \in \{s\}]$. If $X = (S, \mu)$ is a probability space, we shall say that X is a probability space on S. When we say that X is distributed according to some distribution μ on S, we mean that X is the probability space (S, μ).

If $X = (S, \mu)$ is a probability space, then $x \leftarrow X$ means that x is sampled from S according to the distribution μ, or $\Pr[x = s \mid x \leftarrow X] = \Pr[X = s] = \mu(s)$ for all $s \in S$. If S is a finite set, the notation $x \leftarrow S$ means that x is sampled uniformly at random from S.

If S, S' are countable sets and $f : S \to S'$ is a function, then f induces a map f^* from the set of distributions on S to the set of distributions on S'. If μ is a distribution on S, then the distribution $f^*(\mu) = \mu'$ on S' is given by

$$\mu'(s') = \sum_{s \in f^{-1}(s')} \mu(s).$$

If $X = (S, \mu)$ is a probability space and $f : S \to S'$ is a function, then we denote the probability space (S', μ') by $f(X)$.

A triple $(S, \mu, f : S \to S')$, where (S, μ) is a probability space and f is a function, is called a *random variable* on the probability space (S, μ) with values in S'. If we have random variables $X_1, X_2, \ldots, X_l$ and a function $f : S'_1 \times S'_2 \times \cdots \times S'_l \to S''$, we get a random variable $f(X_1, X_2, \ldots, X_l)$ on the joint probability space $(S_1, \mu_1) \times \cdots \times (S_l, \mu_l)$ with values in S''.

In general, we shall identify the random variable with the resulting probability space.

Let X and Y be probability spaces on some finite set S. The *statistical distance* between X and Y is defined to be

$$\mathrm{Dist}(X, Y) = \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|.$$

We note that $\mathrm{Dist}(\cdot, \cdot)$ is a metric on the set of probability spaces on S, and that $0 \leq \mathrm{Dist}(X, Y) \leq 1$. We say that two probability spaces X and Y are *ϵ-close* if $\mathrm{Dist}(X, Y) \leq \epsilon$. If two probability spaces X and Y are ϵ-close to each other for some small ϵ, we say that X is *almost equal* to Y, and vice versa. Two families of probability spaces $X(\tau)$ and $Y(\tau)$ are *statistically indistinguishable* if $\mathrm{Dist}(X(\tau), Y(\tau))$ is negligible in τ.

If X is a probability space on some finite subset S of the real numbers, then the *expected value* of X is

$$\mathrm{E}[X] = \sum_{s \in S} \Pr[X = s] s.$$

The projective plane over a field $\mathbb{F}$ is the set of triples $(x : y : z)$, where x, y and z are in the algebraic closure of $\mathbb{F}$, and at least one of x, y and z are non-zero, subject to the equivalence relation that $(x, y, z) \sim (x', y', z')$ if there exists $\lambda \in \mathbb{F}^*$ such that $(x, y, z) = (\lambda x', \lambda y', \lambda z')$. We denote the equivalence classes by $(x : y : z)$ and call them points.

An *elliptic curve* [40] defined over a field $\mathbb{F}$ of characteristic different from 2 and 3 is the set of points in the projective plane satisfying

$$y^2 z = x^3 + axz^2 + bz^3,$$

where $a, b \in \mathbb{F}$ and $4a^3 - 27b^2 \neq 0$. The point $(0 : 1 : 0)$ is called the point at infinity, and it is the only point with z-coordinate equal to 0. The elliptic curve is a group under a geometrically defined operation, where $(0 : 1 : 0)$ is the identity. The set of $\mathbb{F}$-rational points (points that can be represented by a triple $(x : y : z)$, with $x, y, z \in \mathbb{F}$) is a subgroup.

If $\mathbb{F}$ is a finite field with p^k elements, it can be shown that the map $(x : y : z) \mapsto (x^{p^k} : y^{p^k} : z^{p^k})$ is an endomorphism. It is called the *Frobenius endomorphism*.

If n is a composite number, we can define an elliptic curve over the ring $\mathbb{Z}_n$ to be the set of points

$$E(\mathbb{Z}_n) = \{(x, y, z) \in \mathbb{Z}^3 \setminus n\mathbb{Z}^3 \mid y^2 z \equiv x^3 + axz^2 + bz^3 \pmod{n}\},$$

subject to the equivalence relation that $(x, y, z) \sim (x', y', z')$ if there exists $\lambda \in \mathbb{Z}$ such that $\gcd(\lambda, n) = 1$ and

$$(x, y, z) \equiv (\lambda x', \lambda y', \lambda z') \pmod{n},$$

and where a, b are integers such that $4a^3 - 27b^2 \not\equiv 0 \pmod{n}$. We denote the equivalence classes by $(x : y : z)$ and call them points. The special point $(0 : 1 : 0)$ is on the curve.

It can be shown ([20, 25] and the references therein) that the usual formulae for the elliptic curve group operation over fields, suitably modified for operations in rings, actually gives a group operation on the set $E(\mathbb{Z}_n)$, and the point $(0 : 1 : 0)$ is the identity for this group operation.

If $n = pq$, we can of course consider E to be an elliptic curve over the fields $\mathbb{F}_p$ and $\mathbb{F}_q$, and using the Chinese remainder theorem, it is easy to show that there is a group isomorphism such that

$$E(\mathbb{Z}_n) \simeq E(\mathbb{F}_p) \times E(\mathbb{F}_q).$$

2.2 Computational model

An *algorithm* is a Turing machine that halts after a finite number of steps, regardless of input. It has an input tape and an output tape. The algorithm may have access to a *random tape*, in which case we call the algorithm *probabilistic*. The random tape is just a tape containing random zeros and ones, and is sometimes

called the *coin tosses* of the algorithm. We usually consider the random tape to be infinitely long. A probabilistic algorithm should terminate after a finite number of steps, regardless of input and random tape.

Sometimes, we want to fix the random tape, and we denote by $A(x;r)$ the function computed by A when given the fixed random tape r. Running an algorithm twice with the same random tape is sometimes called *rewinding the random tape*.

For a fixed input x, we can assume that the random tape is of finite length. The uniform distribution on the set of fixed-length random tapes is then a probability space, and the algorithm A and the fixed input x gives a function on this probability space. We denote this random variable by $A(x)$. If X is a probability space on a finite set of possible inputs, we denote by $A(X)$ the random variable given by applying the function $A(x;r)$ to the joint probability space of X and algorithm's coin tosses.

An algorithm may have access to one or more *oracles*. While the word oracle suggests divine origin, our oracles are devices we use to give our algorithms certain well-defined powers of computation. The algorithm may ask a question, and the oracle answers the question in some well-defined way.

The idea is that the oracle performs some computation that the algorithm cannot do on its own, perhaps because we want to allow an algorithm to perform some computation based upon some knowledge[1] (such as a secret key), without giving that knowledge to the algorithm.

An oracle is modelled as a pair of tapes. The algorithm writes queries to one tape (the *query-tape*) and reads answers from the other tape (the *answer-tape*). From the algorithm's point of view, the answer to a query is available immediately after the query was written. Except for the answer, the oracle's computations are hidden from the algorithm.

Every oracle has access to its own random tape which is independent of all other random tapes in the experiment, and the algorithm is not allowed access to the random tape, neither to look at it or to manipulate it. We also allow oracles to preserve state information between queries (that is, the oracle's answer may depend on previous queries to the oracle).

The *cost* of an algorithm is the number of steps the algorithm needs to run. If the algorithm has access to any oracles, the cost should include the number of queries to each oracle. If the cost is relatively small, we say that the algorithm is *efficient*.

We say that a function $f : S_1 \to S_2$ is *easy to compute* if some deterministic and efficient algorithm for computing f can easily be derived from the description of f. We identify f with this algorithm.

[1] We follow Goldreich [21] and distinguish between *information* and *knowledge*. We have information about some value if we can compute the value given unbounded computational resources. We have knowledge about some value if we can compute it given bounded computational resources.

These definition are rather vague and informal. Asymptotically, there is a more precise definition of relatively small cost. An algorithm is *polynomial-time* if the Turing machine halts after a number of steps polynomial in the length of its input. This also implies that the number of queries to any oracles is bounded by a polynomial in the length of the input. A probabilistic algorithm that is polynomial-time is called *efficient*.

A *problem* $\mathcal{P} = (P, X, S, f)$ consists of a set P of problem instances, a probability space X on that set, a finite set S of possible answers, and a function $f : P \to S$ that gives the answer to every problem instance. To simplify the exposition, we denote sampling a problem instance according to X by $x \leftarrow \mathcal{P}$. The correct answer to instance x is denoted by $\mathcal{P}(x)$.

A *subproblem* $\mathcal{P}' \subseteq \mathcal{P}$ is a tuple (P', X', S', f') where $P' \subseteq P$, X' is the probability space X restricted to P', f' is f restricted to P' and $S' = S$.

The *success probability* of an algorithm A in solving a problem $\mathcal{P}$ is

$$\mathrm{Succ}_A^{\mathcal{P}} = \Pr[A(x) = \mathcal{P}(x) \mid x \leftarrow \mathcal{P}].$$

The probability is computed over the joint probability space of the algorithm's random tape (if it is probabilistic) and the input.

Since the set of possible answers to a problem is finite, there is always a probability that an algorithm will answer correctly by guessing. The *advantage* of an algorithm measures how much better it is at answering the problem than an algorithm that guesses an answer according to some fixed distribution. The definition is

$$\mathrm{Adv}_A^{\mathcal{P}} = \inf_Y |\Pr[A(x) = \mathcal{P}(x) \mid x \leftarrow \mathcal{P}] - \Pr[y = \mathcal{P}(x) \mid x \leftarrow \mathcal{P} \wedge y \leftarrow Y]|,$$

where the distribution Y ranges over all possible distributions on the set of possible answers to $\mathcal{P}$. Note that this definition of advantage may not always be appropriate.

A *reduction* from a problem $\mathcal{P}$ to a problem $\mathcal{P}'$ is an algorithm A that uses an oracle for solving $\mathcal{P}'$ to solve $\mathcal{P}$. The *cost* of the reduction is the cost of running A. A reduction A is only interesting if its cost is relatively small, and the success probability of A is not small compared to the success probability of the oracle.

A problem $\mathcal{P} = (P, X, S, f)$ is *self-reducible* if there is an algorithm A (with a small cost) that takes as input an instance x of $\mathcal{P}$ and outputs an instance x' of $\mathcal{P}$ along with a function $g : S \to S$, such that $g(f(x')) = f(x)$, and $A(x)$ is, except with small probability, independent of x and ϵ-close to X for some small ϵ.

Any self-reducible problem that is easy to solve on some significant subset (probability-wise) of its problem instances is easy to solve for almost all of its instances.

A *parameterised problem* $\mathcal{P}$ is a family $\{\mathcal{P}_\tau\}$ of problems indexed by integers $\tau > 0$. For an algorithm A, we define *success probability* as

$$\mathrm{Succ}_A^{\mathcal{P}}(\tau) = \Pr[A(x) = \mathcal{P}_\tau(x) \mid x \leftarrow \mathcal{P}_\tau],$$

and *advantage* as

$$\begin{aligned}\mathrm{Adv}_A^{\mathcal{P}}(\tau) = \inf_Y |\Pr[A(x) = \mathcal{P}_\tau(x) \mid x \leftarrow \mathcal{P}_\tau] - \\ \Pr[y = \mathcal{P}_\tau(x) \mid x \leftarrow \mathcal{P}_\tau \wedge y \leftarrow Y]|,\end{aligned} \tag{2.1}$$

where the distribution Y ranges over all possible distributions on the set of possible answers to $\mathcal{P}_\tau$.

We drop the word parameterised if it is clear from the context that the problem is parameterised.

Let $\mathcal{P}$ and $\mathcal{P}'$ be parameterised problems. A *reduction* from $\mathcal{P}$ to $\mathcal{P}'$ is an algorithm A that reduces every problem in $\mathcal{P}_\tau$ to the corresponding problem in $\mathcal{P}'_\tau$. A reduction is *efficient* if the algorithm A is efficient in τ, and if the oracle for $\mathcal{P}'$ has significant advantage, then A has significant advantage.

If there is an efficient reduction from $\mathcal{P}$ to $\mathcal{P}'$, we say that $\mathcal{P}$ *reduces to* $\mathcal{P}'$. If $\mathcal{P}$ reduces to $\mathcal{P}'$ and $\mathcal{P}'$ reduces to $\mathcal{P}$, we say that $\mathcal{P}$ and $\mathcal{P}'$ are *equivalent*.

In general, we say that a problem $\mathcal{P}$ is (c, ϵ)-*hard* if there are no algorithms that solve $\mathcal{P}$ with advantage greater than ϵ at a cost less than c. If we simply say *hard*, we actually mean (c, ϵ)-hard for some large cost c and some small probability ϵ, so that it is extremely unlikely that any adversary has the resources to solve the problem.

We say that a parameterised problem $\mathcal{P}$ is *hard* if for any efficient algorithm, its advantage is negligible in τ.

Let $\mathcal{P}$ be a parameterised problem and suppose that A is an algorithm that solves $\mathcal{P}$. A may have access to some oracles. Suppose A' is another algorithm that solves $\mathcal{P}$, that it has access to the same oracles as A has, but that it not only outputs the answer to $\mathcal{P}$, but also outputs a *trace tr* that contains some information about its calculations. We shall need an experiment in order to define what a trace-variant is.

Experiment 1.
Input: $\mathcal{P}$, A, A', τ.

1. $x \leftarrow \mathcal{P}_\tau$.
2. $r \leftarrow \{0,1\}^*$.
3. $y \leftarrow A(x; r)$.
4. $(y', tr) \leftarrow A'(x; r)$.
5. If $y = y'$, then output 1, otherwise output 0.

Output: 0 or 1.

We say that A' is a *trace-variant* of A if

$$\Pr[\mathbf{Exp1}(\mathcal{P}, A, A', \tau) = 0]$$

is negligible in τ. Note that A and A' are run with the same random tape. If A and A' have access to any oracles, then the oracles' random tapes are also rewound.

Obviously, there are all kinds of trivial trace-variants of any algorithm.

A *trapdoor problem* $\mathcal{P}$ is a set of pairs $(\mathcal{P}', \sigma)$ along with a distribution on that set, where $\mathcal{P}'$ is a problem and σ is a function that is easy to compute, such that for any instance x of $\mathcal{P}'$, $\sigma(x) = \mathcal{P}'(x)$.

The *underlying problem* is the problem given by first sampling $(\mathcal{P}', \sigma)$ from $\mathcal{P}$, forgetting σ, and then sampling a problem instance x from $\mathcal{P}'$.

A *parameterised trapdoor problem* $\mathcal{P}$ is a family $\{\mathcal{P}_\tau\}$ of trapdoor problems indexed by integers $\tau > 0$. We say that a trapdoor problem is *hard* if the underlying parameterised problem is hard.

2.3 Common problems

We shall briefly discuss several problems that are common in cryptography and that are generally assumed to be hard. For a general discussion of these problems, see [28].

2.3.1 Factoring

Definition. Let $n > 1$ be a composite integer. The *factoring* problem $\mathcal{FACT}_n$ is to find the prime factorisation of n.

It can be shown that anyone who can compute (a multiple) of $\phi(n)$ can factor.

Let $n > 0$ be a composite integer with two prime factors p and q, such that both p and q are distinct τ-bit integers. Let N_τ be a probability space on the set of all such integers, and let $N = \{N_\tau\}_{\tau \geq 1}$. The distribution on N_τ is the distribution given by some prime sampling algorithm. It is generally believed that $\mathcal{FACT}_N$ is a hard problem for most reasonable prime sampling algorithms.

Definition. Let n be a composite integer with two primes factors p and q. Let $e > 1$ be an integer that is relatively prime to $\phi(n)$, and let $f : \mathbb{Z}_n \to \mathbb{Z}_n$ be the function defined by $f(m) = m^e$.

The *RSA* [37] problem $\mathcal{RSA}_{n,e}$ is the following: The instances are elements of $\mathbb{Z}_n$ chosen uniformly at random, and the answer to instance x is $f^{-1}(x)$.

To fully specify the RSA problem, we should not only say how n is chosen, but also how e is chosen. Typically, this is the smallest prime e above some bound that satisfies $\gcd(e, \phi(n)) = 1$. When we do not particularly care about e, we ignore it.

We derive the RSA problem $\mathcal{RSA}_N$ from N. It is obvious that $\mathcal{RSA}_N$ reduces to $\mathcal{FACT}_N$. It is not known whether the RSA problem is easier than factoring, but the best known method for computing the inverse function is by factoring the modulus.

The *RSA trapdoor problem* associates with the problem $\mathcal{RSA}_{n,e}$ the function $x \mapsto x^d$, where $d \equiv e^{-1} \pmod{\phi(n)}$.

2.3.2 Discrete logarithms

Definition. Let G be a finite cyclic group, let g be a generator, and let $y \in G$. The *discrete logarithm* of y to the *base* g is an integer $0 \leq a < |G|$ such that $y = g^a$. We sometimes write $\log_g y \equiv a \pmod{|G|}$.

The discrete logarithm problem $\mathcal{DL}_G$ is the set of triples (G, g, y), where y is sampled uniformly from G and the answer is $\log_g y$.

If $|G|$ is a smooth integer, then the Pohlig-Hellman [28] algorithm shows us that the discrete logarithm problem in $|G|$ is not difficult.

It is easy to show that if one can compute discrete logarithms to some base g, then one can compute the order of the group. It is also easy to show that if one can compute discrete logarithms to one base, one can compute discrete logarithms to any base.

Suppose the group order $|G|$ is known. If X is distributed uniformly on $\{0, \ldots, |G| - 1\}$, then yg^X is distributed uniformly on G, it is independent of y and $\log_g yg^X \equiv X + \log_g y \pmod{|G|}$. This means that the discrete logarithm problem is self-reducible when the group order is known.

If Γ is a probability space on a set of groups the *generalised discrete logarithm problem* $\mathcal{DL}_\Gamma$ derived from Γ is a set of triples (G, g, y) where G is sampled from Γ and g, y are sampled uniformly from G. If y is in the subgroup of G generated by g, then the answer is $\log_g y$, otherwise the answer is the special symbol $\perp$.

Let $N = \{N_\tau\}$ be as in Section 2.3.1. To N_τ we associate the groups $\Gamma_\tau = \{\mathbb{Z}_n^* \mid n \in N_\tau\}$ and we get the family $\Gamma = \{\Gamma_\tau\}$ associated to N. It can be shown that $\mathcal{FACT}_N$ reduces to $\mathcal{DL}_\Gamma$.

2.3.3 The Diffie-Hellman problems

Definition. Let G be a cyclic group generated by g. The *Computational Diffie-Hellman* [16] problem $\mathcal{CDH}_G$ is the set of quadruples (G, g, x, y), where x and y are sampled uniformly from G and the answer is $z \in G$ such that $\log_g z \equiv \log_g x \log_g y \pmod{|G|}$.

Suppose G has prime order. Let (G, g, x, y) be an instance of $\mathcal{CDH}_G$ with answer z. Let U, V and W be uniformly and independently distributed on $\{0, \ldots, |G| - 1\}$. Then (G, g^U, x^{UV}, y^{UW}) is the uniform distribution on the set of problem instances for $\mathcal{CDH}_G$, it is independent of (G, g, x, y) (except when x or y is 1), and the answer is z^{UVW}. This means that $\mathcal{CDH}_G$ is self-reducible.

It is clear that anyone who can compute discrete logarithms in G can also solve the Computational Diffie-Hellman problem in G. As for discrete logarithms, we

can define a generalised Computational Diffie-Hellman problem. If x or y is not in $\langle g \rangle$, then the correct answer is $\perp$.

Maurer [27] showed that, given a group of prime order p and an elliptic curve defined over $\mathbb{F}_p$ with a smooth number of $\mathbb{F}_p$-rational points, there is a reasonably efficient reduction from the Computational Diffie-Hellman problem to the discrete logarithm problem. As a consequence, the Computational Diffie-Hellman problem is widely believed to be as difficult as computing discrete logarithms.

A *safe prime* is a prime p such that $(p-1)/2$ is also prime. Correspondingly, p is a *Sophie-Germain*-prime if $2p+1$ is also prime. Let $N = \{N_\tau\}$ be as in Section 2.3.1, except that we restrict the primes involved to be safe primes. Let Q_n be the subgroup of quadratic residues of $\mathbb{Z}_n^*$, let Γ_τ be the set of groups $\{Q_n \mid n \in N_\tau\}$ with the derived distribution, and let $\Gamma = \{\Gamma_\tau\}$. It can be shown that $\mathcal{FACT}_N$ reduces to $\mathcal{CDH}_\Gamma$.

In contrast to factoring and discrete logarithms, there seems to be no easy way to decide if an algorithm returns the correct answer to a Computational Diffie-Hellman problem. This gives rise to a decision problem.

Definition. Let G be a cyclic group generated by g. The *Decision Diffie-Hellman* [7] problem $\mathcal{DDH}_G$ in G is to distinguish the distribution defined by (g, g^X, g^Y, g^{XY}) from (g, g^X, g^Y, g^Z), where X, Y are independently and uniformly distributed on $\{0, \ldots, |G|-1\}$, and Z is uniformly distributed on $\{0, \ldots, |G|-1\} \setminus \{XY \bmod |G|\}$.

Suppose G has prime order. It is quite clear that if (g, x, y, z) is an instance of $\mathcal{DDH}_G$, then with U, V, W uniformly and independently distributed on $\{0, \ldots, |G|-1\}$, $(g^U, x^{UV}, y^{UW}, z^{UVW})$ is uniformly distributed over the set of problem instances for $\mathcal{DDH}_G$ and it is independent (except when x or y is 1), and the answer remains the same. Hence, the $\mathcal{DDH}_G$ problem is self-reducible.

We can use the following experiment to measure the advantage of an attacker against the Decision Diffie-Hellman problem.

Experiment 2.
Input: A, G, g; $G = \langle g \rangle$.

1. $u, v \leftarrow \{0, \ldots, |G|-1\}$.
2. $b \leftarrow \{0, 1\}$.
3. If $b = 1$, then $w \leftarrow uv$, otherwise $w \leftarrow \{0, \ldots, |G|-1\} \setminus \{uv \bmod |G|\}$.
4. $b' \leftarrow A(G, g, g^u, g^v, g^w)$.
5. If $b = b'$, output 1, otherwise output 0.

Output: 0 or 1.

It is easy to see that

$$\mathrm{Adv}_A^{\mathcal{DDH}_G} = |\Pr[\mathbf{Exp2}(A, G, g) = 1] - 1/2|.$$

A nice survey on the Decision Diffie-Hellman problem is [7]. One interesting result is that there does exist groups in which the Computational Diffie-Hellman problem is believed to be difficult, but the Decision Diffie-Hellman problem is easy [24]. We shall return to this in Chapter 6.

There is a trapdoor variant of $\mathcal{CDH}_G$. The problem is given by (G, g, g^a), where a is sampled uniformly from $\{0, \ldots, |G|-1\}$, the problem instance is (G, g, g^a, x), where x is sampled uniformly from G, and σ is the function $x \mapsto x^a$.

2.4 Elementary results

We will need several elementary results later on, and to streamline the presentation, we list them here.

Lemma 2.1. *Let S be a finite set and let $T \subseteq S$ be a subset. Let X and Y be two probability spaces on S that are ϵ-close. Then*

$$|\Pr[x \in T \mid x \leftarrow X] - \Pr[y \in T \mid y \leftarrow Y]| \leq \epsilon.$$

Proof. Computation. □

Lemma 2.2. *Let S' be a proper, non-empty subset of the finite set S. Let X be uniformly distributed on S, and let Y be uniformly distributed on $S \setminus S'$. Then*

$$\mathrm{Dist}(X, Y) = \frac{2|S'|}{|S|}.$$

Proof. Computation. □

Lemma 2.3. *Let n be a positive integer, and let t_0 be an integer such $n < 2^{t_0}$. Let X be uniformly distributed on $\mathbb{Z}_n$, let Y_t be uniformly distributed on $\{0, \ldots, 2^{t_0+t} - 1\} \subseteq \mathbb{Z}$ and let Z_t be the probability space $[Y_t]_n$. Then*

$$\mathrm{Dist}(X, Z_t) \leq 2^{-t}.$$

Proof. From the definition, we have that

$$\mathrm{Dist}(X, Z_t) = \sum_{i=0}^{n-1} |\Pr[X = [i]] - \Pr[Z_t = [i]]|.$$

Let $r = 2^{t_0+t}/n$ and $m = 2^{t_0+t} \bmod n$. Reordering the sum and computing the probabilities, we find that

$$\mathrm{Dist}(X, Z_t) = \frac{m}{n}\left|1 - r^{-1}\lceil r \rceil\right| + \frac{n-m}{n}\left|1 - r^{-1}\lfloor r \rfloor\right|.$$

Since $r - 1 < \lfloor r \rfloor \leq r \leq \lceil r \rceil < r + 1$, we find that

$$\mathrm{Dist}(X, Z_t) \leq \frac{m}{n}\frac{1}{r} + \frac{n-m}{n}\frac{1}{r} = \frac{n}{2^{t_0+t}} \leq 2^{-t},$$

which concludes the proof. □

Lemma 2.4. *Let G be a finite cyclic group with generator g, and suppose t_0 is an integer such that $|G| < 2^{t_0}$. Then for any integer $t > 0$ there is an algorithm that samples elements of G from a probability space that is 2^{-t}-close to the uniform distribution, using $O(t)$ exponentiations in the group.*

Proof. The algorithm simply samples y uniformly at random from the set $\{0, \ldots, 2^{t_0+t} - 1\}$ and computes g^y. It is clear that this requires the same amount of work as $O(t)$ exponentiations in the group, and the resulting distribution is 2^{-t}-close to uniform by Lemma 2.3. □

Lemma 2.5. *Let G be a finite cyclic group. The probability that an element x sampled uniformly at random from G is a generator for G is*

$$\Pr[x \text{ is a generator} \mid x \leftarrow G] = \frac{\phi(|G|)}{|G|}.$$

Proof. G is isomorphic to $\mathbb{Z}_{|G|}$, and the claim follows from the definition of the Euler phi-function. □

Suppose we have an experiment that has a fixed number of possible outcomes, numbered from 0 to $d-1$, that we can repeat the experiment such that two different outcomes are independent, and that outcome number i occurs with a fixed probability α_i. Let X_i count the number of times the outcome i occurs in a collection of experiments. Then X_i is a random variable, and $(X_0, X_1, \ldots, X_{d-1})$ is *multinomially distributed* with parameters $\alpha_0, \ldots, \alpha_{d-1}$.

Lemma 2.6. *Let $(X_0, X_1, \ldots, X_{d-1})$ be a multinomially distributed random variable with parameters $\alpha_0, \alpha_1, \ldots, \alpha_{d-1}$. Suppose that $\alpha_0 \geq \alpha_i + \epsilon$ for $i = 1, \ldots, d-1$ and $\epsilon > 0$. Then after $(d^3(1-\alpha_0))/(\epsilon^3\alpha_0^2(1-\beta))$ experiments,*

$$\Pr[X_0 = \max\{X_0, \ldots, X_{d-1}\}] \geq \beta.$$

Proof. Let l be the number of experiments, let T_0 be the event that $|X_0/l - \alpha_0| \leq (d-1)\epsilon/d$, and let T_i be the event that $|X_i/l - \alpha_i| \leq \epsilon/d$, $1 \leq i < d$.

We have that

$$\Pr[X_0 = \max\{X_i\}] \geq \Pr[T_0 \wedge T_1 \wedge \cdots \wedge T_{d-1}].$$

If we assume $T_1 \wedge \cdots \wedge T_{d-1}$, then

$$\begin{aligned} |X_0 - l\alpha_0| &= |l - X_1 - X_2 - \cdots - X_{d-1} - l(1 - \alpha_1 - \cdots - \alpha_{d-1})| \\ &\leq |X_1 - l\alpha_1| + \cdots + |X_{d-1} - l\alpha_{d-1}| \\ &\leq \frac{d-1}{d} l\epsilon. \end{aligned}$$

So $\Pr[T_0 \mid T_1 \wedge \cdots \wedge T_{d-1}] = 1$.

Now we consider $\Pr[T_i \mid T_{i+1} \wedge \cdots \wedge T_{d-1}]$, and the marginal distribution of X_i. It has parameter

$$\alpha_i' = \frac{\alpha_i}{1 - \sum_{j=i+1}^{d-1} \alpha_j},$$

so with $Y_i = l - \sum_{j=i+1}^{d-1} X_j$ experiments, we get

$$\begin{aligned}\Pr[T_i|T_{i+1} \wedge \ldots T_{d-1}] &= 1 - \frac{\alpha_i'(1-\alpha_i')}{Y_i\epsilon^2/d^2} \\ &= 1 - \frac{d^2\alpha_i(1-\sum_{j=i}^{d-1}\alpha_j)}{\epsilon^2 Y_i(1-\sum_{j=i+1}^{d-1}\alpha_j)^2} \\ &\geq 1 - \frac{d^2\alpha_i(1-\sum_{j=i}^{d-1}\alpha_j)}{\epsilon^2 Y_i\alpha_0^2}.\end{aligned}$$

Under the condition $T_{i+1} \wedge \cdots \wedge T_{d-1}$, we get that

$$l - \sum_{j=k+1}^{d-1}(l\alpha_j + l\epsilon/d) \leq Y_i \leq l - \sum_{j=k+1}^{d-1}(l\alpha_j - l\epsilon/d).$$

This means that

$$\begin{aligned}\Pr[T_i|T_{i+1} \wedge \ldots T_{d-1}] &\geq 1 - \frac{d^2\alpha_i(1-\sum_{j=1}^{d-1}\alpha_j)}{\epsilon^2\alpha_0^2 l(1-\sum_{j=k+1}^{d-1}(\alpha_j+\epsilon/d))} \\ &\geq 1 - \frac{d^2\alpha_i}{\epsilon^2\alpha_0^2 l(\alpha_0-(d-1)\epsilon/d)} \\ &\geq 1 - \frac{d^3\alpha_i}{\epsilon^3\alpha_0^2 l},\end{aligned}$$

since $\alpha_0 > \epsilon$.

This gives us

$$\begin{aligned}\Pr[T_0 \wedge \cdots \wedge T_{d-1}] &= \prod_{i=1}^{d-1}\Pr[T_i|T_{i+1} \wedge \cdots \wedge T_{d-1}] \\ &\geq \prod_{i=1}^{d-1}\left(1 - \frac{d^3\alpha_i}{\epsilon^3\alpha_0^2 l}\right) \\ &\geq 1 - \frac{d^3(1-\alpha_0)}{\alpha_0^2\epsilon^3 l}.\end{aligned}$$

So with $\beta = 1 - \frac{d^3(1-\alpha_0)}{\alpha_0^2\epsilon^3 l}$, the result follows. □

Chapter 3

Provable security

We shall give a brief introduction to the setting in which we do our work. The goal of provable security is to provide a reduction from the security of the cryptosystem to some well-studied problem (the *underlying problem*). By itself, this says nothing about the security of the cryptosystem. But if we make the assumption that there are no algorithms that solve the underlying problem with a significant probability and at a cost less than some bound, the reduction then says something about the minimal cost for breaking the cryptosystem.

The main goal of this chapter is to explain what "breaking the cryptosystem" may mean, that is, what the security requirements of public key cryptosystems are, and what kind of attacks we allow. We shall also discuss the different models in which we obtain our security proofs, and the assumptions underlying those models.

3.1 Public key cryptosystems

The following definition of a public key cryptosystem is suitable for our purposes.

Definition. A *public key cryptosystem* Π consists of three algorithms, $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$. The *key generation algorithm* $\mathcal{K}$ is an efficient algorithm that takes a *security parameter* as input and outputs a *public key* pk and *private key* sk. The public key specifies among other things a finite set of possible messages and a finite set of possible ciphertexts, denoted by pk_M and pk_C.

The encryption algorithm $\mathcal{E}$ is an efficient algorithm that takes a public key and a message as input and outputs a ciphertext. The decryption algorithm $\mathcal{D}$ is a deterministic polynomial-time algorithm that takes a private key and a ciphertext as input and outputs either a message or the special symbol $\perp$. We require that for any public-private key pair (pk, sk) and any message m in the set of messages specified by pk, $\mathcal{D}(sk, \mathcal{E}(pk, m)) = m$.

Note that the security parameter input to $\mathcal{K}$ should be input as a string of 1's of length τ (denoted by 1^τ), not as (say) a binary encoding. Since we require the

algorithm to be efficient, the number of steps must be bounded by a polynomial in the input length. In the former case, this would be a polynomial in τ. In the latter case, it would be a polynomial in $\log_2 \tau$, which is wrong. We will henceforth ignore this slightly subtle point, and use the notation $\mathcal{K}(\tau)$ as a short-hand for $\mathcal{K}(1^\tau)$.

We shall also be interested in public key cryptosystems with a certain property.

Definition. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public key cryptosystem and let (pk, sk) be a public-private key pair for Π. Let $* : pk_M \times pk_M \to pk_M$ and $*' : pk_C \times pk_C \to pk_C$ be binary operations such that for all $c, c' \in pk_C$, we have that

$$\mathcal{D}(c *' c') = \mathcal{D}(c) * \mathcal{D}(c').$$

We then say that Π is *homomorphic with respect to* $*$ *and* $*'$.

3.2 Adversary models

In a public key cryptosystem, the adversary obviously knows the public key. This means that he can always mount a *chosen plaintext attack*, wherein he can encrypt any message of his choice with the given public key.

We can assume that the adversary observes exactly one ciphertext (the challenge ciphertext) that he does not know the decryption of. Observing more than one unknown ciphertext does not help him, since he can create ciphertexts himself, and then forget about their decryptions.

We model the adversary as two separate stages. First, the adversary receives the public key and may do some arbitrary computations. He then outputs some state information that is passed on to the second stage, and possibly some information that influences the challenge ciphertext. This is often called the "find"-stage.

After the first stage is complete, the challenge ciphertext is created outside of the adversary's control. It may, depending on the attack, be influenced by the output of the first stage, but obviously not determined by it.

The second stage receives the public key, the state information output by the first stage and the challenge ciphertext. It tries to determine some information about the challenge ciphertext. This is often called the "guess"-stage.

The adversary A is then a pair of algorithms (A_1, A_2), where A_1 carries out the first stage, and A_2 carries out the second.

It is conceivable that the adversary can do more than just computations based on the public key and the ciphertext. If the adversary can somehow obtain decryptions of ciphertexts he has created, he may be able to learn something about the secret key. If the attacker can modify a ciphertext and somehow get some information about the decryption of the modified ciphertext, it is possible that he can recover some information about the decryption of the original ciphertext, or the secret key.

This suggests that the adversary should be allowed to do more than just computations. To model this, we allow the adversary to decrypt ciphertexts using the secret key. One variant is to give the adversary access to the decryption oracle during the first stage [30]. This is called a *non-adaptive chosen ciphertext attack*.

While this may in certain situations be a realistic scenario, it is usually unrealistic to assume that the adversary does not have any additional powers during the second stage. The adversary's task is obviously trivial if we give him access to a decryption oracle in the second stage. The solution [36] is to give him access to a restricted decryption oracle, that refuses to decrypt the challenge ciphertext, but decrypts any other ciphertext. This is called an *adaptive chosen ciphertext attack*, or simply a *chosen ciphertext attack*.

In practice, this seems to give the adversary unreasonable advantages. But it makes sense, since if a cryptosystem is secure against a (non-adaptive) chosen ciphertext adversary, then it will certainly be secure against less powerful, realistic adversaries.

3.3 Security goals

The *decryption problem* for Π and a fixed public-private key pair (pk, sk) is the set of possible ciphertexts output by $\mathcal{E}$ when the message is sampled uniformly at random from pk_M. The adversary is given the public key pk and the ciphertext c. The answer to each instance is the decryption of the ciphertext.

Definition. A public key cryptosystem Π is *one-way* if the corresponding decryption problem is hard.

In many practical situations, one-way is not sufficient to guarantee security. Goldwasser and Micali [22] first proposed the notion of *semantic security*. This is a variant of Shannon's *perfect secrecy* [38] for adversaries with bounded computational power. The idea is that it should be difficult to decrypt a ciphertext even when the message space is restricted, and that it should be difficult to recover any information about the decryption of the ciphertext, beyond what is a-priori known from the message distribution.

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public key cryptosystem. We shall allow the adversary to influence the choice of message. We do this by allowing the adversary's "find"-stage to output an arbitrary function $f : pk_M \to \{0, 1\}$ along with a probability space X on pk_M such that

$$|\Pr[f(x) = 0 \mid x \leftarrow X] - 1/2|$$

is negligible. The message m to be encrypted is the sampled from X, and the adversary's "guess"-stage is given the challenge ciphertext. Its task is to guess $f(m)$.

We assume that there are efficient algorithms available for sampling X and computing f.

We use the following experiment to measure the adversary's advantage.

Experiment 1.
Input: Π, $A = (A_1, A_2)$, τ.

1. $(pk, sk) \leftarrow \mathcal{K}(\tau)$.
2. $(X, f, o) \leftarrow A_1(pk)$.
3. $m \leftarrow X$, $c \leftarrow \mathcal{E}(pk, m)$.
4. $b \leftarrow A_2(pk, c, o)$.
5. If $f(m) = b$, output 1, otherwise output 0.

Output: 0 or 1.

Here, o is some state information that A_1 wants to pass on to A_2. The adversary's advantage is then defined to be

$$\mathrm{Adv}_A(\tau) = |\Pr[\mathbf{Exp1}(\Pi, A, \tau) = 1] - 1/2|.$$

Definition. We say that Π is *semantically secure* if any efficient adversary as above has negligible advantage.

We note that some definitions of semantic security require X to be uniformly distributed. Also note that semantic security is not a problem in the sense of Section 2.2, since the problem solver participates in an arbitrary way in sampling the challenge ciphertext and the answer.

It is quite obvious that any system that is semantically secure is also one-way.

Working with semantic security is awkward, but there are simpler notions available. Suppose the adversary's "find"-stage outputs a probability space X given by

$$\Pr[X = m] = \begin{cases} \frac{1}{2} & m = m_0 \text{ or } m = m_1, \\ 0 & \text{otherwise,} \end{cases}$$

where m_0, m_1 are two distinct messages in pk_M, and the function f is defined to be $f(m_b) = b$, and otherwise arbitrary.

The adversary's "guess"-stage must then decide which of the two messages the challenge ciphertext decrypts to.

Definition. We say that Π has *indistinguishable encryptions* if any efficient adversary as described above has probability negligibly different from 1/2 in guessing which message was encrypted.

Indistinguishable encryptions is often called *polynomial security* in the literature.

It is quite obvious that if Π is semantically secure, then Π has indistinguishable encryptions. The converse is also true [22].

Theorem 3.1. *Let* Π *be a public key cryptosystem. If* Π *has indistinguishable encryptions, then* Π *is semantically secure.*

Proof. Suppose we have an adversary $A = (A_1, A_2)$ against semantic security with significant advantage. We shall turn it into an adversary $A' = (A'_1, A'_2)$ against indistinguishable encryptions. A'_1 runs A_1 and receives (X, f). It samples m_0, m_1 from X such that $f(m_b) = b$. This can be done efficiently with arbitrarily small probability of failure.

When A'_2 receives the challenge ciphertext c, which is an encryption of m_b, it runs A_2 and receives its guess b' for $f(m_b) = b$. Since the message m_b has been sampled according to X, the input to A_2 is exactly as it expects. It therefore guesses $f(m_b) = b$ correctly with a significant advantage, hence A'_2 guesses b correctly with significant advantage. □

A second alternative security goal is the following. The "find"-stage outputs the probability space X given by

$$\Pr[X = m] = \begin{cases} \frac{1}{2} & m = m_0, \\ \frac{1}{|pk_M|-1} & \text{otherwise}, \end{cases}$$

where $m_0 \in pk_M$, and the function f is defined to be $f(m_0) = 1$ and $f(m) = 0$, when $m \neq m_0$.

The adversary's "guess"-stage must then decide if the challenge ciphertext decrypts to m_0 or not.

Definition. We say that Π has *encryptions indistinguishable from random* if any efficient adversary as described above has probability negligibly different from $1/2$ in guessing if the message m_0 was encrypted.

It is quite obvious that if Π is semantically secure, then Π has encryptions indistinguishable from random. The converse is also true.

Theorem 3.2. *Let* Π *be a public key cryptosystem. If* Π *has encryptions indistinguishable from random, then* Π *is semantically secure.*

Proof. Suppose we have an adversary $A = (A_1, A_2)$ against indistinguishable encryptions with significant advantage ϵ. We shall turn it into an adversary $A' = (A'_1, A'_2)$ against encryptions indistinguishable from random. The result will then follow by Theorem 3.1 if A' has significant advantage.

A'_1 runs A_1 and receives $m_0, m_1 \in pk_M$. It flips a coin b and outputs m_b. A'_2 runs A_2 and receives b'. If $b' = b$, A'_2 outputs 1, otherwise it outputs 0.

The ciphertext given to A'_2 is either an encryption of m_b, or of some other random message. If the challenge ciphertext decrypts to m_b, it is clear that A_2 guesses b correctly with probability $1/2 + \epsilon$.

If the challenge ciphertext does not decrypt to m_b, we have two possibilities. If it decrypts to m_{1-b} (which happens with probability $1/(|pk_M|-1)$), A_2 guesses

b correctly with probability $1/2 - \epsilon$. If the challenge ciphertext does not decrypt to m_{1-b} (which happens with probability $(|pk_M| - 2)/(|pk_M| - 1)$, A_2 has no information about b and guesses correctly with probability $1/2$.

Summing up, we get

$$\begin{aligned}\mathrm{Adv}_{A'} &= \frac{1}{2}|\Pr[A_2' = 1 \mid \mathcal{D}(c) = m_b] - \Pr[A_2' = 1 \mid \mathcal{D}(c) \neq m_b]| \\ &= \frac{1}{2}\left|\frac{1}{2} + \epsilon - \left(\frac{1}{2} - \epsilon\right)\frac{1}{|pk_M| - 1} - \frac{1}{2}\frac{|pk_M| - 2}{|pk_M| - 1}\right| \\ &= \frac{1}{2}\epsilon\frac{|pk_M|}{|pk_M| - 1},\end{aligned}$$

which concludes the proof. □

From the proof, we see that if $|pk_M| = 2$, the two notions indistinguishable encryptions and encryptions indistinguishable from random coincide, as they should.

A different security goal is *non-malleability*. Briefly, it says that it is difficult to modify a ciphertext in such a way that the decryption of the modified ciphertext is predictably related to the original ciphertext. We shall not work with non-malleability, and we only remark that no homomorphic cryptosystem can be non-malleable.

3.4 Security notions

The security goals are independent of the adversary models, so we can mix them freely to form *security notions* [3]. It is now interesting to ask if a cryptosystem that satisfies some security notion also satisfies other security notions.

It is clear that if a cryptosystem achieves some security goal against some adversary class, then it achieves the same security goal against weaker adversary classes. It is also easy to show that any cryptosystem that achieves non-malleability against some adversary class also achieves semantic security against the same adversary class.

For a general-purpose public key cryptosystem, one wants both semantic security and non-malleability against adaptive chosen ciphertext adversaries. This ensures that an adversary cannot learn anything about the message encrypted, and he cannot tamper with the ciphertext (the ciphertext "envelope" containing the message is both opaque and tamper-proof).

It has been shown [3] that any cryptosystem that is semantically secure against chosen ciphertext adversaries also achieves non-malleability against chosen ciphertext adversaries. Since non-malleability is somewhat difficult to work with, this result simplifies our work.

Since no homomorphic public key cryptosystem can be non-malleable, semantic security against non-adaptive chosen ciphertext adversaries is the strongest security notion a homomorphic cryptosystem can achieve.

We shall use semantic security as a shorthand for semantic security against chosen plaintext attack, and abbreviate this by IND. We shall say that a cryptosystem is secure against (non-adaptive) chosen ciphertext attacks if it is semantically secure against (non-adaptive) chosen ciphertext attacks, and abbreviate this by CCA1 and CCA2[1] .

3.5 Key encapsulation methods

This presentation is based on [11].

While sending short messages has many applications, quite often one wants to send a long message using public-key methods. Since public key cryptosystems are quite slow, sending long messages would take a long time. Usually, symmetric (or private key) cryptosystems are much faster than public key cryptosystems. Hence, it could be interesting to combine a symmetric cryptosystem with a public key cryptosystem.

One could do this by using the public key cryptosystem to encrypt a key for a symmetric cryptosystem, and then encrypt the message using the symmetric cryptosystem. But the keys used for a symmetric cryptosystem are in general chosen uniformly at random from some key space. Because of this, there may be better approaches to designing public key cryptosystems for long messages.

A *(one-time) symmetric cryptosystem* Σ consists of two algorithms $\mathcal{SE}$ and $\mathcal{SD}$, a set of symmetric keys Σ_K, a set of messages Σ_M and a set of ciphertexts Σ_C. The *symmetric encryption algorithm* $\mathcal{SE}$ takes as input a symmetric key and a message, and outputs a ciphertext. The *symmetric decryption algorithm* $\mathcal{SD}$ is deterministic, takes as input a symmetric key and a ciphertext, and outputs a message or the special symbol $\perp$. We require that for all message $m \in \Sigma_M$ and $k \in \Sigma_K$, $\Pr[\mathcal{SD}(k, \mathcal{SE}(k, m)) = m] = 1$.

Note that the length of distinct messages may be different, and the length of the ciphertexts may depend on the length of the message.

We say that a symmetric cryptosystem is secure against *passive attacks* if any adversary that first selects two messages m_0 and m_1 (of equal length), receives an encryption of either message under some secret key, and then tries to guess which message was encrypted, guesses correctly with negligible advantage. Note that the adversary only receives a single ciphertext under any one key, hence "one-time".

A symmetric cryptosystem is secure against *active attacks* if the advantage of any adversary that is allowed to decrypt any ciphertext under the secret key (except, of course, the challenge ciphertext), is negligible.

Typical symmetric cryptosystems include the one-time pad, block ciphers in a secure mode of encryption, and general stream ciphers. To protect against active

[1] Historically, the non-adaptive attack was proposed first, and called *chosen ciphertext attack*, denoted by CCA. Then the adaptive attack was proposed, and people took to denoting the non-adaptive attack as CCA1 and the adaptive attack as CCA2.

attacks, a message authentication code (MAC) is usually added, following the encrypt-then-authenticate paradigm.

Definition. A *key encapsulation method* (KEM) Λ consists of three algorithms, $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$. $\mathcal{K}$ is an efficient algorithm that takes a *security parameter* as input and outputs a *public key* pk and *private key* sk. The public key specifies among other things a finite set of possible symmetric keys and a finite set of possible ciphertexts, denoted by pk_K and pk_C.

$\mathcal{E}$ is an efficient algorithm that takes a public key as input and outputs a symmetric key and a ciphertext encapsulating the symmetric key. $\mathcal{D}$ is a deterministic polynomial-time algorithm that takes a private key and a ciphertext as input and outputs either a symmetric key or the special symbol $\perp$. We require that for any public-private key pair (pk, sk), $\Pr[\mathcal{D}(sk, c) = k \mid (k, c) \leftarrow \mathcal{E}(pk)] = 1$.

The intuitive notion is that the key encapsulation method is secure if it is impossible to distinguish the encapsulated symmetric key from a symmetric key chosen uniformly at random. We use the following experiment to measure the advantage of an adversary.

Experiment 2.
Input: $\Lambda = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, A, τ.

1. $(pk, sk) \leftarrow \mathcal{K}(\tau)$.
2. $(x, c) \leftarrow \mathcal{E}(pk)$.
3. $b \leftarrow \{0, 1\}$.
4. If $b = 1$, then $z \leftarrow x$, otherwise $z \leftarrow pk_K$.
5. $b' \leftarrow A(pk, c, z)$.
6. If $b = b'$, output 1, otherwise output 0.

Output: 0 or 1.

We define the advantage of an adversary A to be

$$\mathrm{Adv}_A(\tau) = |\Pr[\mathbf{Exp2}(\Lambda, A, \tau) = 1] - 1/2|.$$

In a chosen ciphertext attack, we give the adversary access to a restricted decryption oracle. (We do not distinguish between adaptive and non-adaptive chosen ciphertext attacks.)

We say that a key encapsulation method is semantically secure if any efficient passive adversary has negligible advantage, and it is secure against chosen ciphertext attacks if any efficient chosen ciphertext adversary has negligible advantage.

A *hybrid cryptosystem* is a pair (Λ, Σ). The key generation algorithm takes as input a security parameter, and outputs a public key for Λ such that $pk_K = \Sigma_K$. The encryption and decryption algorithms are given in Figure 3.1. Depending on the symmetric cryptosystem, the range of the security parameter for the key generation algorithm may be restricted.

The security goals and adversary models for hybrid cryptosystems are as for public key cryptosystems. It is possible to prove the following theorem.

Encryption.
Input: pk, $m \in \Sigma_M$.

1. $(k, c) \leftarrow \mathcal{E}(pk)$.
2. $c' \leftarrow \mathcal{SE}(k, m)$.
3. Output (c, c').

Output: A hybrid ciphertext in $pk_C \times \Sigma_C$.

Decryption.
Input: sk, $(c, c') \in pk_C \times \Sigma_C$.

1. $k \leftarrow \mathcal{D}(sk, c)$.
2. If $k = \perp$, output $\perp$.
3. $m \leftarrow \mathcal{SD}(k, c')$.
4. Output m.

Output: A message $m \in \Sigma_M$ or the special symbol $\perp$.

Figure 3.1: Encryption and decryption algorithms for hybrid cryptosystems.

Theorem 3.3. *The hybrid cryptosystem (Λ, Σ) is semantically secure if Λ is semantically secure and Σ is secure against passive attacks. (Λ, Σ) is secure against chosen ciphertext attacks if Λ is secure against chosen ciphertext attacks and Σ is secure against active attacks.*

3.6 Models for provable security

In general, we want to prove the security of a cryptosystem using as few assumptions as possible. If we place no limitations on the adversary, except that imposed by the computational model and certain bounds on computational power, we are working in the real world, and we call this the *standard model*.

3.6.1 Random oracle model

A *hash function* is a function that takes as input arbitrary binary strings and outputs a fixed-length bit string. We are interested in a special class of hash functions with desirable cryptographic properties.

Definition. Let $h : \{0,1\}^* \rightarrow \{0,1\}^s$ be a hash function. If x_0 and x_1 are distinct bit strings such that $h(x_0) = h(x_1)$, we say that the pair (x_0, x_1) is a *collision* for h. A *collision resistant hash function* is a hash function such that the problem of finding collisions is hard.

There are families of hash functions that are provably collision resistant, but they tend to be very inefficient and are not interesting in practice.

Several very efficient hash functions have been proposed, that seem to be collision resistant, such as the SHA-i [19]. The interesting thing about these hash functions is that they seem to behave as *random functions*. Informally, we can say that the only way to get any information about the value of a random function at some point is to evaluate the function.

It seems difficult to capture this notion in the standard model, so the *random oracle model* was formulated [5]. The idea is to prove security using random functions, and then replace the random functions with hash functions like SHA-i.

Let S and S' be two sets, and let $\mathcal{H}$ be the set of all functions from S to S'. A *random oracle* from S to S' is an oracle that is made available to all algorithms and oracles that participate in an experiment. At the start of the experiment, the oracle is initialised by choosing a function h from $\mathcal{H}$ uniformly at random. Whenever the random oracle receives a query x, it computes $h(x)$ and returns the result. The other algorithms and oracles participating in the experiment can only get information about h by querying the oracle.

An alternative formulation that is often convenient, is a description of a random oracle in terms of an algorithm. The oracle algorithm accepts as queries elements of the set S. For the first query x the oracle receives, the oracle samples an element y of S' uniformly at random. It then stores the pair (x, y) in a sorted list, and returns the answer y.

For each subsequent query x', the oracle checks if it has a pair (x', y') in its list. If it does have such a pair, it returns the answer y'. If it does not have such a pair, it samples an element y' of S' uniformly at random, stores the pair (x', y') in the sorted list, and returns the answer y'.

It is quite clear that these two formulations are equivalent.

While the random oracle model is quite popular, a proof of security in the random oracle model does not imply that the cryptosystem is secure when the random oracle is replaced with a real hash function. While the security certainly depends on what hash function is used, there are also schemes [2] that are provably secure in the random oracle model, but are insecure no matter what hash function is used.

Therefore, a proof of security in the random oracle model should only be taken as heuristic evidence for the security of the system, and great care should be taken when instantiating the system.

3.6.2 Generic model

It is interesting to study algorithms that operate on arbitrary finite abelian groups. These are often called *generic algorithms*. Abstractly, a finite abelian group is isomorphic to a product of different $\mathbb{Z}_{p^r}$, but for some groups, this isomorphism seems to be difficult to compute. Shoup [39] showed that no generic algorithm can compute that isomorphism efficiently.

A consequence is that when the set Γ_τ is the set of abstract groups of prime order p, where p is a τ-bit prime and $\tau > 1$ and $\Gamma = \{\Gamma_\tau\}$, then the discrete logarithm problem $\mathcal{DL}_\Gamma$ is provably hard. This shows that there can be no generic attacks on the discrete logarithm problem, but it does not say anything about how difficult the discrete logarithm problem is for real families of groups.

The *generic model* works as follows. Let $\mathbb{Z}_n$ be the group structure, and let $\mathcal{H}$ be the set of 1-1 functions from $\mathbb{Z}_n$ into a finite set S, where $|S| \geq n$. At the start of the experiment, a group oracle is initialised with a function $f \in \mathcal{H}$ chosen

uniformly at random.

The group oracle accepts three types of queries. The first query type consists of the special symbol $\perp$, and the oracle responds with $f(0)$ and $f(1)$. The second query type consists of a single element $s \in S$. If $s \notin f(\mathbb{Z}_n)$, then the group oracle answers $\perp$, otherwise it returns $f(-f^{-1}(s))$ (note that $f^{-1}(s)$ is well-defined for $s \in f(\mathbb{Z}_n)$). The third query type consists of a pair $(s_1, s_2) \in S \times S$. If $s_1 \notin f(\mathbb{Z}_n)$ or $s_2 \notin f(\mathbb{Z}_n)$, then the answer is $\perp$, otherwise the answer is $f(f^{-1}(s_1)+f^{-1}(s_2))$.

Every algorithm and oracle in the experiment is given access to this group oracle.

As for the random oracle model, we can also describe the group oracle in terms of an algorithm.

The algorithm keeps two lists: one list of pairs $\{(s, l)\} \subseteq S \times \mathbb{Z}_n$, sorted on both coordinates, and one list of "forbidden" elements $\{s\} \subseteq S$. The first list will be a partial definition of a 1-1 function $f : \mathbb{Z}_n \to S$, so we call it the f-list. The second list will contain a partial list of $S \setminus f(\mathbb{Z}_n)$.

If $s \in S$ is received in a query, and s is in the forbidden list, $\perp$ is returned immediately.

Whenever $s \in S$ is received in a query, there is no pair (s, l) in the f-list, and s is not in the "forbidden" list, the algorithm flips a biased coin. The coin is 0 with probability $(n - N)/(|S| - N)$, where N is the number of elements in the f-list. If the coin flip gives 0, s is stored in the "forbidden" list and $\perp$ is returned. If the coin flip gives 1, l is sampled uniformly at random from $\mathbb{Z}_n$ such that there is no pair (s', l) in the f-list, and then (s, l) is added to the f-list, and the query proceeds as usual.

Otherwise, the query is processed as usual, and the algorithm should return $f(l)$ for some $l \in \mathbb{Z}_n$. If there is no pair (s', l) in the f-list, then s' is sampled uniformly from S such that there is no pair (s', l') in the f-list, and such that s' is not in the forbidden list. Then (s', l) is added to the f-list, and s' is returned.

It is clear that the two descriptions of the group oracle are equivalent.

Quite often, cryptographic systems are formulated in terms of generic abelian groups. In this case, it may make sense to prove that the system is secure against generic algorithms. As for the random oracle model, this does not actually imply that a concrete instantiation of the generic system is secure. But it does provide some heuristic evidence (albeit weaker than a proof in the random oracle model) for the security of concrete instantiations.

3.7 Plaintext aware encryption

We shall discuss one technique that is often used to prove security. The basic idea is that if the adversary is unable to create ciphertexts without knowing their decryption, a decryption oracle is useless and (non-adaptive) chosen ciphertext attacks reduce to chosen plaintext attacks.

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public key cryptosystem. A *forger for* Π is an algorithm B that takes as input a public key pk and outputs a ciphertext c. We give B access

to a special encryption oracle $\mathcal{E}_{pk}$. When it is first queried with a probability space X on pk_M, it samples a message m_0 according to X, a ciphertext c_0 according to $\mathcal{E}(pk, m)$, and returns the answer c_0. Any subsequent query is answered with the special symbol $\perp$.

It is quite clear that we can modify any such forger to also output c_0 in addition to c.

A *knowledge extractor for* Π is an algorithm D that on input of a public key pk, a ciphertext c_0, a trace and a challenge ciphertext c, outputs a message $m \in pk_M$ or the special symbol $\perp$. The idea is that the trace should be enough for a knowledge extractor to decrypt the ciphertext.

Removing the oracle $\mathcal{E}_{pk}$ and the ciphertext c_0 from the above gives us definitions for *weak forger* and *weak knowledge extractor*.

We use the following experiment to measure the success probability of a knowledge extractor. (The weak variant is obvious.)

Experiment 3.
Input: $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, B', D, τ.

1. $(pk, sk) \leftarrow \mathcal{K}(\tau)$.
2. Initialise the $\mathcal{E}_{pk}$ oracle.
3. $(c, c_0, tr) \leftarrow B'^{\mathcal{E}_{pk}}(pk)$.
4. If $c = c_0$, output 0 and stop.
5. $m \leftarrow D(pk, c_0, tr, c)$.
6. $m' \leftarrow \mathcal{D}(sk, c)$.
7. If $m = m'$, output 1, otherwise output 0.

Output: 0 or 1.

Now we can say exactly what it means that a public key cryptosystem is plaintext aware.

Definition. A public key cryptosystem Π is *(weakly) plaintext aware* if there exists a (weak) knowledge extractor D such that for any (weak) forger B for Π, there exists a trace-variant B' of B such that

$$\Pr[\mathbf{Exp3}(\Pi, B', D, \tau) = 0]$$

is negligible in τ.

The following theorem can then be proven (as in [3]).

Theorem 3.4. *Let* Π *be a public key cryptosystem that is semantically secure and (weakly) plaintext aware. Then* Π *is secure against (non-adaptive) chosen ciphertext attacks.*

The main idea of the proof is that every time the adversary queries its decryption oracle, it is in effect a (weak) forger. This means that there exists a trace-variant that makes the same query, but where the query includes the trace. The decryption oracle can then use the trace along with the knowledge extractor to do the decryption.

The requirement that Π is semantically secure is often included in the definition of plaintext awareness.

Plaintext awareness is especially interesting in the random oracle model. It is quite obvious that we can modify any algorithm in the random oracle model to output, in addition to its normal output, a complete list of any queries it has made to any random oracles it has access to. So in the random oracle model, the list of random oracle queries is a very interesting trace for a forger to output.

Chapter 4

Subgroup membership problems

As we have seen, the strongest security goals are formulated in terms of indistinguishability. There is a class of general mathematical problems, called subgroup membership problems, that has a very natural indistinguishability-property.

4.1 Subgroup membership problems

Let Γ_τ be a probability space on a set of pairs (G, K), where G is a finite abelian group and K is a subgroup of G. For each pair (G, K), define a probability space X_G on G such that X_G when restricted to K or $G \setminus K$ is uniformly distributed, and

$$\Pr[x \in K \mid x \leftarrow X_G] = \Pr[x \in G \setminus K \mid x \leftarrow X_G] = \frac{1}{2}.$$

Let $\Gamma = \{\Gamma_\tau\}$.

We assume that an efficient algorithm is available for Γ that samples K and $G \setminus K$ (almost) uniformly at random, and we denote sampling x from the resulting probability space by $x \overset{\approx}{\leftarrow} K$ and $x \overset{\approx}{\leftarrow} G \setminus K$.

Definition. The *subgroup membership problem* $\mathcal{SM}_\Gamma$ derived from the family Γ described above is the following problem: The instances are triples (G, K, x), where (G, K) is sampled from Γ_τ and x is sampled according to X_G. The answer to the instance (G, K, x) is 1 if $x \in K$, otherwise 0.

We use the word *distinguisher* for an algorithm that solves a subgroup membership problem.

We denote by $\mathcal{SM}_{(G,K)}$ the subproblem of $\mathcal{SM}_\Gamma$ where the instances are restricted to triples (G, K, x), with $x \in G$. It is clear that if we can show that something holds for an arbitrary subproblem $\mathcal{SM}_{(G,K)}$, then it also holds for $\mathcal{SM}_\Gamma$. (Note that the subgroup membership problem is equivalent to deciding if the residue of x is the neutral element in the factor group G/K.)

It is easy to see that for $\mathcal{SM}_{(G,K)}$, we have that

$$\mathrm{Adv}_A^{\mathcal{SM}_{(G,K)}} = \frac{1}{2}|\Pr[A(G,K,x)=1 \mid x \leftarrow K] - \Pr[A(G,K,x)=1 \mid x \leftarrow G \setminus K]|.$$

Proposition 4.1. *Suppose $\mathcal{SM}_{(G,K)}$ is a subgroup membership problem, that the factor group G/K is cyclic and $|G/K|$ has no small prime factors, or that $|G/K|$ has known, prime order. Then $\mathcal{SM}_{(G,K)}$ is self-reducible.*

Proof. If $|G/K|$ has no small prime factors, let t_0 be such that $|G/K| < 2^{t_0}$. Let U be uniformly distributed on the set $\{0,\ldots,2^{t_0+t}-1\}$, for some $t > 0$. By Lemma 2.3, U mod $|G|$ is 2^{-t}-close to uniformly distributed on $\{0,\ldots,|G|-1\}$.

Alternatively, if $|G/K|$ has known, prime order ℓ, then let U be uniformly distributed on $\{1,\ldots,\ell-1\}$.

Let f_u be the map $x \mapsto x^u$. Sampling u from U samples f_u (almost) uniformly from the set of automorphism on G/K. (With small probability, it could be something other than an automorphism.)

The algorithm takes as input $x \in G$. It samples u from U and x' from K, and computes $x'' = x'x^u$, then it outputs x'' and the identity function.

If the input $x \in K$, then it is clear that $x'' \in K$ and its distribution is (almost) uniform and independent of x.

If $x \in G \setminus K$, then except with small probability, the residue class of x in G/K is a generator, and so x^U is (almost) uniformly distributed in G/K. It then follows that $x^U x'$ is (almost) uniformly distributed in $G \setminus K$ and it is independent of x. □

Proposition 4.2. *Let G be a group and let $K' \subseteq K \subseteq G$ be proper subgroups. If A can distinguish K' from G with advantage ϵ, then either A can distinguish K' from K with advantage greater than or equal to $\epsilon/2$, or it can distinguish K from G with advantage greater than or equal to $\epsilon/3$. Alternatively,*

$$\mathrm{Adv}_A^{\mathcal{SM}_{(G,K')}} \leq \mathrm{Adv}_A^{\mathcal{SM}_{(G,K)}} + \frac{3}{2}\mathrm{Adv}_A^{\mathcal{SM}_{(K,K')}}.$$

Proof. Set $X_0 = K'$, $X_1 = K \setminus K'$ and $X_2 = G \setminus K$. Let $wt(X) = \Pr[A(G,K',x) = 1 \mid x \in X]$. We may without loss of generality assume that $\epsilon > 0$ and that

$$\begin{aligned} 2\epsilon &= wt(X_1 \cup X_2) - wt(X_0) \\ &= \frac{|X_1|}{|X_1|+|X_2|}wt(X_1) + \frac{|X_2|}{|X_1|-|X_2|}wt(X_2) - wt(X_0). \end{aligned}$$

Setting $2\delta_0 = wt(X_1) - wt(X_0)$ and $2\delta_1 = wt(X_2) - wt(X_0 \cup X_1)$, and using the fact that

$$wt(X_0 \cup X_1) = \frac{|X_0|}{|X_0|+|X_1|}wt(X_0) + \frac{|X_1|}{|X_0|+|X_1|}wt(X_1)$$

we get that

$$wt(X_2) = 2\delta_1 + \frac{|X_1|}{|X_0| + |X_1|} 2\delta_0 + wt(X_0).$$

This becomes

$$\begin{aligned}\epsilon &= \frac{|X_2|}{|X_1| + |X_2|}\delta_1 + \left(\frac{|X_2|}{|X_1| + |X_2|}\frac{|X_1|}{|X_0| + |X_1|} + \frac{|X_1|}{|X_1| + |X_2|}\right)\delta_0 \\ &\leq |\delta_1| + \frac{3}{2}|\delta_0|.\end{aligned}$$

Finally, we note that $|\delta_0|$ is A's advantage in distinguishing $K \setminus K'$ from K' and $|\delta_1|$ is A's advantage in distinguishing $G \setminus K$ from K', which concludes the proof. □

Proposition 4.3. *Let $\mathcal{SM}_{(G,K)}$ and $\mathcal{SM}_{(G',K')}$ be subgroup membership problems, and let X, X', Y, Y' be the uniform distributions on K, K', $G \setminus K$ and $G' \setminus K'$, respectively. Suppose there is some efficient algorithm B that takes elements of G into elements of G', such that $B(X)$ and X' are ϵ-close and $B(Y)$ and Y' are ϵ-close.*

Then any distinguisher A' for $\mathcal{SM}_{(G',K')}$ can be turned into a distinguisher A for $\mathcal{SM}_{(G,K)}$, and we have that

$$\mathrm{Adv}_{A'}^{\mathcal{SM}_{(G',K')}} \leq \mathrm{Adv}_{A}^{\mathcal{SM}_{(G,K)}} + \epsilon.$$

Proof. Let A' be a distinguisher for $\mathcal{SM}_{(G',K')}$. Let A be the algorithm that on input of G, K and x simply samples y from $B(x)$, runs A' on input of G', K' and y and returns the output of A'.

For brevity, we suppress the input G, K, G' and K' to A and A'. Note that by Lemma 2.1,

$$\begin{aligned}&|\Pr[A(x) = 1 \mid x \leftarrow K] - \Pr[A'(y) = 1 \mid y \leftarrow K']| \\ &\qquad = |\Pr[A'(y) = 1 \mid y \leftarrow B(K)] - \Pr[A'(y) = 1 \mid y \leftarrow K']| \leq \epsilon.\end{aligned}$$

The same holds for $G \setminus K$ and $G' \setminus K'$. Hence we get

$$\begin{aligned}\mathrm{Adv}_{A'}^{\mathcal{SM}_{(G',K')}} &= \frac{1}{2}|\Pr[A'(y) = 1 \mid y \leftarrow K'] - \Pr[A(y) = 1 \mid y \leftarrow G' \setminus K']| \\ &\leq \frac{1}{2}|\Pr[A'(x) \mid x \leftarrow K] - \Pr[A(x) \mid x \leftarrow G \setminus K]| + \epsilon \\ &= \mathrm{Adv}_{A}^{\mathcal{SM}_{(G,K)}} + \epsilon,\end{aligned}$$

which concludes the proof. □

4.2 Splitting problems

Let Γ_τ be a probability space on a set of triples (G, K, H), where G is a finite abelian group, and K and H are subgroups of G such that $G = KH$ and $K \cap H = \{1\}$. Let $\Gamma = \{\Gamma_\tau\}$.

We assume that an efficient algorithm is available for Γ that samples G (almost) uniformly at random, and we denote sampling x from the resulting probability space by $x \overset{\approx}{\leftarrow} G$.

For any such triple $(G, K, H) \in \Gamma_\tau$, any element $x \in G$ can be written uniquely as $x = x_1 x_2$ with $x_1 \in K$, $x_2 \in H$. In other words, there is an isomorphism $G \xrightarrow{\sim} K \times H$. The natural question to ask is if this isomorphism is easy to compute. (The isomorphism $K \times H \xrightarrow{\sim} G$ is simply multiplication.)

Definition. The *splitting problem* $\mathcal{SP}_\Gamma$ derived from the family Γ described above is the following problem: The problem instances are tuples (G, K, H, x), where x is sampled uniformly at random from G. The answer is a pair $(x_1, x_2) \in K \times H$ such that $x = x_1 x_2$.

If the required sampling algorithms for Γ are available, we can derive a subgroup membership problem $\mathcal{SM}_\Gamma$ from $\mathcal{SP}_\Gamma$, by forgetting H. It is obvious that $\mathcal{SM}_\Gamma$ reduces to $\mathcal{SP}_\Gamma$.

Let (G, K, H, x) be an instance of a splitting problem with answer (x_1, x_2). We sometimes say that (x_1, x_2) is the *splitting of* x and that the *projection of* x *on* K*, respectively* H is x_1, respectively x_2. We also use the notation $\pi_K(x)$ and $\pi_H(x)$, for x_1 and x_2, respectively.

As for subgroup membership problems, $\mathcal{SP}_{(G,K,H)}$ denotes a subproblem of $\mathcal{SP}_\Gamma$.

The splitting problem has also been called *projection problem* [31].

Proposition 4.4. *Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem. If there are efficient algorithms available to sample K and H (almost) uniformly at random, then the splitting problem $\mathcal{SP}_{(G,K,H)}$ is self-reducible.*

Proof. Given an instance x, sample z_1 (almost) uniformly from K and z_2 (almost) uniformly from H and output the instance xz_1z_2 and the function $(y_1, y_2) \mapsto (y_1 z_1^{-1}, y_2 z_2^{-1})$. The output is (almost) uniformly distributed in G, and independent of the input. □

Proposition 4.5. *Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem such that* $\gcd(|K|, |H|) = 1$*. Then anyone who knows $|K|$ and $|H|$ can solve the splitting problem using a deterministic and efficient algorithm.*

Proof. If $m = |H|$ and $m' \equiv |H|^{-1} \pmod{|K|}$, then $x_1 = x^{mm'}$, and $x_2 = x x_1^{-1}$. □

Definition. Let Γ_τ be a probability space on a set of tuples (G, K, H, σ), such that $\mathcal{SP}_{(G,K,H)}$ is a splitting problem and σ is an easy to compute function that solves the splitting problem. Let $\Gamma = \{\Gamma_\tau\}$. We get a *trapdoor splitting problem* $\mathcal{TSP}_\Gamma$. We denote the underlying splitting problem by $\mathcal{SP}_\Gamma$.

4.3 Subgroup discrete logarithm problems

For several of the splitting problems we discuss, it will be easy to compute discrete logarithms in H. Because of applications to cryptography, we need to ask if it is easier to compute the discrete logarithm of the projection than to compute the projection itself.

Definition. Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem, and let $g \in G$ be such that $H \subseteq \langle g \rangle$ (hence, H must be cyclic). The *H-subgroup discrete logarithm* of $y \in G$ to the base g is the discrete logarithm of $\pi_H(y)$ to the base $\pi_H(g)$.

Let Γ_τ be a probability space on a set of tuples (G, K, H, g), such that for any tuple (G, K, H), $\mathcal{SP}_{(G,K,H)}$ is a splitting problem. (Note that this implies that there must be an algorithm available for sampling G (almost) uniformly at random.) Let $\Gamma = \{\Gamma_\tau\}$.

Definition. The *subgroup discrete logarithm* problem $\mathcal{SDL}_\Gamma$ derived from the family Γ described above is the following problem: The instances are tuples (G, K, H, g, y), where (G, K, H, g) is sampled from Γ and y is sampled uniformly from G. The answer is $\log_{\pi_H(g)} \pi_H(y)$.

As usual, we shall discuss subproblems $\mathcal{SDL}_{(G,K,H,g)}$. We shall write the subgroup discrete logarithm of $y \in G$ as $\log_{(H,g)} y$.

The subgroup discrete logarithm problem has appeared as the *partial discrete logarithm problem* in the literature.

Definition. Let Γ_τ be a probability space on a set of tuples (G, K, H, g, σ, f'), such that $\mathcal{SDL}_{(G,K,H,g)}$ is a subgroup discrete logarithm problem, σ is an easy to compute function that solves the splitting problem, and f' is an easy to compute function that solves the discrete logarithm problem in H. Let $\Gamma = \{\Gamma_\tau\}$. We get a *trapdoor subgroup discrete logarithm* problem $\mathcal{TSDL}_\Gamma$. We denote the underlying subgroup discrete logarithm problem by $\mathcal{SDL}_\Gamma$.

It is quite obvious that if the element g is in H, and discrete logarithms are easy to compute in H, then the subgroup discrete logarithm problem and the splitting problem are essentially the same.

The subgroup discrete logarithm problem is clearly very similar to the discrete logarithm problem.

It is easy to show that if one can compute subgroup discrete logarithms to some base g, then one can compute $|H|$. It is also easy to show that if one can compute subgroup discrete logarithms to one base, one can compute discrete logarithms to any base.

Suppose the order $|H|$ is known. If X is distributed uniformly on $\{0, \ldots, |H|-1\}$ and Z is distributed uniformly on K, then $yg^X Z$ is distributed uniformly on G, it is independent of y and $\log_{(H,g)} yg^X Z \equiv X + \log_{(H,g)} y \pmod{|H|}$. This means that the subgroup discrete logarithm problem is self-reducible when the order of H is known.

We shall prove two results about the subgroup discrete logarithm problem. There are similar results for the discrete logarithm problem. Note that these reductions may or may not be useful in practice, since there is a significant increase in the work required. But since several of our bounds are quite weak, much better results could perhaps be obtained in practice, depending on the specific properties of the algorithm found.

Proposition 4.6. *Let $\mathcal{SDL}_{(G,K,H,g)}$ be a subgroup discrete logarithm problem, and let H' be a prime-order subgroup of H, $|H'| = \ell$ and ℓ^k the biggest power of ℓ that divides H. Given the order $|H|$ and an oracle A that distinguishes K from $KH' \setminus K$ with advantage ϵ, and such that*

$$\Pr[A(x) = 1 \mid x \leftarrow KH' \setminus K] = \alpha,$$

we can for any $0 < \beta < 1$ recover the subgroup discrete logarithm modulo ℓ^k with probability $1 - k\ell\beta$ making

$$\left\lceil \frac{4k}{\beta\epsilon^2} \max\left\{\alpha(1-\alpha), (\alpha+\epsilon)(1-\alpha-\epsilon)\right\} \right\rceil$$

queries to A.

Proof. Let H' be a subgroup of H of prime order ℓ, and let $h = g^{|H|/\ell|}$. By Proposition 4.1, $\mathcal{SM}_{(KH',K)}$ is self-reducible, so we may assume that $\Pr[A(x) = 1 \mid \log_{(H',h)} = i]$ is independent of i when $i = 1, \ldots, \ell - 1$, and that $\Pr[A(x) = 1 \mid x \leftarrow K] = \Pr[A(x) = 1 \mid x \leftarrow KH' \setminus K] + \epsilon$.

Let x be an instance of the subgroup discrete logarithm problem. Raising x to the $|H|/\ell$th power will turn x into an instance x' of the subgroup discrete logarithm problem $\mathcal{SDL}_{(KH',K,H',h)}$. If the subgroup discrete logarithm of x' is m, then $x'h^{-m} \in K$. This means that we can use A to test if $\log_{(H',h)} x \equiv m \pmod{|H'|}$.

Each test is a Bernoulli trial, so let $\alpha = \Pr[A(x) = 1 \mid x \in KH' \setminus K]$. We repeating each trial s times and let X be the number of times 1 occurs. When we test the wrong m, we expect the number of ones to be $s\alpha$, while when we test the correct m, we expect $s(\alpha + \epsilon)$ ones.

When we test the wrong m, we get that

$$\Pr[|X/s - \alpha| < \epsilon/2] \geq 1 - \frac{\alpha(1-\alpha)}{s\epsilon^2/4}.$$

If we fix some maximal failure probability β, we get that

$$s_0 = \frac{\alpha(1-\alpha)}{\beta\epsilon^2/4}.$$

This means that, except with probability β, after so many Bernoulli trials, the proportion of ones will be within $\epsilon/2$ of α.

When we test the correct m, we get that

$$\Pr[|X/s - \alpha - \epsilon| < \epsilon/2] \geq 1 - \frac{(\alpha+\epsilon)(1-\alpha-\epsilon)}{s\epsilon^2/4}.$$

Again, if we fix some maximal failure probability β, we get that

$$s_1 = \frac{(\alpha+\epsilon)(1-\alpha-\epsilon)}{\beta\epsilon^2/4}.$$

So we make a total of $s = \max\{s_1, s_2\}$ tests for each possible value of the subgroup discrete logarithm problem, and choose the value that gives the most ones.

The total probability that this test is successful is at least $(1-\beta)^{\ell-1}(1-\beta) \geq 1 - \ell\beta$.

To conclude the proof, we note that an ℓ-adic lift as used in the Pohlig-Hellman algorithm can be used to recover the discrete logarithm modulo the prime power. □

Suppose $|H|$ is smooth and square-free. Let H' be a prime-ordered subgroup of H, and let H'' be the subgroup of H of order $|H|/|H'|$. It is quite clear that Proposition 4.3 applies. This means that if we can argue that all of the subgroups of H are equal, in the sense that either the subgroup membership problem $\mathcal{SM}_{(G,KH'')}$ is difficult for all prime-order subgroups H' of H, or for none, the subgroup discrete logarithm problem reduces to the subgroup membership problem $\mathcal{SM}_{(G,K)}$.

Let $\mathcal{SDL}_{(G,K,H,g)}$ be a subgroup discrete logarithm problem. Let d be an integer relatively prime to $|H|$, and define the function

$$lsd(x) = (\log_{H,g} x) \bmod d.$$

It is quite clear that lsd has a limited self-reducibility property, in that for any $x \in G$ and $x' \in K$, $lsd(x) = lsd(xx')$. But the success probability of an algorithm for computing lsd may depend on the value of $\log_{H,g} x$, so we need to be a bit more careful.

Let $\mathcal{P} = (P, X, S, f)$ be a problem, and suppose that the set of answers S has a group structure (written additively). Let A be an algorithm that tries to solve $\mathcal{P}$. We say that A has *systematic advantage*

$$\begin{aligned}\mathrm{SAdv}_A^{\mathcal{P}} = \Pr[A(x) = \mathcal{P}(x) \mid x \leftarrow \mathcal{P}] - \\ \max_{s \in S\setminus\{0\}} \Pr[A(x) = \mathcal{P}(x) + s \mid x \leftarrow \mathcal{P}].\end{aligned}$$

Let d be fixed. For a subgroup discrete logarithm problem $\mathcal{SDL}_{(G,K,H,g)}$, we get the problem of computing lsd. For $\mathcal{SDL}_\Gamma$, we define the systematic advantage to be

$$\mathrm{SAdv}_A^{\mathcal{SDL}_\Gamma}(\tau) = \mathrm{E}[\mathrm{SAdv}_A^{\mathcal{SDL}_{(G,K,H,g)}} \mid (G,K,H,g) \leftarrow \Gamma_\tau].$$

Note that if $d = 2$, then systematic advantage reduces to advantage as defined in (2.1).

We shall prove the following result.

Proposition 4.7. *Let $\mathcal{SDL}_\Gamma$ be a subgroup discrete logarithm problem, and let d be a positive integer such that for every subproblem $\mathcal{SDL}_{(G,K,H,g)}$ of $\mathcal{SDL}_\Gamma$, d is relatively prime to $|H|$ and it is easy to find some number e such that $de \equiv 1 \pmod{|H|}$. If there is some efficient algorithm for computing lsd with significant systematic advantage, then $\mathcal{SDL}_\Gamma$ is not hard.*

The case $d = 2$ was proved in [8]. We have a somewhat more general result, so we include the proofs here. We first prove two lemmas. The first says that the systematic advantage gives us a stronger property for a restricted problem.

We remark that even if we do not know the order $|H|$, but we know an integer n such that $\gcd(n, d) = 1$ and $|H|$ divides n, then we can find e by computing inverses modulo n instead of $|H|$.

Lemma 4.8. *Let $\mathcal{SDL}_{(G,K,H,g)}$ be a subgroup discrete logarithm problem, and let A be an algorithm that computes lsd with systematic advantage $\epsilon > 0$.*

Then there is an algorithm A' that computes lsd such that for all $x \in G$ with $\log_{H,g} x \le \lfloor |H|\epsilon/3 \rfloor$,

$$\epsilon/3 \le \Pr[T_0(x)] - \max_{0<s<d} \Pr[T_s(x)],$$

where $T_s(x)$ is the event that $A'(x) \equiv lsd(x) + s \pmod{d}$.

Proof. The algorithm A' takes as input x, samples X uniformly from K and Y uniformly from $\{0, \ldots, |H| - 1\}$ and outputs $(A(Xxg^Y) - Y) \bmod d$.

Let X be uniformly distributed in K, Y be uniformly distributed in $\{0, \ldots, |H| - 1\}$, and let $x \in G$ be such that $\log_{H,g} x \le \lfloor |H|\epsilon/3 \rfloor$. Then $\log_{H,g} Xxg^Y = Y + \log_{H,g} x$ with probability at least $1 - \epsilon/3$, and $\log_{H,g} Xxg^Y = Y + \log_{H,g} x - |H|$ with probability at most $\epsilon/3$.

Let $Z = Xxg^Y$, and set $\alpha = \max_{0<s<d} \Pr[T_s(x) \mid x \leftarrow G]$. We compute

$$\begin{aligned}
\Pr[T_0(Z)] &= \Pr[T_0(Z) \mid Y \le \lfloor |H|\epsilon/3 \rfloor](1 - \epsilon/3) \\
&\quad + \Pr[T_0(Z) \mid Y > \lfloor |H|\epsilon/3 \rfloor]\epsilon/3 \\
&\ge \Pr[T_0(Z) \mid Y \le \lfloor |H|\epsilon/3](1 - \epsilon/3) \\
&\ge \frac{\alpha + \epsilon - \epsilon/3}{1 - \epsilon/3}(1 - \epsilon/3) = \alpha + 2\epsilon/3,
\end{aligned}$$

and

$$\begin{aligned}
\Pr[T_s(Z)] &= \Pr[T_s(Z) \mid Y \le \lfloor |H|\epsilon/3 \rfloor](1 - \epsilon/3) \\
&\quad + \Pr[T_s(Z) \mid Y > \lfloor |H|\epsilon/3 \rfloor]\epsilon/3 \\
&\le \frac{\alpha}{1 - \epsilon/3}(1 - \epsilon/3) + \epsilon/3 = \alpha + \epsilon/3.
\end{aligned}$$

The result follows. □

Now we show that a simple majority vote-type algorithm can amplify the success probability of an oracle for lsd.

Lemma 4.9. *Let A' be the algorithm in Lemma 4.8. Then there is an algorithm A'' that computes lsd with success probability β using less than $O(1/(\epsilon^8(1-\beta)))$ (or $O(1/(\epsilon^3(1-\beta)))$ if d is small) invocations of A'.*

Proof. The algorithm A'' takes as input x and runs A' several times, counting the different answers. Then it returns the most frequent answer.

First of all, we may suppose that 0 is the correct answer. Let X_i, $0 \le i < d$, count the number of times A' returns the value i. Since different runs of A' are independent, $(X_0, X_1, \ldots, X_{d-1})$ will be a multinomially distributed random variable with parameters $(\alpha_0, \alpha_1, \ldots, \alpha_{d-1})$.

By assumption, $\alpha_0 \ge \alpha_i + \epsilon/3$ for $1 \le i < d$. If d is smaller than $O(1/\epsilon)$, the result follows immediately from Lemma 2.6.

So suppose $d > 4/\epsilon$. Let $\{S_1, S_2, \ldots, S_{d'}\}$ be a partition of $\{1, 2, \ldots, d-1\}$, and let $Z_i = \sum_{s \in S_i} X_s$ and $\alpha'_i = \sum_{s \in S_i} \alpha_s$. Let $Z_0 = X_0$ and $\alpha'_0 = \alpha_0$. It is possible to find a partition such that there is at most one $\alpha'_i < \epsilon/4$ and

$$\alpha'_0 \ge \alpha'_i + \epsilon/2, \quad i = 1, \ldots, d' - 1.$$

It is clear that $(Z_0, \ldots, Z_{d'-1})$ is a multinomially distributed random variable with parameters $(\alpha'_0, \ldots, \alpha'_{d'-1})$, and $d' \le 4/\epsilon$. It is also clear that

$$\Pr[X_0 = \max\{X_0, \ldots, X_{d-1}\}] \ge \Pr[Z_0 = \max\{Z_0, \ldots, Z_{d'-1}\}],$$

and the result again follows from Lemma 2.6. □

Proof of Proposition 4.7. Note that if we know $lsd(x)$, then $\log_{H,g} xg^{-lsd(x)}$ is divisible by d. We can divide by d by computing the eth power. To recover every d-adic digit of $\log_{H,g} x$ in this way requires $\lceil \log_d |H| \rceil$ computations of $lsd(x)$.

If every computation of $lsd(x)$ fails with probability γ, then the entire computation is correct with probability $(1-\gamma)^{\lceil \log_d |H| \rceil} \approx 1 - \lceil \log_d |H| \rceil \gamma$, so the maximal failure rate of the lsd computation is polynomial in $\log_d |H|$.

Lemma 4.8 and Lemma 4.9 say that we can compute lsd with sufficiently small failure rate, keeping the number of invocations of A polynomial in $1/\epsilon$ and $\log_d |H|$, if the subgroup discrete logarithm is restricted.

Finally, since the probability that an instance sampled uniformly satisfies this restriction is polynomial in $1/\epsilon$, the self-reducibility property of the subgroup discrete logarithm problem allows us to lift the restriction. This concludes the proof. □

Given an integer d relatively prime to $|H|$, we can write

$$\log_{H,g} x = \sum_i \alpha_i d^i.$$

We can define $lsd_i(x) = \alpha_i$.

Proposition 4.10. *Let $\mathcal{SDL}_{(G,K,H,g)}$ be a subgroup discrete logarithm problem. If there is some efficient algorithm A for computing lsd_i with significant systematic advantage, then there is an algorithm for computing subgroup discrete logarithms when the answers are restricted to be less than $d^{\lfloor \log_d |H| \rfloor - i}$, using A a polynomial (in $\log |H|$) number of times.*

Proof. If we know that $\log_{H,g} x$ is less than d^l, then for any $i < \lfloor \log_d |H| \rfloor - l$, we have that $\log_{H,g} x^{d^i} = d^i \log_{H,g} x$, so $lsd(x) = lsd_i(x^{d^i})$. In order to apply Lemma 4.9 and proceed as in the proof of Proposition 4.7, we need to show how to randomise the queries to the algorithm A.

Following Lemma 4.8, we assume that $\log_{H,g} x \leq \lfloor |H|\epsilon/3 \rfloor$. Then we sample r uniformly at random from $\{0, \ldots, d^i - 1\}$, and r' uniformly at random from $\{d^i, \ldots, |H|-1\}$. It is clear that with probability $1-\epsilon/3$, $d^i \log_{H,g} x + r + r' < |H|$, and that A has systematic advantage at least $\epsilon/3$ on the reduced instance. □

The interesting case is for $d = 2$. From the above proof, it is clear that we can give the algorithm A that tries to compute the ith bit of the subgroup discrete logarithm the lower order bits, since except with probability $\epsilon/3$, they are exactly r. This means that all of the bits are simultaneously hard to compute.

We can consider this informally as a limiting process. As the restriction on the subgroup discrete logarithm decreases, the number of simultaneously hard bits increases. In the limit, where it is difficult to prove that the subgroup discrete logarithm is zero, given the knowledge that it is zero, we get the subgroup membership problem.

4.4 Symmetric subgroup membership problems

Let Γ be defined as in Section 4.2, and suppose that there is an efficient algorithm available that samples K and H (almost) uniformly at random. As usual, sampling using this algorithm is denoted by $x \overset{\approx}{\leftarrow} K$ and $x \overset{\approx}{\leftarrow} H$. It is clear that we can derive two families Γ' and Γ'' from Γ by forgetting in turn H and K, and that we can derive subgroup membership problems from these families.

Definition. The *symmetric subgroup membership problem* $\mathcal{SSM}_\Gamma$ is the double problem $\mathcal{SM}_{\Gamma'}$ and $\mathcal{SM}_{\Gamma''}$. We say that the $\mathcal{SM}_\Gamma$ is *hard* if both $\mathcal{SM}_{\Gamma'}$ and $\mathcal{SM}_{\Gamma''}$ are hard.

Note that $\mathcal{SSM}_\Gamma$ is not a problem in the sense defined in Section 2.2, but it is convenient to borrow the terminology. As for subgroup membership problem, we usually consider subproblems $\mathcal{SSM}_{(G,K,H)}$.

Theorem 4.11. *Let $\mathcal{SSM}_{(G,K,H)}$ be a symmetric subgroup membership problem such that G is cyclic, and let A be an algorithm that decides the Decision Diffie-Hellman problem on G. Then for any $\delta' > 0$ and certain algorithms A_1, A_2, A_3, A'_1, A'_2 and A'_3 that use A once as an oracle and otherwise do $O(\log 1/\delta')$ exponentiations in G, we have that*

$$\mathrm{Adv}_A^{\mathcal{DDH}_G} \leq \mathrm{Adv}_{A_1}^{\mathcal{SM}_{(G,K)}} + \mathrm{Adv}_{A_2}^{\mathcal{SM}_{(G,K)}} + \mathrm{Adv}_{A_3}^{\mathcal{SM}_{(G,H)}} + f(|K|) + O(\delta + \delta'),$$

and

$$\mathrm{Adv}_A^{\mathcal{DDH}_G} \leq \mathrm{Adv}_{A_1'}^{\mathcal{SM}_{(G,H)}} + \mathrm{Adv}_{A_2'}^{\mathcal{SM}_{(G,H)}} + \mathrm{Adv}_{A_3'}^{\mathcal{SM}_{(G,K)}} + f(|H|) + O(\delta + \delta'),$$

where the sampling algorithms for K and H are δ-close to uniform, and

$$f(x) = \frac{|G| - \phi(|G|)}{|G|} + \frac{4x}{|G|} + \frac{x - \phi(x)}{x}.$$

We shall need the following result quite often, so we state it as a separate lemma.

Lemma 4.12. *Let G be a finite cyclic group, and let K and H be non-trivial subgroup of G such that $K \cap H = \{1\}$ and $G = KH$. Let g be a generator for K. Let X be uniformly distributed on $\{0, \ldots, |G|-1\}$, Y be uniformly distributed on G, while Z is uniformly distributed on H. Let $U = (g^X, Y^X)$ and let $V = (g^X, Y^X Z)$. Then*

$$\mathrm{Dist}(U, V) \leq 1 - \frac{\phi(|H|)}{|H|}.$$

Proof. Let $X_1 = X \bmod |K|$ and $X_2 = X \bmod |H|$, and let $Y = Y_1 Y_2$, where $Y_1 \in K$ and $Y_2 \in H$. It is clear that X_1, X_2, Y_1 and Y_2 are independently and uniformly distributed, and that

$$U = (g^{X_1}, Y_1^{X_1} Y_2^{X_2}) \text{ and } V = (g^{X_1}, Y_1^{X_1} Y_2^{X_2} Z).$$

If we let $U' = Y_2^{X_2}$ and $V' = Y_2^{X_2} Z$, it is clear that

$$\mathrm{Dist}(U, V) = \mathrm{Dist}(U', V')$$

and that V' is uniformly distributed on H.

Now we condition on a fixed $y_2 \in H$. If this y_2 is a generator, then it is clear that $y_2^{X_2}$ is uniformly distributed on H. By Lemma 2.5, it follows that

$$\mathrm{Dist}(U', V') \leq 1 - \frac{\phi(|H|)}{|H|},$$

which concludes the proof. □

Proof of Theorem 4.11. A is an algorithm that takes as input G, c_1, c_2, c_3, c_4, and returns 0 or 1. We shall need the experiments listed in Figure 4.1.

The idea is to make small changes to the input distribution of A, interleaved with changes that correspond to the subgroup membership problems. Then we shall show that in the end, A has no advantage against the final input distribution, and the only significant changes in advantage correspond to the subgroup membership problems. Therefore, if A has significant advantage, at least one of the derived algorithms must have significant advantage against a subgroup membership problem.

Experiment 1.
Input: A, G.

1. $u, v, w \leftarrow \{0, \ldots, |G| - 1\}$.
2. $x \leftarrow G$.
3. $b \leftarrow \{0, 1\}$.
4. If $b = 1$, then $z \leftarrow x^{uv}$, otherwise $z \leftarrow x^{w}$.
5. $b' \leftarrow A(x, x^u, x^v, z)$.
6. If $b = b'$, then output 1, otherwise output 0.

Output: 0 or 1.

Experiment 2.
Input: A, G, $x \in G$.

1. $u, v, w \leftarrow \{0, \ldots, |G| - 1\}$.
2. $b \leftarrow \{0, 1\}$.
3. If $b = 1$, then $z \leftarrow x^{uv}$, otherwise $z \leftarrow x^{w}$.
4. $b' \leftarrow A(x, x^u, x^v, z)$.
5. If $b = b'$, then output 1, otherwise output 0.

Output: 0 or 1.

Experiment 3.
Input: A, G, $y \in G$.

1. $u, v, w \leftarrow \{0, \ldots, |G| - 1\}$.
2. $x \leftarrow K$.
3. $b \leftarrow \{0, 1\}$.
4. If $b = 1$, then $z \leftarrow y^{v}$, otherwise $z \leftarrow y^{w}$.
5. $b' \leftarrow A(x, x^v, y, z)$.
6. If $b = b'$, then output 1, otherwise output 0.

Output: 0 or 1.

Experiment 4.
Input: A, G, $h \in G$.

1. $u, v, w \leftarrow \{0, \ldots, |G| - 1\}$.
2. $x \leftarrow K$, $y \leftarrow G$.
3. $b \leftarrow \{0, 1\}$.
4. If $b = 1$, then $z \leftarrow hy^{v}$, otherwise $z \leftarrow y^{w}$.
5. $b' \leftarrow A(x, x^v, y, z)$.
6. If $b = b'$, then output 1, otherwise output 0.

Output: 0 or 1.

Figure 4.1: Experiments for the proof of Theorem 4.11.

First of all, we note that by using the sampling algorithms of the symmetric subgroup membership problem, and by Lemma 2.3, even if we do not know $|G|$, we can implement modified experiments whose underlying probability spaces are $O(\delta+\delta')$-close to the original experiment, for any $\delta' > 0$. The extra work needed is $O(\log 1/\delta')$ exponentiations in G.

Consider first Experiment 1. Let T_1 denote the event that the experiment returns 1. If the x chosen at random is a generator, then it is clear that the experiment runs exactly as Experiment 2 in Section 2.3. By Lemma 2.5, we get that

$$\mathrm{Adv}_A^{\mathcal{DDH}_G} \leq |\Pr[T_1] - 1/2| + \frac{|G| - \phi(|G|)}{|G|}. \tag{4.1}$$

Next we consider Experiment 2. Suppose the input x to Experiment 2 is sampled uniformly from $G \setminus K$. Let T_2 be the event that the experiment returns 1 in this case. The only difference now between Experiment 1 and Experiment 2 is that Experiment 1 samples x uniformly from G, not $G \setminus K$. By Lemma 2.2, we see that

$$|\Pr[T_1] - \Pr[T_2]| \leq \frac{2|K|}{|G|}. \tag{4.2}$$

Now consider the case when the input x to Experiment 2 is sampled uniformly from K, and let T_2' be the event that the experiment returns 1 in this case. It is clear that we can construct an algorithm A_1 for deciding $\mathcal{SM}_{(G,K)}$ from Experiment 2 such that

$$\mathrm{Adv}_{A_1}^{\mathcal{SM}_{(G,K)}} = |\Pr[T_2] - \Pr[T_2']|. \tag{4.3}$$

Next, we consider Experiment 3. Suppose the input y to Experiment 3 is sampled uniformly from K. Let T_3' be the event that the experiment returns 1 in this case. It is clear that if the x sampled is a generator for K, then Experiment 2 and Experiment 3 proceed identically. So

$$|\Pr[T_3'] - \Pr[T_2']| \leq \frac{|K| - \phi(|K|)}{|K|}. \tag{4.4}$$

Let T_3 be the event that the experiment returns 1 when the input y to Experiment 3 is sampled uniformly from $G \setminus K$. Again, it is clear that we can construct an algorithm A_2 for deciding $\mathcal{SM}_{(G,K)}$ from Experiment 3 such that

$$\mathrm{Adv}_{A_2}^{\mathcal{SM}_{(G,K)}} = |\Pr[T_3] - \Pr[T_3']|. \tag{4.5}$$

Finally, we consider Experiment 4. Suppose the input h to Experiment 4 is sampled uniformly from H. Let T_4 be the event that Experiment 4 returns 1 in this case. The only difference between T_4 and T_3 is that in Experiment 3, y is sampled uniformly from $G \setminus K$, while in Experiment 4 it is sampled uniformly from G, and the final result is multiplied by h. By Lemma 2.2 and Lemma 4.12, we have that

$$|\Pr[T_3] - \Pr[T_4]| \leq \frac{2|K|}{|G|} + \frac{|H| - \phi(|H|)}{|H|}. \tag{4.6}$$

Again, we can construct an algorithm A_3 for deciding $\mathcal{SM}_{(G,H)}$ from Experiment 4 such that

$$\mathrm{Adv}_{A_3}^{\mathcal{SM}_{(G,H)}} = |\Pr[T_4] - \Pr[T_4']|. \tag{4.7}$$

If the input h to Experiment 4 is sampled uniformly from $G \setminus H$, then it is clear that the input z to A in Experiment 4 is uniformly distributed in G, no matter what the value of b is. So if T_4' is the event that Experiment 4 returns 1 when h is sampled uniformly from $G \setminus H$, we know that

$$\Pr[T_4'] = \frac{1}{2}. \tag{4.8}$$

Combining equations (4.1)–(4.8) proves the first claim, and the second claim follows by symmetry. □

4.5 A catalogue

We shall discuss several subgroup membership problems that have appeared in the literature, and some new variations on those problems.

All of these subgroup membership problems have corresponding splitting problems. Some of them are interesting subgroup discrete logarithm problems, and some are possibly hard symmetric subgroup membership problems. The splitting problems and subgroup discrete logarithm problems can also be turned into trapdoor problems, and in general Proposition 4.5 allows for efficient splitting.

For every problem, we also discuss how we can sample elements from the relevant subgroups/subsets. We also discuss briefly the relationship with other problems.

4.5.1 Quadratic Residue problem

Let p and q be primes congruent to 3 modulo 4, and let $n = pq$. Let

$$J_n = \{x \in \mathbb{Z}_n^* \mid \left(\frac{x}{n}\right) = 1\}.$$

Let Q_n be the subgroup of quadratic residues of J_n. Then we have a subgroup membership problem $\mathcal{SM}_{(J_n,Q_n)}$ and a splitting problem $\mathcal{SP}_{(J_n,Q_n,\langle -1\rangle)}$. This is the *Quadratic Residue* problem $\mathcal{QR}_n$, which first appeared in [22].

Since $\langle -1\rangle$ has order 2, the subgroup membership problem and the splitting problem are equivalent.

While anyone who knows the order of Q_n can solve the splitting problem, it is quicker to compute the Legendre symbol modulo either p or q.

We can sample elements uniformly from K by sampling elements uniformly from $\mathbb{Z}_n^*$ and squaring them. H consists of 1 and -1, so we can sample uniformly from

H too. Note that if one can sample uniformly and independently from K and H, then one can sample uniformly from both G and $G \setminus K$.

If is obvious that anyone who can factor the modulus n can solve $\mathcal{QR}_n$. It is not known if these are equivalent, but the best known general attack against $\mathcal{QR}_n$ is to factor the modulus.

4.5.2 Higher Residue problem

Let p and q be primes congruent to 3 modulo 4, and let $n = pq$. Suppose c is an odd prime such that $c|p-1$, and $c\nmid q-1$. Then, in an analogue of the quadratic residue problem, we have the *Higher Residue* problem, where H is a subgroup of Q_n of order c. This was first investigated by Cohen Benaloh for non-prime but smooth c ([6] and the references therein). We get a subgroup membership problem and a splitting problem. If c is small, we also get a natural subgroup discrete logarithm problem.

The natural generalisation is to let $p = 2ac + 1$, $q = 2bd + 1$ and $n = pq$, a, b prime, such that $\gcd(ac, bd) = \gcd(ab, cd) = 1$. This was investigated by Naccache and Stern [29]. We let $G = Q_n$, K be the unique subgroup of order ab and H be the subgroup of order cd. To get a natural subgroup discrete logarithm problem, we let c and d be smooth integers. Discrete logarithms in H can then be computed using the Pohlig-Hellman algorithm. It is natural to assume that cd is known. We denote the subgroup membership problem $\mathcal{SM}_{(G,K)}$ by $\mathcal{HR}_n$.

To sample elements of K, we can sample elements uniformly at random from $\mathbb{Z}_n^*$, and then apply the map $x \mapsto x^{2cd}$. Note that this map is an automorphism on K, but on H it is the zero homomorphism.

Suppose we are given an element $x \in H$. Knowing the integer cd and its prime factorisation, finding the order k of x is easy. For any prime ℓ dividing k and any representative m for the residue class x, it is clear that $m^{k/\ell}$ is congruent to 1 modulo either p or q, but not both, so $\gcd(m^{k/\ell} - 1, n)$ gives us the factorisation of n.

Therefore, we cannot give away elements of H. In order to sample elements of $G \setminus K$, we publish a generator g for G. By sampling u uniformly from $\{1, \ldots, cd - 1\}$ and r uniformly from K, rg^u is uniformly distributed in $G \setminus K$.

Under the assumption that c and d are known (not only cd), Naccache and Stern [29] proposed an attack that required approximately

$$O(n^{1/2}/(cd)^2) \tag{4.9}$$

work. This means that if cd is too large, there are better algorithms for factoring the modulus than the general algorithms.

The basic idea of the attack is as follows. We have that $n = 4abcd + 2(ac + bd) + 1$. Then we get that

$$n - 1 \equiv 2ac \pmod{d} \text{ and } n - 1 \equiv 2bd \pmod{c}.$$

In other words, knowledge of c and d gives us some knowledge about a and b that speeds up the search. If c and d are sufficiently small, factoring n seems to be the best attack on the subgroup membership problem $\mathcal{SM}_{(G,K)}$.

Having a generator for H is sometimes desirable. Instead of having c and d be relatively prime, as above, we can let $c = d$, so that $p = 2ac+1$, $q = 2bc+1$, and $n = 4abc^2 + 2c(a+b) + 1$. Note that $c|(n-1)$, so we may as well assume that c is known.

The interesting thing about the subgroup of order c^2 is that it is non-cyclic. If an element $[w]$ of H is made public, and $w \neq 1 \pmod{p}$ and $w \neq 1 \pmod{q}$, then no matter what group operations are performed on $[w]$, the result will never yield a factor of n. It therefore seems safe to make elements of order c public.

Every element of order c generates a subgroup. If we have two elements x and y that generate distinct subgroups, we can use a Pollard ρ-type attack to generate a collision modulo p and not modulo q. This means that we should not publish elements from more than one subgroup.

In general, we will be interested in the modified group structure where K is the subgroup of order ab, H is a cyclic subgroup of order c generated by an element g, and $G = KH$. We denote the subgroup membership problem $\mathcal{SM}_{(G,K)}$ by $\mathcal{HR}'_n$.

As above, we can sample uniformly at random from K by sampling uniformly at random from $\mathbb{Z}_n^*$, then applying the map $x \mapsto x^{2c}$. We publish a generator g for H, so sampling uniformly at random from H is possible.

If we can recover the product ab, then we can factor n given any element m with Jacobi symbol -1 modulo n, because m^{abc} is congruent to 1 modulo one of the prime factors, and -1 modulo the other. Therefore, $\gcd(m^{abc} - 1, n)$ gives the factorisation of n.

First of all, $n - 1 \equiv 0 \pmod{c}$, so the attack of Naccache and Stern described above does not work. We get that

$$\frac{n-1}{4c^2} = ab + \frac{a+b}{2c}.$$

This means that ab is relatively close to $(n-1)/(4c^2)$. Note that $ac \approx bc \approx \sqrt{n}$. We can use a Baby-step Giant-step type algorithm to recover ab in

$$O(\sqrt{(a+b)/c}) = O(\sqrt{\sqrt{n}/c^2}) = O(n^{1/4}/c)$$

steps. This is not bad compared with (4.9).

There are obvious elliptic curve analogues of the above group structures, based on elliptic curves modulo composite integers. Let n be a product of two primes p and q. Find elliptic curves E' and E'' defined over $\mathbb{F}_p$ and $\mathbb{F}_q$, respectively, such that $\#E(\mathbb{F}_p) = ac$ and $\#E(\mathbb{F}_q) = bd$. Then we use the Chinese remainder theorem to derive an elliptic curve E defined over $\mathbb{Z}_n$ such that $\#E(\mathbb{Z}_n) = abcd$.

Hasse's theorem says that $\#E(\mathbb{F}_p) = p+1-t_p$ (t_p is the trace of the Frobenius endomorphism), and that $-2\sqrt{p} \leq t_p \leq 2\sqrt{p}$ (the same holds for q). The trace is distributed roughly uniformly in this range. We get that

$$\begin{aligned}\#E(\mathbb{Z}_n) &= \#E'(\mathbb{F}_p)\#E''(\mathbb{F}_q) = abcd \\ &= (p+1-t_p)(q+1-t_q) \\ &= n - (pt_p + qt_q) + (p+q) - (t_p + t_q) + 1 \\ &= n + O(n^{3/4}).\end{aligned}$$

This means that when cd is known, the product ab can be recovered with a Baby-step Giant-step type algorithm using $O(n^{3/8}/\sqrt{cd})$ steps.

When $c = d$, we repeat the above analysis, and we find that we can recover ab using $O(n^{3/8}/c)$ steps.

4.5.3 Decision Diffie-Hellman problem

The Decision Diffie-Hellman Problem was discussed in Section 2.3. We shall show that the Decision Diffie-Hellman problem is equivalent to a subgroup membership problem [10, 41], and the Computational Diffie-Hellman problem is equivalent to a splitting problem.

Let G' be a cyclic group with generator g_1. Let $G = G' \times G'$. Let g_2 be any element in G'. Then the pair (g_1, g_2) generates a non-trivial, proper subgroup K of G. Let $H = \langle (1, g_1) \rangle$.

If Γ' is a family of groups, we can derive a family of group structures Γ as above. For each group $G = G' \times G'$, there are $|G'|$ different subgroups of the above form, and we sample from them with uniform probability. We get a subgroup membership problem $\mathcal{SM}_\Gamma$ and splitting problem $\mathcal{SP}_\Gamma$.

Suppose we have a problem instance (G, K, w). Now K is defined by two elements (g_1, g_2), and w is defined by a pair (x, y). This gives us a quadruple (g_1, g_2, x, y). It is clear that $(x, y) \in K$ if an only if $\log_{g_1} g_2 \log_g x \equiv \log_g y \pmod{|G'|}$. The quadruple (g_1, g_2, x, y) is a problem instance for the Decision Diffie-Hellman problem $\mathcal{DDH}_{G'}$, and it is sampled with exactly the same probability. This means that $\mathcal{SM}_\Gamma$ is simply a reformulation of $\mathcal{DDH}_{\Gamma'}$.

In the same way, if (G, K, H, w) is an instance of $\mathcal{SP}_G$, we get a quadruple (g_1, g_2, x, y). The answer is two pairs (x, z) and $(1, z')$, such that $\log_{g_1} g_2 \log_{g_1} x \equiv \log_{g_1} z \pmod{|G'|}$. An instance of $\mathcal{CDH}_{G'}$ is a triple (g_1, g_2, x), and the answer is z as described.

It is clear that we can transform an instance of $\mathcal{SP}_{(G,K,H)}$ into an instance of $\mathcal{CDH}_{G'}$ by discarding y. We can transform an instance of $\mathcal{CDH}_{G'}$ into an instance of $\mathcal{SP}_{(G,K,H)}$ (where K may vary) by sampling y uniformly at random from G'.

We see that $\mathcal{SP}_\Gamma$ is simply a reformulation of $\mathcal{CDH}_{\Gamma'}$.

It is worth noting that if K is a subgroup of G generated by (g_1, g_2), and K' is generated by (g_1^a, g_1^b), the map $(x, y) \mapsto (x^a, y^b)$ takes K into K'. The

map is an automorphism on H when $b \not\equiv 0 \pmod{|G'|}$. This corresponds to the self-reducibility property of the Diffie-Hellman problems.

Sampling from $K = \langle (g_1, g_2) \rangle$ can be done by sampling k uniformly at random from $\{0, \ldots, |G'| - 1\}$ and computing $(g_1, g_2)^k$. Sampling uniformly from H can obviously be done by sampling uniformly from G'.

Since G is non-cyclic, Proposition 4.5 cannot be used to solve the splitting problem. But anyone who knows the discrete logarithm $\log_{g_1} g_2$ can compute the splitting, and this shows how to derive a trapdoor splitting problem.

4.5.4 Decision Composite Residuosity problem

This subgroup membership problem was first proposed by Paillier [35]. Our description also incorporates several later simplifications [14].

Let $n = pq$ be such that $\gcd(n, \phi(n)) = 1$. We shall consider the group $G = \mathbb{Z}^*_{n^2}$. It is clear that $|G| = \phi(n)n$.

Let $d \equiv n^{-1} \pmod{\phi(n)}$. Then we can define a map from $\mathbb{Z}^*_n$ into G by $[a]_n \mapsto [a^{dn}]_{n^2}$. It is clear that $a \equiv a^{dn} \pmod{n}$, so this map is one-to-one. It is also an homomorphism. We let the image of $\mathbb{Z}^*_n$ under this map be K.

Its inverse map $G \to \mathbb{Z}^*_n$ is the map $[a]_{n^2} \mapsto [a]_n$, often called reduction modulo n. It is clear that the elements $[1 + an]_{n^2}$ map to $[1]_n$, and that any element that maps to $[1]_n$ is of this form. We let H be this kernel of reduction modulo n.

It is quite clear that $K \cap H = \{1\}$ and that $KH = G$.

Now we note that

$$(1+n)^a \equiv 1 + \binom{a}{1} n + \binom{a}{2} n^2 \cdots \equiv 1 + an \pmod{n^2}.$$

From this, we see that discrete logarithms are easy to compute in H. One group isomorphism $\mathbb{Z}_n \to H$ is the map $[a]_n \mapsto [1+n]^a_{n^2}$. Since for any $[b]_{n^2} \in H$, $b \equiv 1 \pmod{n}$, the reverse isomorphism is $[b]_{n^2} \mapsto [(b-1)/n]_n$.

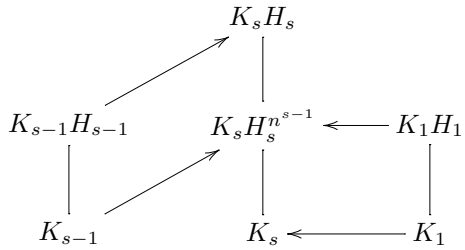
The subgroup membership problem $\mathcal{SM}_{(G,K)}$ is called the *Decision Composite Residuosity* problem ($\mathcal{DCR}_n$). The corresponding splitting problem $\mathcal{SP}_{(G,K,H)}$ is called the *Computational Composite Residuosity* problem ($\mathcal{CCR}_n$). This is equivalent to the subgroup discrete logarithm problem $\mathcal{SDL}_{(G,K,H,[1+n])}$.

We can sample elements uniformly from K and H by sampling elements uniformly in $\mathbb{Z}^*_n$ and $\mathbb{Z}_n$, and then using the corresponding homomorphisms.

It is quite clear that anyone who can solve $\mathcal{FACT}_n$ can solve $\mathcal{CCR}_n$. Likewise, anyone who can solve $\mathcal{RSA}_n$ with encryption exponent n can solve $\mathcal{CCR}_n$. It is not known if $\mathcal{CCR}_n$ is equivalent to either of these, but the best known method for solving the $\mathcal{CCR}_n$ and $\mathcal{DCR}_n$ is to factor the modulus n.

Damgård and Jurik [14] describe a generalised group structure, using $\mathbb{Z}^*_{n^{s+1}}$ for some $s \geq 1$. As above, they show that $G = \mathbb{Z}^*_{n^{s+1}} \simeq \mathbb{Z}^*_n \times \mathbb{Z}_{n^s} \simeq K \times H$. They get a subgroup membership problem $\mathcal{SM}_{(G,K)}$, and they show that it does not get much easier as s increases. We sketch the proof.

Let K_s and H_s be the subgroups of $\mathbb{Z}^*_{n^{s+1}}$ isomorphic to $\mathbb{Z}^*_n$ and $\mathbb{Z}_{n^s}$, respectively. Note that $[1+n]_{n^{s+1}}$ is a generator for H_s. We shall proceed by induction, using the following diagram:



Consider the map $f : \mathbb{Z}^*_{n^2} \to \mathbb{Z}_{n^{s+1}}$ given by $f([w]_{n^2}) = [w^{n^{s-1}}]_{n^{s+1}}$. Since for any $a, b \in \mathbb{Z}$, $(a + bn^t)^{n^{s+1-t}} \equiv a^{n^{s+1-t}} \pmod{n^{s+1-t}}$, and composition with reduction modulo n shows that $f(\mathbb{Z}_{n^2}) \subseteq \mathbb{Z}_{n^{s+1}}$, it is clear that f is a well-defined group homomorphism.

So suppose $x \in H_1$. Then for some $a \in \{0, \ldots, n-1\}$, $x = [(1+n)^a]_{n^2}$ and $f(x) = [((1+n)^a)^{n^{s-1}}]_{n^{s+1}} = [(1+n)^{an^{s-1}}]_{n^{s+1}} \in H_s^{n^{s-1}}$. This means that $f(H_1) = H_s^{n^{s-1}}$.

For any element $x \in K_1$, there must be an integer w such that $x = [w^n]_{n^2}$. But then $f(x) = [(w^n)^{n^{s-1}}]_{n^{s+1}} = [w^{n^s}]_{n^{s+1}}$, so $f(K_1) \subseteq K_s$. It is clear that f is a group homomorphism. Composing with reduction modulo n, and noting that $x \mapsto x^n$ is an automorphism of $\mathbb{Z}^*_n$, we see that the kernel of f restricted to K_1 is trivial, so $f(K_1) = K_s$.

By Proposition 4.3, this means that if we can distinguish $\mathcal{SM}_{(K_sH_s^{n^{s-1}}, K_s)}$, then we can distinguish $\mathcal{SM}_{(K_1H_1, K_a)}$.

Next, we consider the homomorphism $f' : K_sH_s \to K_{s-1}H_{s-1}$ given by $f'([w]_{n^{s+1}}) = [w]_{n^s}$. It is clear that the kernel of f' is $H_s^{n^{s-1}}$. Also, if $w \equiv w' \pmod{n^s}$ (and $\gcd(w, n) = \gcd(w', n) = 1$), then $w \equiv w'(1+n)^{rn^{s-1}} \pmod{n^{s+1}}$ for some $r \in \mathbb{Z}$.

Let A be an algorithm that takes as input $[w]_{n^s} \in K_{s-1}H_{s-1}$, samples r uniformly from $\{0, \ldots, n-1\}$, and outputs $[w(1+n)^{rn^{s-1}}]_{n^{s+1}}$. It is quite clear that if the input is uniformly distributed in K_{s-1}, then the output is uniformly distributed in $K_sH_s^{n^{s-1}}$. Likewise, if $x \in H_{s-1}$ is uniformly distributed in H_{s-1}, then the output is almost uniformly distributed in H_s.

Again, by Proposition 4.3, if we can distinguish $\mathcal{SM}_{(K_sH_s, K_sH_s^{n^{s-1}})}$, then we can distinguish $\mathcal{SM}_{(K_{s-1}H_{s-1}, K_{s-1})}$.

Now we consider only distinguishing algorithms up to a certain fixed cost. Let ϵ_i be the maximal advantage of these algorithms against the subgroup membership problem $\mathcal{SM}_{(K_iH_i, K_i)}$. By Proposition 4.2 and the above discussion, we

get that $\epsilon_i \leq \epsilon_{i-1} + 3\epsilon_1/2$, or $\epsilon_s \leq (3s-1)\epsilon_1/2$. In other words, the advantage increases at most linearly as s increases.

Galbraith [20] gave an elliptic curve generalisation of this group structure. The basic idea is that modulo n^2, the projective point $(n:1:0)$ has order n, but modulo n, it reduces to the point at infinity. So just as for $\mathbb{Z}_{n^2}^*$, $E(\mathbb{Z}_{n^2}) \simeq E(\mathbb{Z}_n) \times \mathbb{Z}_n$, with $\mathbb{Z}_n$ taken additively.

It is easy to sample elements uniformly at random from the part isomorphic to $\mathbb{Z}_n$, but to sample from the part isomorphic to $E(\mathbb{Z}_n)$, we need a generator for $E(\mathbb{Z}_n)$ (or a subgroup, if $E(\mathbb{Z}_n)$ is non-cyclic).

4.5.5 Okamoto-Uchiyama

A precursor to the Composite Residuosity group structure was described by Okamoto and Uchiyama [34].

Let p and q be primes such that $\gcd(pq, \phi(pq)) = 1$, and let $n = p^2q$. As above, we find that $\mathbb{Z}_n^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^* \times \mathbb{Z}_p$, where $\mathbb{Z}_p$ is taken to be an additive group. K is the subgroup isomorphic to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$, and H is the subgroup isomorphic to $\mathbb{Z}_p$.

We can sample elements uniformly from K by sampling uniformly from $\mathbb{Z}_n^*$, and then computing the nth power. Since $\gcd(pq, \phi(pq)) = 1$, raising to the nth power is an automorphism on K, while it is the zero homomorphism on H.

Sampling elements from H is a bit more tricky. Any integer representing a residue class in H is congruent to 1 modulo p. This means that anyone who knows a non-trivial element $[a]$ in H can factor n by computing $\gcd(a-1, n)$.

But given an element g whose order is divisible by abp, we can sample elements almost uniformly at random from $G \backslash K$ by sampling u from $\{0, \ldots, n-1\}$, subject to $\gcd(u, n) = 1$. Then we sample $r \in \mathbb{Z}_n^*$ uniformly at random and compute $r^n g^u$.

Anyone who can solve the subgroup discrete logarithm problem, can recover the order of H, which is p, so he can factor n. Anyone who can factor n can solve the splitting problem. As discussed, anyone who can solve the subgroup splitting problem can factor n and then solve the discrete logarithm problem in H. So the subgroup discrete logarithm problem and the splitting problem are equivalent, and both are equivalent to factoring the underlying modulus.

4.5.6 Groups of known order

Let p and q be primes, and let $n = pq$. Let G be an abelian group of order n. Then G has unique subgroups K, H of order p and q, respectively. If the order of G is easily computable, but the orders of K and H are hard to find, then we have a potential candidate for a subgroup membership problem.

We shall describe two families of such groups. Note that for both families, the two subgroups K and H are essentially the same, so we get symmetric subgroup membership problems.

Let n be a composite number with two prime factors such that $2n+1$ is also prime. Then the multiplicative group $\mathbb{F}^*_{2n+1}$ has order $2n$, and we have a unique subgroup of order n. This group structure was suggested in [31]. A related topic was discussed in [26]. We denote the derived symmetric subgroup membership problem by $\mathcal{SOU}_n$.

Let n be a composite number with two prime factors, let p' be a prime such that $|p'+1-n| < 2\sqrt{p'}$, and let E be an elliptic curve defined over $\mathbb{F}_{p'}$ such that $\#E(\mathbb{F}_{p'}) = n$. We derive a symmetric subgroup membership problem $\mathcal{SOU}_{E/\mathbb{F}_{p'}}$. (One would probably resort to complex multiplication techniques to find E.)

To sample elements almost uniformly from the subgroups K and H, generators for K and H must be made available.

It seems plausible that $\mathcal{SOU}_n$ and $\mathcal{SOU}_{E/\mathbb{F}_{p'}}$ both are hard. Deciding if an element is in one of the subgroups is equivalent to deciding if its order is small or large. It is clear that anyone who can factor the modulus n can also break the system. Likewise, anyone who can compute discrete logarithms in the subgroups K and H can factor the modulus and hence break the system. Anyone who can compute discrete logarithms in G, but not in K and H, can decide the subgroup membership problems. Also, anyone who can decide if an element is a generator for the whole group, can decide the subgroup membership problems.

It is possible for n to have more than two prime factors, or for the two prime factors to be of different size. While factoring algorithms like the number field sieve does not seem able to take advantage of such a structure, the elliptic curve method [25] does, and this gives a fairly large lower bound on the size of the smallest prime factor.

4.5.7 Groups of unknown order

Let $p = 2ac+1$ and $q = 2bd+1$ be τ-bit primes such that a, b, c and d are pairwise relatively prime, and let $n = pq$. Let $G = Q_n$, let K be the unique subgroup of order $2ab$, and let H be the unique subgroup of order cd. Then we get a splitting problem $\mathcal{SP}_{(G,K,H)}$ and a symmetric subgroup membership problem $\mathcal{SSM}_{(G,K,H)}$ which we denote by $\mathcal{GOU}_n$.

In order to sample elements almost uniformly from K and H, generators for both groups must be made public and Lemma 2.4 applied.

In general, the factors a, b, c and d should be large. Almost certainly, $|K|$ is significantly smaller than n, so if the product $ab = |K|$ is known, it will be feasible to find a and b via factorisation, and then factor n. Therefore, it should be difficult to compute $|K|$, even when a generator for K is known. The same holds for H.

This means that a, b, c and d should each contain at least one big prime factor. This prevents anyone from finding the order of K and H by computing discrete

logarithms. Also, it blocks Pollard ρ-type attacks on the subgroup, which would have been feasible if $|K|$ or $|H|$ were small.

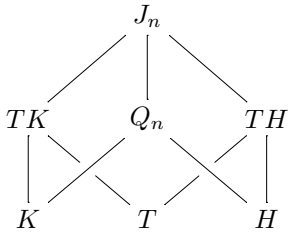
Note that unlike in the previous section, the order of G is not known, so the prime factors in a, b, c and d can still be quite small.

As in Section 4.5.2, we can let $c = d$. Then K is the subgroup of order ab and H is some subgroup of order c. Since $c|(n-1)$, c becomes known, but it does not seem to matter, even if an element of order c is made public. c should not be too large, however, as the analysis in Section 4.5.2 also applies to this case. And since c is known, the subgroup membership problem $\mathcal{SM}_{(G,H)}$ is trivial.

We denote this subgroup membership problem by $\mathcal{GOU}'_n$.

4.6 Further reductions

Let $p = 2ac + 1$, $q = 2bd + 1$, $p \equiv q \equiv 3 \pmod 4$, and $n = pq$, just as in Sections 4.5.1 and 4.5.7. Let J_n be the subgroup $\mathbb{Z}_n^*$ of elements with Jacobi symbol 1, Q_n be the subgroup of quadratic residues, K be the subgroup of order ab and H the subgroup of order cd. We denote the subgroup of order 2 generated by -1 by T. We get the following subgroup lattice:



We have already discussed the subgroup membership problems $\mathcal{SM}_{(J_n,Q_n)}$, $\mathcal{SM}_{(Q_n,K)}$ and $\mathcal{SM}_{(Q_n,H)}$. We first discuss some combinations of these problems.

Proposition 4.13. *Suppose there is a distinguisher for $\mathcal{SM}_{(J_n,K)}$ with advantage ϵ. Then either there is a distinguisher for $\mathcal{SM}_{(J_n,Q_n)}$ with advantage at least $\epsilon/3$, or there is a distinguisher for $\mathcal{SM}_{(Q_n,K)}$ with advantage at least $\epsilon/3$.*

Proof. Apply Proposition 4.2. □

Proposition 4.14. *For any distinguisher for $\mathcal{SM}_{(J_n,TK)}$ with advantage ϵ, there is a corresponding distinguisher for $\mathcal{SM}_{(Q_n,K)}$ with the same advantage, and vice versa.*

Proof. We shall show how to apply Proposition 4.3 to both directions.

First, we note that the map from J_n to Q_n given by $x \mapsto x^2$ is an automorphism on the groups Q_n, K and H. If X is uniformly distributed on $J_n \setminus TK$, then the projection of X on H is uniformly distributed on $H \setminus \{1\}$. Since squaring

is an automorphism, the projection of X^2 on H is also uniformly distributed on $H \setminus \{1\}$.

If X is uniformly distributed on TK, then X^2 is clearly uniformly distributed on K. So by Proposition 4.3 we have shown that $\mathcal{SM}_{(J_n,TK)}$ is not more difficult than $\mathcal{SM}_{(Q_n,K)}$.

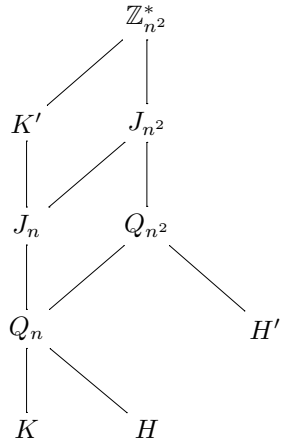
Second, let Y be uniformly distributed on $\{0,1\}$. If X is uniformly distributed on $Q_n \setminus K$, then $X(-1)^Y$ is uniformly distributed on $J_n \setminus TK$. If X is uniformly distributed on K, then $X(-1)^Y$ is uniformly distributed on TK. So again by Proposition 4.3 we have shown that $\mathcal{SM}_{(Q_n,K)}$ is not more difficult than $\mathcal{SM}_{(J_n,TK)}$. □

We shall now combine the above group structure with the group structure of Section 4.5.4. Let J_{n^2} be the elements of $\mathbb{Z}^*_{n^2}$ with Jacobi symbol 1, and let Q_{n^2} be the quadratic residues of $\mathbb{Z}^*_{n^2}$.

Recall that we have a homomorphism $\mathbb{Z}^*_{n^2} \to \mathbb{Z}^*_n$ given by taking the residue class of $\mathbb{Z}_{n^2}$ represented by the integer w to the residue class of $\mathbb{Z}_n$ represented by w. Let H' be the kernel of this homomorphism.

We also have a homomorphism $\mathbb{Z}^*_n \to \mathbb{Z}^*_{n^2}$ given by taking the residue class of $\mathbb{Z}^*_n$ represented by the integer w to the residue class of $\mathbb{Z}^*_{n^2}$ represented by w^n. Let K' be the image of $\mathbb{Z}^*_n$ under this homomorphism. We identify the subgroups of $\mathbb{Z}^*_n$ with the corresponding subgroups of K'.

We get the following subgroup lattice:



Proposition 4.15. *For any distinguisher for $\mathcal{SM}_{(Q_{n^2},Q_n)}$ with advantage ϵ, there is a corresponding distinguisher $\mathcal{SM}_{(\mathbb{Z}^*_{n^2},K')}$ with the same advantage.*

Suppose an element $j \in \mathbb{Z}^*_{n^2} \setminus J_{n^2}$ is known. Then for any distinguisher for $\mathcal{SM}_{(\mathbb{Z}^*_{n^2},K')}$ with advantage ϵ, there is a corresponding distinguisher $\mathcal{SM}_{(Q_{n^2},Q_n)}$ with the same advantage.

Proof. Obviously, the map $x \mapsto x^2$ takes the uniform distributions on $\mathbb{Z}^*_{n^2} \setminus K'$ and K' to the uniform distributions on $Q_{n^2} \setminus Q_n$ and Q_n, respectively. This means

that $\mathcal{SM}_{(\mathbb{Z}^*_{n^2}, K')}$ cannot be more difficult than $\mathcal{SM}_{(Q_{n^2}, Q_n)}$, by Proposition 4.3, and the first claim follows.

Let Y and Z be uniformly distributed on $\{0, 1\}$. If X is uniformly distributed on Q_n, then $X(-1)^Y j^{nZ}$ is uniformly distributed on K'. If X is uniformly distributed on $Q_{n^2} \setminus Q_n$, then $X(-1)^Y j^{nZ}$ is uniformly distributed on $\mathbb{Z}^*_{n^2} \setminus K'$. As above, we find that $\mathcal{SM}_{(Q_{n^2}, Q_n)}$ cannot be more difficult than $\mathcal{SM}_{(\mathbb{Z}^*_{n^2}, K')}$, by Proposition 4.3, and the second claim follows. □

Proposition 4.16. *Suppose there is a distinguisher for $\mathcal{SM}_{(Q_{n^2}, K)}$ with advantage ϵ. Then either there is a distinguisher for $\mathcal{SM}_{(Q_{n^2}, Q_n)}$ with advantage $\epsilon/3$, or there is a distinguisher for $\mathcal{SM}_{(Q_n, K)}$ with advantage $\epsilon/3$.*

Proof. Apply Proposition 4.2. □

Chapter 5

Homomorphic cryptosystems

In this section, we shall study three different homomorphic cryptosystems. We shall prove all three systems semantically secure against chosen plaintext attacks based on subgroup membership problems. All three cryptosystems are homomorphic with respect to a group operation. For several of the cryptosystems, the group operation can be regarded as addition modulo an integer, which may be very convenient.

We shall also discuss the security of some of the cryptosystems against a non-adaptive chosen ciphertext attack, under a non-standard assumption.

5.1 The general cryptosystem

Let $\mathcal{TSP}_\Gamma$ be a trapdoor splitting problem, and suppose that to every tuple (G, K, H, σ), there is an algorithm for sampling K (almost) uniformly at random, an associated group G', and a probability space F on a set of easy to compute maps $\{f : G' \to G\}$. We require that for every map f in F, the composition $\pi_H \circ f : G' \to H$ is a group isomorphism, and it is easy to derive a group isomorphism $f' : H \to G'$ such that $f' = (\pi_H \circ f)^{-1}$.

The basic idea is that H carries the message, and K contains "noise" that hides the message. f is used to insert the message in H. The splitting algorithm can be used to remove the noise, and f' can be used to recover the message. The exact construction of Π_1 from $\mathcal{TSP}_\Gamma$ is described in Figure 5.1.

Proposition 5.1. *The cryptosystem Π_1 is an homomorphic public key system.*

Proof. It is quite obvious that $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ are efficient, and that $\mathcal{D}$ is deterministic. It is also clear that if $rf(m)$ is the ciphertext output by $\mathcal{E}$, then the projection on H is $\pi_H(f(m))$, so $\mathcal{D}$ will output m. Therefore, Π_1 is a public key system.

Key generation.
Input: $\mathcal{TSP}_\Gamma$, τ.

1. $(G, K, H, \sigma, F) \leftarrow \Gamma_\tau$.
2. $f \leftarrow F$.
3. $pk \leftarrow (G, K, H, G', f)$.
4. $sk \leftarrow (G, \sigma, G', f')$.
5. Output (pk, sk).

Output: A public-private key pair (pk, sk).

Encryption.
Input: $pk = (G, K, H, G', f)$, $m \in G'$.

1. $x \leftarrow f(m)$, $r \stackrel{\approx}{\leftarrow} K$.
2. $c \leftarrow rx$.
3. Output c.

Output: A ciphertext $c \in G$.

Decryption.
Input: $sk = (G, \sigma, G', f')$, $c \in G$.

1. $(c_1, c_2) \leftarrow \sigma(c)$.
2. $m \leftarrow f'(c_2)$.
3. Output m.

Output: A message $m \in G'$.

Figure 5.1: The cryptosystem Π_1.

Finally, if $rf(m)$ and $r'f(m')$ are two ciphertexts, then the decryption of $rf(m)r'f(m')$ is clearly $f'(\pi_H(rr'f(m)f(m'))) = f'(\pi_H(f(m)))f'(\pi_H(f(m'))) = mm'$, and the cryptosystem is homomorphic. □

Proposition 5.2. *Suppose that for any map $f \in F$, we have that for all $m \in G'$, $f(m) = \pi_H(f(m))$. Then the public key cryptosystem Π_1 is one-way if and only if the splitting problem $\mathcal{SP}_\Gamma$ is hard.*

Proof. We prove the statements by giving reductions.

It is clear that any algorithm that solves the splitting problem can be used instead of the trapdoor function σ used in $\mathcal{D}$.

Now suppose that (G, K, H, x) is an instance of a splitting problem. The group G' is given, and we can sample f from F, to construct a public key (G, K, H, G', f), which has been sampled just as $\mathcal{K}$ would have done it.

Note that when messages are sampled uniformly from G', $\mathcal{E}$ samples ciphertexts uniformly from G. Therefore, x is a ciphertext for pk sampled exactly as normal ciphertexts, and if m is the decryption of x, then $f(m)$ is the projection on H of x, and $(x(f(m))^{-1}, f(m))$ is the splitting of x. □

Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem such that H is cyclic. Suppose there is some efficient, deterministic algorithm to compute discrete logarithms in H. There is a natural map $\mathbb{Z} \to G$ given by $m \mapsto g^m$, where $g \in G$ is such that its projection on H is a generator for H. If we decide on some fixed map $\kappa : \mathbb{Z}_{|H|} \to \mathbb{Z}$ to choose representatives for the residue classes of $\mathbb{Z}_{|H|}$, we get a map $f : \mathbb{Z}_{|H|} \to G$ given by $[m] \mapsto g^{\kappa(m)}$.

It is quite clear that if we have a trapdoor subgroup discrete logarithm problem $\mathcal{TSDL}_\Gamma$, then we can use this map to get a variant cryptosystem $\Pi_{1'}$. It is described in Figure 5.2.

For $\Pi_{1'}$, Proposition 5.2 is not true unless the element g is a generator H. We can recover the following, slightly weaker result.

Proposition 5.3. *The public key cryptosystem $\Pi_{1'}$ is one-way if and only if the subgroup discrete logarithm problem $\mathcal{SDL}_\Gamma$ is hard.*

Proof. We prove the statements by giving reductions.

It is clear that any algorithm that solves the subgroup discrete logarithm problem can be used instead of the trapdoor function σ and the logarithm computation used in $\mathcal{D}$.

Let (G, K, H, g, y) be a problem instance sampled from a subgroup discrete logarithm problem $\mathcal{SDL}_\Gamma$. We get the public key $pk = (G, K, H, g)$, and this has been sampled exactly according to $\mathcal{K}(\tau)$. y is a ciphertext under the public key pk, and it has been sampled exactly as ciphertexts are sampled. Its decryption m is congruent to $\log_{(H,g)} y$ modulo $|H|$, so anyone who can decrypt the ciphertext can solve the subgroup discrete logarithm problem. □

Key generation.
Input: $\mathcal{TSDL}_\Gamma$, τ.

1. $(G, K, H, \sigma, g) \leftarrow \Gamma_\tau$.
2. $pk \leftarrow (G, K, H, g)$.
3. $(g_1, g_2) \leftarrow \sigma(g)$.
4. $sk \leftarrow (G, \sigma, g_2)$.
5. Output (pk, sk).

Output: A public-private key pair (pk, sk).

Encryption.
Input: $pk = (G, K, H, g)$, $[m] \in \mathbb{Z}_{|H|}$.

1. $x \leftarrow g^m$, $r \overset{\approx}{\leftarrow} K$.
2. $c \leftarrow rx$.
3. Output c.

Output: A ciphertext $c \in G$.

Decryption.
Input: $sk = (G, \sigma, G', f')$, $c \in G$.

1. $(c_1, c_2) \leftarrow \sigma(c)$.
2. $m \leftarrow \log_{g_2} c_2$.
3. Output m.

Output: A message $m \in \mathbb{Z}_{|H|}$.

Figure 5.2: The cryptosystem $\Pi_{1'}$.

Now we use the indistinguishability property of the subgroup membership problem to prove a stronger security result for the cryptosystems Π_1 and $\Pi_{1'}$.

Theorem 5.4. *The public key cryptosystems Π_1 and $\Pi_{1'}$ are semantically secure if and only if the subgroup membership problem $\mathcal{SM}_\Gamma$ is hard.*

Proof. Again, we give reductions.

Let (G, K, H, G', f) be a public key. Suppose we have a ciphertext and want to decide if c decrypts to a message m or to some other, random message. First, we set $x = f(m)$. If $\mathcal{D}(sk, c) = m$, then cx^{-1} is uniformly distributed in K. If $\mathcal{D}(sk, c) \neq m$, then cx^{-1} is uniformly distributed in $G \setminus K$. Hence, we can decide if c decrypts to m or not with the same advantage that we can decide the subgroup membership problem $\mathcal{SM}_{(G,K)}$.

Next, suppose we have a subgroup membership problem $\mathcal{SM}_{(G,K)}$ and an instance x. As in the proof of Proposition 5.2, we first sample a public key. Let $m \in pk_M$ be a message, and sample x' uniformly from K. If $x \in K$, then $xx'f(m)$ is an encryption of m. If $x \in G \setminus K$, then $xx'f(m)$ is not an encryption of m, but of some other, random message. In both cases, the distribution of the ciphertexts are exactly as $\mathcal{E}$ would produce, so we can decide the subgroup membership problem $\mathcal{SM}_{(G,K)}$ with the same advantage that we can distinguish encryptions of m from other encryptions of other messages. □

We briefly describe various concrete instances of Π_1 and $\Pi_{1'}$.

Goldwasser-Micali We can instantiate Π_1 with the Quadratic Residue problem from Section 4.5.1 (using a suitable family of moduli), and we get the Goldwasser-Micali cryptosystem [22].

In this case, the public key is $pk = (n)$. Sampling an element uniformly at random is done by first sampling an element uniformly at random from $\mathbb{Z}_n^*$, then squaring it. The message space is $\{1, -1\}$, which is a subgroup of J_n, and $f = f'$ is the identity.

The secret key is one of the prime factors of n, say $sk = (p)$. To decrypt a ciphertext c, one simply computes the Legendre symbol of c with respect to p.

If we let the message space be $\{0, 1\}$ with the mapping $0 \mapsto 1$ and $1 \mapsto -1$, then the cryptosystem becomes additive and we get an instance of $\Pi_{1'}$.

ElGamal We can instantiate Π_1 with the Computational Diffie-Hellman problem from Section 4.5.3 (using a suitable family of groups), we get the classical ElGamal cryptosystem [18].

In this case, the public key is $pk = (G', g, y)$ where G' is the underlying group structure, g is a generator, and (g, y) defines the subgroup K. The subgroup H is then trivially isomorphic to G', which is the message space. Sampling uniformly in K is simply a matter of sampling k uniformly from $\{0, \ldots, |G'| - 1\}$ and computing $(g, y)^k = (g^k, y^k)$.

The secret key is $sk = (G', a)$, where a is an integer such that $y = g^a$. Then the splitting of (z, z') can be computed as $((z, z^a), (1, z'z^{-a}))$.

Paillier The Computational Composite Residuosity problem in Section 4.5.4, $\mathcal{CCR}_n$, can be considered either as a splitting problem or as a subgroup discrete logarithm problem. We take it to be the latter, so the Paillier cryptosystem [35] is an instance of $\Pi_{1'}$.

The public key will be $pk = (n)$, and the secret key $sk = (d)$, where d is the order of K, and the splitting is computed using Proposition 4.5.

Paillier [35] also suggested a subgroup variant. This is essentially the same system, but the modulus is chosen as in Section 4.5.7 and K is taken to be a suitable subgroup of the nth powers (which are isomorphic to $\mathbb{Z}_n^*$). Propositions 4.15 and 4.16 say that the subgroup variant is secure if both $\mathcal{DCR}_n$ and $\mathcal{GOU}_n$ are hard.

Okamoto-Uchiyama The construction in Section 4.5.5 is a subgroup discrete logarithm problem, and we get an instance of $\Pi_{1'}$ [34].

Let $p = 2a + 1$, $q = 2b + 1$ be primes and $n = p^2q$. The public key consists of (n, g), where $g \in \mathbb{Z}_n^*$ has order divisible by p.

An interesting point is that given only the modulus n, we can sample a public key with almost the same distribution as the key generation algorithm. To see this, note that a random element in $\mathbb{Z}_n^*$ has order divisible by p with probability approximately $1 - 1/p$.

Now suppose that we know that p is a t-bit integer, that is, $2^{t-1} \leq p < 2^t$. If we sample a from $\{2^t, \ldots, 2^{t+1} - 1\}$, then $p < a < 4p$. We sample r uniformly at random from $\mathbb{Z}_n^*$ and create the ciphertext $r^n g^a$. This is a valid ciphertext with decryption m. So we know that $a \equiv m \pmod{p}$, or that $a - m = kp$. Since $0 \leq m < p$, k must be 1, 2 or 3. In other words, if we can decrypt ciphertexts for this instance of the cryptosystem, we can factor the underlying modulus.

Naccache-Stern If we instantiate $\Pi_{1'}$ with $\mathcal{HR}_n$, we get the Naccache-Stern cryptosystem [29]. By instantiating $\Pi_{1'}$ with $\mathcal{HR}'_n$, we get a similar variant, which fits into both Π_1 and $\Pi_{1'}$, but the latter is most natural.

Let $p = 2ac + 1$, $q = 2bd + 1$ be primes, such that a, b are primes and c, d are smooth integers (either equal, or relatively prime). Let $n = pq$. The public key is (n, g), where $g \in G$ defines the function f by $f(m) = g^m$. If $\gcd(c, d) = 1$, then g should be an element of order divisible by $abcd$. If $c = d$, then g should be an element of order c that is not congruent to 1 modulo either p or q.

Combinations It is possible to combine $\mathcal{HR}_n$ (or $\mathcal{HR}'_n$) with $\mathcal{CCR}_n$ to get an instance of $\Pi_{1'}$. Proposition 4.15 and Proposition 4.16 say that the resulting instance of $\Pi_{1'}$ is secure if both $\mathcal{HR}_n$ (or $\mathcal{HR}'_n$) and $\mathcal{CCR}_n$ are hard. Such a cryptosystem would have somewhat higher bandwidth. The disadvantages would include a larger public key (one would need to store generators for K and G) and slower decryption.

5.2 Non-standard assumptions

To motivate the work in this section, we briefly describe a non-adaptive chosen ciphertext attack against the Paillier cryptosystem which may be faster than factoring the modulus. This is basically a variant of the attacks in [15], but using a trick of looking for smooth numbers in $\mathbb{Z}_n$, not in $\mathbb{Z}_{n^2}$.

So let B be a set of small primes (possibly also including -1). Note that every number in B gives us a valid ciphertext for the Paillier cryptosystem. We use the decryption oracle to obtain a decryption of every number in B, and store them in a table.

When we get the challenge ciphertext c, we choose random nth powers c', which decrypt to 0, until $cc' \bmod n$ is B-smooth. If the factorisation of cc' is $\prod_{\ell_i \in B} \ell_i^{\alpha_i}$, we can create a ciphertext $c'' = cc' \prod_{\ell_i \in B} \ell_i^{-\alpha_i}$, and we know that

$$c'' = cc' \prod_{\ell_i \in B} \ell_i^{-\alpha_i} \equiv 1 \pmod{n}.$$

Because of the homomorphic property and the decryptions of the elements in B, we know how the decryption has been changed. And because $c'' = [(1+n)^{m'}]_{n^2}$ for some m', we know its decryption, and therefore the decryption of the original ciphertext c.

While this attack is of limited practical interest, it does provide some motivation to look at non-adaptive chosen ciphertext security for homomorphic cryptosystems.

5.2.1 Knowledge-of-exponent

Let G be a group. For some groups, it seems difficult to find elements of the group, except by using the group operation to compute combinations of previously known elements. We formalise this in the following definition.

Definition. We say that a family of groups $\Gamma = \{\Gamma_\tau\}$ satisfies the *knowledge-of-exponents* assumption, if for any efficient algorithm A that takes as input a group G sampled from Γ_τ and a tuple of group elements $(g_1, \ldots, g_m)$, and outputs an element x of G with significant (in τ) probability, there is a trace-variant A' of A that also outputs a tuple of integers $(k_1, \ldots, k_m)$ such that $x = g_1^{k_1} \ldots g_m^{k_m}$.

This concept was first proposed by Damgård [12]. Other work on the subject is [4, 23]. Given the subgroup membership problem $\mathcal{DDH}_{G'} = \mathcal{SM}_{(G,K)}$, they hypothesise that the subgroup K has this property. That is, given a subgroup of $G' \times G'$ generated by (g_1, g_2), the only way to find an element in this subgroup is to compute powers (g_1^k, g_2^k), for various k.

We note that in the generic model described in Section 3.6.2, if the representation set S is much larger than the group size, then the knowledge-of-exponents assumption holds.

We shall propose three new groups for which it is plausible that this property holds. The two first structures come from the subgroups of the symmetric subgroup membership problems $\mathcal{SOU}_n$ and $\mathcal{GOU}_n$. As for the Diffie-Hellman-based example, it simply seems difficult to generate elements in any other way than by computing powers of a given generator.

The third group structure that we shall propose is qualitatively different. Let $n = pq$ be an RSA modulus, and let E be an elliptic curve defined over $\mathbb{Z}_n$, given by the equation

$$y^2 z \equiv x^3 + axz^2 + bz^3 \pmod{n}.$$

For an elliptic curve defined over a finite field $\mathbb{F}_p$, it is easy to find points on the curve. Simply choose an x-coordinate uniformly at random, and with probability approximately $1/2$, $x^3 + ax + b$ will be a quadratic residue modulo p. Computing a square root y of $x^3 + ax + b$ modulo p is easy, and $(x : y : 1)$ is an $\mathbb{F}_p$-rational point on E.

When we move to the ring $\mathbb{Z}_n$, the situation changes. Of course, we have the trivial point $(0 : 1 : 0)$, but it seems difficult to find any other point. Choosing an x-coordinate does not lead to a point, since computing square roots modulo n is equivalent to factoring, and factoring seems hard. Starting with a y-coordinate, we get a cubic equation for the x-coordinate, but solving cubic equations seems to be hard.

If we are given a set of points, we can obviously use the group operation to find new points on the curve. We can also try to intersect the curve with other curves. One possibility is the quadratic curve

$$C : yz \equiv ax^2 + bxz + cz^2 \pmod{n}.$$

If we are given three distinct, non-trivial points $P_1, P_2, P_3 \in E(\mathbb{Z}_n)$ that are not collinear, we can find such a quadratic curve through those points. Inserting the quadratic curve equation into the elliptic curve equation, we get a polynomial of degree 4 where we know three of the solutions, hence we can find the fourth and from that a point P_4 in $E(\mathbb{Z}_n)$.

To analyse this, we consider the situation modulo the prime factor p. By Bezout's theorem, C and E intersect in exactly 6 points when counting multiplicities. The four points $P_1, \ldots, P_4$ are given, and it is clear that a fifth intersection point is $O = (0 : 1 : 0)$. Since the line $z = 0$ is tangent to both E and C in O, their intersection number at O is greater than 1. This means that there are five points of intersection, and the divisor of C is $\operatorname{div}(C) = (P_1) + \cdots + (P_4) + 2(O)$.

If C' is the plane curve given by $z^2 = 0$, then $\operatorname{div}(C') = 6(O)$. Let f in the function field of E be defined by $f(x, y, z) = (yz - ax^2 - bxz - cz^2)/z^2$. It is then clear that

$$\operatorname{div}(f) = \operatorname{div}(C) - \operatorname{div}(C') = 0,$$

which gives us

$$(P_1) - (O) + (P_2) - (O) + (P_3) - (O) + (P_4) - (O) = 0,$$

or $P_1 + P_2 + P_3 + P_4 = O$. The four points sum to zero.

It seems as if most approaches to finding points in $E(\mathbb{Z}_n)$ either lead to polynomial equations modulo n, which seem to be hard to solve, or to trivial relations among points. It therefore seems plausible that the an elliptic curve modulo n satisfies the knowledge-of-exponent assumption.

The qualitative difference between this group and the other group structures is that it is trivial to determine if a point $(x : y : z)$ is in $E(\mathbb{Z}_n)$ or not.

5.2.2 A security proof

We shall consider the cryptosystem Π_1 derived from a splitting problem $\mathcal{SP}_\Gamma$, but let us suppose additionally that for every triple $(G, K, H) \in \Gamma_\tau$, G is cyclic, and there is some group $\tilde{K}$ along with easily computable homomorphisms $\phi : G \to \tilde{K}$ and $\psi : \tilde{K} \to K$, such that $\phi \circ \psi$ is the map $x \mapsto x^{|H|}$, and ϕ restricted to K is an isomorphism.

We shall assume that a description of $\tilde{K}$ is easy to recover from a description of (G, K, H), and vice versa. We denote by $\tilde{\Gamma}$ the family of groups derived from Γ.

We shall also assume that a generator g for K is available for sampling elements. The public key for Π_1 is therefore (G, g, f).

Theorem 5.5. *Let Π_1 be derived from a splitting problem $\mathcal{SP}_\Gamma$ as described above. If the knowledge-of-exponent assumption holds for $\tilde{\Gamma}$, then Π_1 is weakly plaintext aware.*

Proof. First of all, we note that if $pk = (G, g, f)$ is a valid public key for Π_1, then $pk' = (G, g^{|H|}, f)$ is also a valid public key, and the two are interchangeable.

So assume that B is a weak forger for Π_1, that is, B takes as input a public key pk, and outputs a ciphertext $c \in G$. We use B to construct an algorithm A that takes as input a group $\tilde{K}$ along with a generator $\tilde{g}$, and outputs an element of $\tilde{K}$.

Given $\tilde{K}$, A recovers (G, K, H). It sets $g = \psi(\tilde{g})$, samples f and runs B on input (G, g, f). B outputs $c \in G$, and A outputs $\phi(c)$.

Now it is clear that A is an algorithm that takes as input the group $\tilde{K}$ and a generator for $\tilde{K}$, and outputs an element $\tilde{c}$ of $\tilde{K}$, so by the knowledge-of-exponent assumption, there exists some trace-variant A' that on the same input and random tape outputs the same element $\tilde{c}$ and an integer k such that $\tilde{c} = \tilde{g}^k$.

Now it is quite clear that if B is a forger, we can create a trace-variant B' of B that outputs not only a ciphertext, but also the decryption of the ciphertext.

B' takes as input a public key (G, g, f). It first runs B on input $(G, g^{|H|}, f)$, and gets a ciphertext c as output. Then it rewinds the random tape and runs A' on input of $(\tilde{K}, \phi(g))$. It is then quite clear that A', except with negligible probability, outputs $\tilde{c}$ and k such that $\tilde{c} = \phi(c)$.

But then we have that

$$\phi(cg^{-k|H|}) = \phi(c)\tilde{g}^{-k} = \tilde{c}(\tilde{c}^{-1}) = 1,$$

which means that $cg^{-k|H|}$ must be in H, and since $g^{-k|H|}$ is in K, we can recover the decryption using f^{-1}. This also shows us how the weak knowledge extractor works. □

It is now quite clear that by Theorem 3.4, we can conclude that Π_1 is secure against non-adaptive chosen ciphertext attacks, assuming that the derived subgroup membership problem is hard and that the knowledge-of-exponent assumption for $\tilde{\Gamma}$ holds. As a corollary, we get that Π_1 is secure against non-adaptive chosen ciphertext attacks in the generic model.

The two most interesting instantiations of this problem is for the subgroup variant of the Paillier cryptosystem, and the elliptic curve variant of the Paillier cryptosystem described by Galbraith [20].

5.3 A second homomorphic cryptosystem

Let Γ_τ be a set of tuples (G, K, H, g) such that $\mathcal{SP}_{(G,K,H)}$ is a splitting problem, g is a generator for K, and $|H|$ has no small prime divisors. Let $\Gamma = \{\Gamma_\tau\}$. We shall be interested in the derived subgroup membership problem $\mathcal{SM}_\Gamma$, or the symmetric subgroup membership problem $\mathcal{SSM}_\Gamma$.

The basic idea is that we shall do as for the ElGamal cryptosystem, but that the Diffie-Hellman key exchange part should be done in K and the message should be either in G, or restricted to H. The exact construction of Π_2 from Γ is described in Figure 5.3.

Proposition 5.6. *The cryptosystem Π_2 is an homomorphic public key system.*

Proof. Clear. □

Theorem 5.7. *The public key cryptosystem Π_2 is semantically secure for messages in H if the subgroup membership problem $\mathcal{SM}_\Gamma$ is hard, and semantically secure for messages in G if the symmetric subgroup membership problem $\mathcal{SSM}_\Gamma$ is hard.*

Proof. We assume that $A = (A_1, A_2)$ is a chosen plaintext adversary against indistinguishability for Π_2 with advantage ϵ.

Experiment 1.
Input: (G, K, H, g), $x \in G$.

1. $k \leftarrow \{0, \ldots, |G| - 1\}$, $y \leftarrow g^k$.
2. $(m_0, m_1, o) \leftarrow A_1(G, K, H, g, k)$.
3. $b \leftarrow \{0, 1\}$.
4. $e \leftarrow x^k m$.
5. $b' \leftarrow A_2(m_0, m_1, x, e, o)$.
6. If $b = b'$, output 1, otherwise output 0.

Output: 0 or 1.

Key generation.
Input: Γ, τ.

1. $(G, K, H, g) \leftarrow \Gamma_\tau$.
2. $k \overset{\approx}{\leftarrow} \{0, \ldots, |G| - 1\}$.
3. $y \leftarrow g^k$.
4. $pk \leftarrow (G, K, H, g, y)$.
5. $sk \leftarrow (G, k)$.
6. Output (pk, sk).

Output: A public-private key pair (pk, sk).

Encryption.
Input: $pk = (G, K, H, g, y)$, $m \in G$.

1. $w \overset{\approx}{\leftarrow} \{0, \ldots, |K| - 1\}$.
2. $x \leftarrow g^w$, $z \leftarrow y^w$, $e \leftarrow zm$.
3. Output (x, e).

Output: A ciphertext $c \in G \times G$.

Decryption.
Input: $sk = (G, k)$, $(x, e) \in G \times G$.

1. $z \leftarrow x^k$.
2. $m \leftarrow ez^{-1}$.
3. Output m.

Output: A message $m \in G$.

Figure 5.3: The cryptosystem Π_2 derived from a (symmetric) subgroup membership problem. The message is restricted to H if $\mathcal{SM}_{(G,H)}$ is not hard.

In a realisation of Experiment 1, we could perhaps not sample k uniformly at random, but δ-close to uniform (Lemma 2.3), where δ can be made arbitrarily small at little cost.

Note that Step 1 does exactly as $\mathcal{K}$ would do, and Step 4 does exactly as $\mathcal{E}$ would do, if it had first sampled x. (They would of course not know k, but would have sampled x as g^w and computed y^w.)

Let T' be the event that Experiment 1 outputs 1 when its input x is in K. We note that if $x \in K$, then Experiment 1 proceeds exactly as a real attack, so

$$\mathrm{Adv}_A^{\mathrm{IND}} \leq |\Pr[T'] - 1/2|. \tag{5.1}$$

Let T be the event that Experiment 1 outputs 1 when its input x is in $G \setminus K$. We note that we can obviously derive an algorithm A' from Experiment 1 that distinguishes K from $G \setminus K$ such that

$$|\Pr[T'] - \Pr[T]| \leq \mathrm{Adv}_{A'}^{\mathcal{SM}_{(G,K)}} + \delta. \tag{5.2}$$

Now we modify Experiment 1 as follows to get Experiment 1′: We replace Step 4 with the following step:

4. $z \leftarrow H$, $c \leftarrow x^k m z$.

Let T_1 be the event that Experiment 1′ outputs 1 when its input x is in $G \setminus K$.

By Lemma 4.12, the distributions of (x, y, x^k) and $(x, y, x^k z)$ are $(|H| - \phi(|H|))/|H|$-close, and otherwise the two algorithms are identical, so

$$|\Pr[T] - \Pr[T_1]| \leq \frac{|H| - \phi(|H|)}{|H|}. \tag{5.3}$$

It is clear that if the message space is H, then the input to A_2 is independent of the bit b chosen, so no matter what A_2 does, it guesses correctly with probability $1/2$, and

$$\Pr[T_1] = 1/2. \tag{5.4}$$

By combining (5.1)–(5.4), we immediately get that

$$\mathrm{Adv}_A^{\mathrm{IND}} \leq \mathrm{Adv}_{A'}^{\mathcal{SM}_{(G,K)}} + \frac{|H| - \phi(|H|)}{|H|} + \delta.$$

Under the assumption that $\mathcal{SM}_{(G,K)}$ is hard, this proves the first claim.

To prove the second claim, we first make a modification to Experiment 1′ to get Experiment 1″: We replace Step 4 with the following step:

4. $z \leftarrow G \setminus H$, $c \leftarrow x^k m z$.

Let T_2 be the event that Experiment 1″ outputs 1 when its input x is in $G \setminus K$. It is clear that if z was sampled uniformly at random, then the input to A_2 would be independent of the bit b chosen, so no matter what A_2 does, it would guess

correctly with probability 1/2. By Lemma 2.2, the uniform distribution on $G \setminus H$ is $2|H|/|G|$-close to the uniform distribution on G, so we have that

$$|\Pr[T_2] - 1/2| \leq \frac{2|H|}{|G|}. \tag{5.5}$$

To bound the difference in probability between T_1 and T_2, we introduce the following experiment:

Experiment 2.
Input: (G, K, H, g), $z \in G$.

1. $k \leftarrow \{0, \ldots, |G| - 1\}$, $y \leftarrow g^k$.
2. $(m_0, m_1, o) \leftarrow A_1(G, K, H, g, k)$.
3. $b \leftarrow \{0, 1\}$.
4. $x \leftarrow G \setminus K$, $e \leftarrow x^k m z$.
5. $b' \leftarrow A_2(m_0, m_1, x, e, o)$.
6. If $b = b'$, output 1, otherwise output 0.

Output: 0 or 1.

Let R' be the event that Experiment 2 outputs 1 when its input z is in H, and let R be the corresponding event with the input z is in $G \setminus H$.

It is quite clear that if the input z to Experiment 2 is in H, then Experiment 2 behaves exactly as Experiment 1′ behaves when its input x is in $G \setminus K$. Likewise, if $z \in G \setminus H$, then Experiment 2 behaves exactly as Experiment 1″ behaves when its input x is in $G \setminus K$. So we can derive an algorithm A'' from Experiment 2 such that

$$|\Pr[T_1] - \Pr[T_2] = |\Pr[R'] - \Pr[R]| \leq \mathrm{Adv}_{A''}^{\mathcal{SM}_{(G,H)}} + \delta. \tag{5.6}$$

Combining (5.1)–(5.3), (5.5), and (5.6), we get that

$$\mathrm{Adv}_A^{\mathrm{IND}} \leq \mathrm{Adv}_{A'}^{\mathcal{SM}_{(G,K)}} + \mathrm{Adv}_{A''}^{\mathcal{SM}_{(G,K)}} + \frac{|H| - \phi(|H|)}{|H|} + 2\delta,$$

which proves the second claim. □

Note that this cryptosystem does not require a trapdoor problem, unlike Π_1 and $\Pi_{1'}$. We shall see more of the advantages of this approach in Section 7.

This variant of ElGamal can usefully be instantiated with the Decision Composite Residuosity problem (Section 4.5.4), to get an additively homomorphic cryptosystem. Various technical[1] benefits of such an approach was discussed in [13].

It can also be usefully instantiated with the symmetric subgroup membership problems discussed in Section 4.5.6 and Section 4.5.7.

[1] A non-technical benefit is apparently avoiding a patent on the Paillier cryptosystem.

Chapter 6

Key encapsulation methods

A key encapsulation method is a public key cryptosystem whose only job is to encapsulate keys for symmetric cryptosystems. The idea is that the weaker goal of transporting a random key could be easier to do than the more complicated task of transporting messages with an unknown distribution.

6.1 Security against passive attacks

Let $\mathcal{SP}_{(G,K,H)}$ be a splitting problem. The basic idea is that we can sample elements uniformly from K and H to be the random key, and hide them by multiplying. We recover the random key with the splitting algorithm. The key encapsulation method is described in Figure 6.1.

It is quite clear that the key encapsulation method Λ_1 is secure against key recovery if the splitting problem is hard. As we shall see, deciding if the correct key has been recovered is possible if either K can be distinguished from $G \setminus K$ or if H can be distinguished from $G \setminus H$.

Theorem 6.1. *Let $\mathcal{TSP}_\Gamma$ be a trapdoor splitting problem, such that for every tuple (G, K, H), the group order $|G|$ has no small prime factors. The key encapsulation method Λ_1 is semantically secure if and only if the symmetric subgroup membership problem $\mathcal{SSM}_\Gamma$ is hard.*

Proof. Given a key-ciphertext pair $((x, y), c) \in G \times G \times G$, it is quite clear that if $x \in K$ or $y \in H$, then (x, y) is with overwhelming probability the correct key.

Now we show that an algorithm for distinguishing the keys output by Λ_1 from random keys must lead to an algorithm for distinguishing either K or H.

Suppose the sampling algorithm given by the symmetric subgroup membership problem samples elements δ-close to the uniform distribution. We note that by using the sampling algorithms of the symmetric subgroup membership problem, and by Lemma 2.3, even if we do not know $|G|$, we can implement modified experiments whose underlying probability spaces are $O(\delta+\delta')$-close to the original experiments. The extra work needed is $O(\log 1/\delta')$ exponentiations in G.

Key generation.
Input: $\mathcal{TSP}_\Gamma$, τ.

1. $(G, K, H, \sigma) \leftarrow \Gamma_\tau$.
2. $pk \leftarrow (G, K, H)$.
3. $sk \leftarrow (G, \sigma)$.
4. Output (pk, sk).

Output: A public-private key pair (pk, sk).

Encryption.
Input: $pk = (G, K, H)$.

1. $x \overset{\approx}{\leftarrow} K$, $y \overset{\approx}{\leftarrow} H$.
2. Output $((x, y), xy)$.

Output: A key-ciphertext pair in $G \times G \times G$.

Decryption.
Input: $sk = (G, \sigma)$, $c \in G$.

1. $(x, y) \leftarrow \sigma(c)$.
2. Output (x, y).

Output: A key in $G \times G$.

Figure 6.1: The key encapsulation method Λ_1.

Experiment 1.
Input: (G, K, H), A, $x \in G$.

1. $x' \leftarrow K$.
2. $r \leftarrow \{0, \ldots, |G| - 1\}$.
3. $y \leftarrow H$.
4. $b \leftarrow \{0, 1\}$.
5. If $b = 1$, then $(c, c', c'') \leftarrow (x^r x', y, x^r x' y)$, otherwise $(c, c') \leftarrow G \times G$ and $c'' \leftarrow cc'$.
6. $b' \leftarrow A((c, c'), c'')$.
7. If $b = b'$, then output 1, otherwise output 0.

Output: 0 or 1.

Experiment 2.
Input: (G, K, H), A, $y \in G$.

1. $x \leftarrow G$.
2. $y' \leftarrow H$.
3. $r \leftarrow \{0, \ldots, |G| - 1\}$.
4. $b \leftarrow \{0, 1\}$.
5. If $b = 1$, then $(c, c', c'') \leftarrow (x, y^r y', xy^r y')$, otherwise $(c, c') \leftarrow G \times G$ and $c'' \leftarrow cc'$.
6. $b' \leftarrow A((c, c'), c'')$.
7. If $b = b'$, then output 1, otherwise output 0.

Output: 0 or 1.

Figure 6.2: Experiments for the proof of Theorem 6.1.

Let T_1 be the event that Experiment 1 returns 1 when the input x is in K. Let T_1' be the event that Experiment 1 returns 1 when the input x is in $G \setminus K$. Let T_2 be the event that Experiment 2 returns 1 when the input y is in H. Let T_2' be the event that Experiment 2 returns 1 when the input y is in $G \setminus H$.

It is clear that if the input to Experiment 1 is in K, then the experiment proceeds exactly as an attack against the cryptosystem would do, except that in a real attack, x would not be sampled uniformly but δ-close to uniformly. In other words,

$$\epsilon \leq |\Pr[T_1] - 1/2| + \delta.$$

Next, we see that if the input to Experiment 1 is in $G \setminus K$, then when x is a generator for G, x^r is uniformly distributed in G, so $(x^r x', y)$ is uniformly distributed in $G \times H$. By Lemma 2.5, the probability that x is a generator is larger than $\phi(|G|)/|G|$.

If the input to Experiment 2 is in H, then no matter what r is, $(x, y^r y')$ is uniformly distributed in $G \times H$. This means that

$$|\Pr[T_1'] - \Pr[T_2]| \leq \frac{|G| - \phi(|G|)}{|G|}.$$

Finally, suppose that the input y to Experiment 2 is in $G \setminus H$, and that it is a generator for G. Then $(x, y^r y')$ is distributed uniformly in $G \times G$. It is then clear that

$$|\Pr[T_2'] - 1/2| \leq \frac{|G| - \phi(|G|)}{|G|}.$$

Putting it all together, we get that

$$\begin{aligned}
\epsilon &\leq |\Pr[T_1] - \Pr[T_1'] + \Pr[T_1'] - \Pr[T_2] + \\
&\quad \Pr[T_2] - \Pr[T_2'] + \Pr[T_2'] - 1/2| + \delta \\
&\leq |\Pr[T_1] - \Pr[T_1']| + \frac{|G| - \phi(|G|)}{|G|} + \\
&\quad |\Pr[T_2] - \Pr[T_2']| + \frac{|G| - \phi(|G|)}{|G|} + \delta.
\end{aligned}$$

Since (up to $O(\delta')$) $|\Pr[T_1] - \Pr[T_1']|$ and $|\Pr[T_2] - \Pr[T_2']|$ are the advantages of some algorithms distinguishing K from $G \setminus K$ and H from $G \setminus H$, respectively, the assumption that the symmetric subgroup membership problem was hard means that ϵ must be negligible. □

We can create a simple ElGamal-like cryptosystem using this key encapsulation method and the shift cipher in G. The symmetric cryptosystem has message space G, ciphertext space G and key space $G \times G$. The encryption algorithm is $\mathcal{SE}((x, y), m) = xm$, and the decryption algorithm is $\mathcal{SD}((x, y), c) = cx^{-1}$.

When this cryptosystem is instantiated with the symmetric subgroup membership problem $\mathcal{SSM}_{(G,K,H)} = \mathcal{SOU}_n$ described in Section 4.5.6, we see that encryption requires one exponentiation in K and one in H, while decryption requires essentially the same work.

Compared to ElGamal over G, which requires two exponentiations in G for encryption and one for decryption, this is quite efficient. But Π_2 in Section 5.3 requires two exponentiations in K for encryption, and only one exponentiation in K for decryption. This is rather more efficient than the above system, and just as secure.

We conclude that the key encapsulation method Λ_1, while secure, is not very useful.

6.2 Security against active attacks

The key encapsulation method proposed in the previous section seems difficult to harden against adaptive attacks. We shall introduce a new assumption, and design a key encapsulation method secure against adaptive attacks in the random oracle model.

The basic idea is that if we pass the key in Λ_1 through a hash function and use the hash output as a key, it will be very difficult to recover the splitting from the key, and the decryption of other ciphertexts would not give any information about the challenge ciphertext. The reason this is not trivial, is that the adversary can use the decryption oracle to check if a group element is in one of the subgroups or not.

Definition. A *gap splitting problem* is a splitting problem $\mathcal{SP}_{(G,K,H)}$, but any algorithm that tries to solve the splitting problem has access to a splitting oracle that on input of (x, y) and c returns 1 if (x, y) is the splitting of c, and 0 otherwise.

Gap problems were first defined by Okamoto and Pointcheval [32], and they use it to prove that their REACT conversion scheme [33] can be applied to e.g. ElGamal.

The gap problems are somewhat artificial, since the presence of an oracle is assumed. Therefore, any algorithm that solves the the gap problem is just a reduction from the splitting problem to the corresponding subgroup membership problems. If the subgroup membership problems are both easy, we get a *real gap splitting problem*.

The first real gap splitting problem appeared in [24]. The only known examples of such problems derive from certain elliptic curves with bilinear pairings. A bilinear pairing e on a group G is a map from $G \times G$ into a group G' such that for any $x \in G$, $e(x, x) \neq 1$, and for any $x, y \in G$ and $a, b \in \mathbb{Z}$, $e(x^a, y^b) = e(x, y)^{ab}$.

The most interesting variant is the one derived from the Computational Diffie-Hellman problem in G. It is clear that the pairing e can give the solution to the Decision Diffie-Hellman. Given a tuple (g, g^x, g^y, g^z), it is clear that $e(g^x, g^y) = e(g, g)^{xy}$ and $e(g, g^z) = e(g, g)^z$. If $z \equiv xy \pmod{|G|}$, then the two pairings are equal, otherwise they are not.

We can also derive a real gap splitting problem from $\mathcal{SOU}_{E/\mathbb{F}_{p'}}$ discussed in Section 4.5.6, when the elliptic curve E is chosen such that it is possible to compute a pairing on E. Suppose $\#E(\mathbb{F}_{p'}) = n = pq$, and P_1 and P_2 are points of order p and q, respectively. Then $P = P_1 + P_2$ is a generator for $E(\mathbb{F}_{p'})$, and

for some $a, b \in \mathbb{Z}$, $P_1 = aqP$ and $P_2 = bpP$. It is then clear that we can use the pairing to decide subgroup membership. If a point Q is in $\langle P_1 \rangle$, then $Q = a'qP$ for some $a' \in \mathbb{Z}$, and we get that $e(Q, P_2) = e((a'q)P, bpP) = e(P, P)^{a'bn} = 1$.

We describe the key encapsulation method Λ_2 in Figure 6.3.

Key generation.
Input: $\mathcal{TSP}_\Gamma$, τ.

1. $(G, K, H, \sigma) \leftarrow \Gamma_\tau$.
2. Initialise random oracles $h_1 : G \to \{0,1\}^t$ and $h_2 : \{0,1\}^{2t} \to \{0,1\}^t$.
3. $pk \leftarrow (G, K, H, h_1, h_2)$.
4. $sk \leftarrow (G, \sigma, h_1, h_2)$.
5. Output (pk, sk).

Output: A public-private key pair (pk, sk).

Encryption.
Input: $pk = (G, K, H, h_1, h_2)$.

1. $x \overset{\approx}{\leftarrow} K$, $y \overset{\approx}{\leftarrow} H$.
2. Output $(h_2(h_1(x), h_1(y)), xy)$.

Output: A key-ciphertext pair in $\{0,1\}^t \times G$.

Decryption.
Input: $sk = (G, \sigma, h_1, h_2)$, $c \in G$.

1. $(x, y) \leftarrow \sigma(c)$.
2. Output $h_2(h_1(x), h_1(y))$.

Output: A key in $\{0,1\}^t$.

Figure 6.3: The key encapsulation method Λ_2.

It is quite obvious that if the underlying splitting problem is hard, this KEM is secure against passive attacks, because the adversary has to compute the splitting of the ciphertext to get any information about the resulting hash value.

Theorem 6.2. *The key encapsulation method Λ_2 is secure against adaptive attacks in the random oracle model if and only if the gap splitting problem is hard.*

Proof. We are given an active adversary against the KEM, and also an oracle for deciding if a splitting is correct or not. First, we assume that we actually know the splitting algorithm σ, and we design an algorithm D that shall play the role of the decryption oracle and the random oracles.

Then we create a modified algorithm D' that answers the same queries, but without recourse to the splitting algorithm σ, and such that this algorithm looks like D from the adversary's point of view.

D answers queries as follows. Queries to h_1 and h_2 are handled exactly as a real random oracle would handle them (see Section 3.6.1), using a list L of query-answer pairs for the oracle h_1 and a list L' of query-answer pairs for h_2. Decryption oracle queries are handled by using the σ algorithm to split the ciphertext, then by querying itself for the relevant hash values.

This completes the description of the original algorithm, and it is clear that it will work exactly as the real oracles would work.

Now we describe the modified algorithm D'. The list L is split into three lists, L_G, L_K and L_H. Whenever a h_1-query x is received that has not previously been answered, the splitting oracle is used to check if x is in K or H. If it is in K, the response is recorded in L_K, likewise for H, and if it is not in $K \cup H$, then it is recorded in L_G.

The list L' remains the same.

We also need a third list L'' containing decryption oracle query-answer pairs. Whenever a new query c is received by the decryption oracle, it first checks to see if the lists L_K and L_H contain queries (x, b) or (y, b') such that (x, cx^{-1}) or (cy^{-1}, y) is a splitting of c. This can be done using the splitting oracle.

If there are two queries (x, b) and (y, b'), the algorithm makes the h_2-query (b, b') to itself, receiving an answer b''. It adds the query (c, b'') to L'', and outputs b''.

If there is only one query, say $(x, b) \in L_K$, but no query (y, b') in L_H, b' is sampled uniformly at random, the query (cx^{-1}, b') is inserted into L_H, and the algorithm makes the h_2-query (b, b') to itself, receiving an answer b''. It then adds the query (c, b'') to L'', and outputs b''.

If there are no such queries, b'' is sampled uniformly at random the query (c, b'') is added to L'', and b'' is output.

To preserve consistency, we now have to check when receiving a new h_1-query $x \in K$ if for any $(c, b'') \in L''$, (x, cx^{-1}) is a splitting. If not, we proceed normally, otherwise, we need to sample b uniformly at random and b' uniformly at random, subject to the restriction that there be no query $(b, b', b''') \in L'$, for some $b''' \neq b''$. Then we add (x, b) to L_K, (cx^{-1}, b') to L_H, (b, b', b'') to L'.

It is quite clear that the algorithm D' behaves consistently and is indistinguishable from D. Since it does not use the splitting algorithm σ, it is also clear that the adversary can run D' for himself. The end result is that the decryption oracle does not give the adversary anything he could not get from the splitting oracle.

Now we prove the reverse direction, by showing how we can use the decryption oracle to get a splitting oracle that is correct except with negligible error. So suppose we receive a query c and a potential splitting (x, y). Obviously, c is a valid ciphertext. If (x, y) really is the splitting, then $h_2(h_1(x), h_1(y))$ is the correct decryption of c. Otherwise, if (x', y') is the splitting of c, $(h_1(x), h_1(y))$ collides with $(h_1(x'), h_1(y'))$ with probability $\approx 2^{-2t}$. If it is distinct, the output of h_2 collides with probability $\approx 2^{-t}$. In other words, our simulated splitting oracle never answers "no" incorrectly, and answers "yes" incorrectly with negligible probability. □

We briefly consider the performance of this KEM, compared to Diffie-Hellman key encapsulation method (DH-KEM) [1, 16]. It is possible to prove DH-KEM secure in the random oracle model under a gap Diffie-Hellman problem, much as in the above proof.

Note that it would be possible to do DH-KEM in one of the two subgroups. The cost for DH-KEM is two subgroup exponentiations for encryption and one for decryption.

Encryption with Λ_2 costs the same as sampling one random element from each subgroup. Typically, this is done by exponentiation, but in some cases there are faster methods available. Decryption is done using one exponentiation in the group. This will typically cost as much as one exponentiation in each subgroup.

So we see that we can gain an advantage in encryption speed if sampling random elements in at least one group is faster than exponentiations in the subgroups.

When Λ_2 is instantiated with a Computational Diffie-Hellman problem $\mathcal{CDH}_{G'}$, where it is possible to quickly sample elements uniformly at random from G', the resulting scheme is is exactly as fast as DH-KEM in the subgroup H (which is isomorphic to G'). The drawback is that the ciphertexts are twice as long.

We can consider DH-KEM to be a special case of our construction. First we note that if (x, y) and (x', y') are in $G = G' \times G'$, it is clear that they have the same projection on K if and only if $x = x'$. We consider two ciphertexts to be equivalent if their first coordinates are the equal. Then we change the hash function h_2 so that it simply outputs the t first bits of its input. Finally, we change the decryption oracle, so that it refuses to decrypt ciphertexts that are equivalent to the challenge ciphertext.

It is now clear that the second coordinate of the ciphertext can be removed, and the resulting scheme is exactly DH-KEM.

When Λ_2 is instantiated with the Computational Composite Residuosity problem $\mathcal{CCR}_n$, sampling elements in H is essentially free, so encryption costs one exponentiation in K. Decryption can be done using one exponentiation in K and one in H (which is essentially free). This instantiation is therefore much faster at encryption than DH-KEM in K, and marginally slower at decryption.

However, it is clear that DH-KEM in K is exactly as hard as DH-KEM in a subgroup of $\mathbb{Z}_n^*$, and then DH-KEM becomes about twice as fast as Λ_2 for encryption, and four times as fast for decryption, because the numbers it works with are half the size. We also recall that $\mathcal{FACT}_n$ cannot be hard unless $\mathcal{CDH}_{\mathbb{Z}_n^*}$ is hard, and $\mathcal{CCR}_n$ cannot be hard unless $\mathcal{FACT}_n$ is hard. Therefore, DH-KEM in $\mathbb{Z}_n^*$ is faster than, and most likely not less secure than, Λ_2 instantiated with $\mathcal{CCR}_n$.

The final interesting case is to instantiate Λ_2 with the problem $\mathcal{GOU}'_n$ described in Section 4.5.7. Note that in this case, it would seem safe to make H fairly small, say about one quarter of the size of K. Exponentiations in H are therefore

much faster than in K, and splitting can be done with one exponentiation in H and one in K. This means that Λ_2 will be quite a lot faster at encryption than DH-KEM in K, and it will be slightly slower at decryption.

If we compare with DH-KEM in a subgroup G' of a finite field, however, it seems clear that the subgroup K has to be twice as big as G', and therefore that exponentiations in K take double the time required for exponentiations in G'. One may make the assumption that the splitting problem remains hard even if exponents are restricted to a much smaller size, but that seems to be stretching the security assumptions for $\mathcal{GOU}'_n$.

Our conclusion therefore seems to be that, even though we can find underlying group structures where Λ_2 will be faster than DH-KEM, there are group structures available for DH-KEM that are not available for Λ_2, and they make DH-KEM a better choice.

Just as for DH-KEM [1], it is possible to formulate a decision problem based on the gap splitting problem and a hash function.

Definition. Let (G, K, H, σ) be a trapdoor splitting problem, and let $h : G \times G \to \{0,1\}^t$ be a hash function.

The *hash splitting problem* is the following problem: The instances are pairs $(c, b) \in G \times \{0,1\}^t$, and the answer is 1 if $h(\sigma(c)) = b$, 0 otherwise. The instances are sampled by sampling c uniformly at random, then letting $b = h(c)$ with probability 1, otherwise sampling b uniformly at random.

The *oracle splitting problem* is the same problem, but any adversary against the problem gets access to a restricted oracle that computes h for all group elements except the challenge element.

It is quite clear that Λ_2 is secure against active attacks if the oracle splitting problem is hard for h. Indeed, it is just a reformulation of what must be true for the key encapsulation mechanism to be secure, and we may as well assume that our particular instantiation of Λ_2 is secure.

Chapter 7

A secure cryptosystem

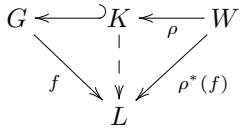
The goal of this chapter is to describe a practical public key cryptosystem that is secure against adaptive attacks in the standard model. To get the cryptosystem, we shall extend the construction of Cramer and Shoup [10] slightly.

7.1 Hash proof systems

The basic idea of Cramer and Shoup [10] is that if it is possible to define a function on some set G, but only give away a description of the function on a subset K, this is a useful primitive for designing cryptosystems. Our discussion follows [10] closely, even though it is superficially different.

Let G be a set, and let K be a subset of G. We say that W is a *witness set* for K if there is an easily computable bijection $\rho : W \to K$. This bijection allows one to prove that an element $x \in G$ really is in K by presenting an element $w \in W$ such that $\rho(w) = x$. (This obviously assumes that it is easy to recognise elements of W.)

For arbitrary sets S, S', denote by $\mathrm{Map}(S, S')$ the set of maps from S to S'. Let L be a group. We are interested in looking at maps from G to L. There is a natural map $\mathrm{Map}(G, L) \to \mathrm{Map}(K, L)$ given by restriction. The bijection ρ gives us a bijection $\rho^* : \mathrm{Map}(K, L) \to \mathrm{Map}(W, L)$. We also denote the natural map $\mathrm{Map}(G, L) \to \mathrm{Map}(W, L)$ as ρ^*. It is clear that the following diagram commutes:



A *projective hash family* is a tuple (G, K, L, W, ρ, M), where G is a set, K is a subset of G, L is a group, W is a witness set for K with isomorphism ρ, M is a subset of $\mathrm{Map}(G, K)$, and for any $f \in M$, the image of K under f is a subgroup of L. We also suppose that L has a subgroup L', such that $L' \cap f(K) = \{1\}$ and

$L = L'f(K)$. This gives us a subgroup membership problem $\mathcal{SM}_{(L,L')}$. We say that $\mathcal{SM}_{(L,L')}$ is the subgroup membership problem *associated* to (G, K) by the projective hash family.

Let (G, K, L, W, ρ, M) be a projective hash family, and let F be uniformly distributed in M. The projective hash family is ϵ-*universal* if for any $f \in \rho^*(M)$, $x \in G \setminus K$ and $y \in L$, we have that

$$\Pr[F(x) = y \mid \rho^*(F) = f] \leq \epsilon.$$

The projective hash family is ϵ-*universal-2* if for any $f \in \rho^*(M)$, $x_0 \in G \setminus K$, $x \in G \setminus (K \cup \{x_0\})$ and $y, y_0 \in L$, we have that

$$\Pr[F(x) = y \mid F(x_0) = y_0 \wedge \rho^*(F) = f] \leq \epsilon. \tag{7.1}$$

It is clear that ϵ-universal follows from ϵ-universal-2.

Let (G, K, L, W, ρ, M) be a projective hash family. Let F be uniformly distributed on M, let X be uniformly distributed in $G \setminus K$, and let Y be uniformly distributed in L'. Define the two random variables $U = (X, \rho^*(F), F(X))$ and $V = (X, \rho^*(F), F(X)Y)$. We say that the projective hash family is ϵ-*smooth* if

$$\mathrm{Dist}(U, V) \leq \epsilon.$$

Let $\mathcal{SM}_\Gamma$ be a subgroup membership problem. A *hash proof system* P for $\mathcal{SM}_\Gamma$ is a function that to each pair $(G, K) \in \Gamma$ assigns a tuple (L, W, ρ, M) such that (G, K, L, W, ρ, M) is a projective hash family.

An *extended hash proof system* $\hat{P}$ for $\mathcal{SM}_\Gamma$ is a function that to each pair $(G, K) \in \Gamma$ assigns a tuple $(S, \hat{L}, W, \hat{\rho}, \hat{M})$ such that $(G \times S, K \times S, \hat{L}, W, \hat{\rho}, \hat{M})$, where S is some set depending on G, is a projective hash family.

A hash proof system P is $\epsilon(\tau)$-smooth if the projective hash families derived using P are $\epsilon(\tau)$-smooth except with negligible (in τ) probability.

An extended hash proof system $\hat{P}$ is $\epsilon(\tau)$-universal-2 if the projective hash families derived using $\hat{P}$ are $\epsilon(\tau)$-universal-2 except with negligible (in τ) probability.

A hash proof system also has to provide algorithms to sample the sets W, M and $\hat{M}$. These algorithms should sample the sets δ-close to uniform, and we denote sampling using these algorithms by $w \stackrel{\approx}{\leftarrow} W$, $f \stackrel{\approx}{\leftarrow} M$, and $\hat{f} \stackrel{\approx}{\leftarrow} \hat{M}$.

7.2 The general construction

Suppose we have a subgroup membership problem $\mathcal{SM}_\Gamma$, a hash proof system P for $\mathcal{SM}_\Gamma$, and an extended hash proof system $\hat{P}$ for $\mathcal{SM}_\Gamma$ such that if (L, W, ρ, M) is the tuple assigned to (G, K) by P, then the tuple assigned to (G, K) by $\hat{P}$ is $(L, \hat{L}, W, \hat{\rho}, \hat{M})$.

We can derive a cryptosystem from the two hash proof systems. It is described in Figure 7.1.

Key generation.
Input: $\mathcal{SM}_\Gamma$, P, $\hat{P}$, τ.

1. $(G, K) \leftarrow \Gamma_\tau$.
2. $(L, W, \rho, M) \leftarrow P(G, K)$.
3. $(L, \hat{L}, W, \hat{\rho}, \hat{M}) \leftarrow \hat{P}(G, K)$.
4. $f \overset{\approx}{\leftarrow} M$.
5. $\hat{f} \overset{\approx}{\leftarrow} \hat{M}$.
6. $sk \leftarrow (G, L, \hat{L}, f, \hat{f})$.
7. $pk \leftarrow (G, W, L, \hat{L}, \rho, \rho^*(f), \hat{\rho}^*(\hat{f}))$.

Output: (pk, sk).

Encryption.
Input: $pk = (G, W, L, \hat{L}, \rho^*(f), \hat{\rho}^*(\hat{f}))$, $m \in L$.

1. $w \overset{\approx}{\leftarrow} W$.
2. $x \leftarrow \rho(w)$.
3. $y \leftarrow \rho^*(f)(w)$.
4. $e \leftarrow ym$.
5. $\hat{y} \leftarrow \rho^*(\hat{f})(w, e)$.

Output: $(x, e, \hat{y})$.

Decryption.
Input: $sk = (G, L, \hat{L}, f, \hat{f})$, $(x, e, \hat{y})$.

1. $\hat{y}' \leftarrow \hat{f}(x, e)$.
2. If $\hat{y}' \neq \hat{y}$, output $\perp$.
3. $y \leftarrow f(x)$.
4. $m \leftarrow ey^{-1}$.
5. Output m.

Output: A message m or $\perp$.

Figure 7.1: The cryptosystem Π_3 derived from the hash proof systems P and $\hat{P}$, and the subgroup membership problem $\mathcal{SM}_\Gamma$.

The security analysis closely follows the analysis in [10].

Suppose that P is ϵ-smooth, that $\hat{P}$ is ϵ'-universal-2, that the sampling algorithms for P and $\hat{P}$ are δ-close to uniform, and that the sampling algorithms for the subgroup membership problem are δ'-close to uniform.

Suppose $A = (A_1, A_2)$ is a chosen ciphertext adversary against Π_3. We shall use the following experiment to construct a distinguisher A' for (G, K).

Experiment 1.
Input: $A = (A_1, A_2)$, (G, K), P, $\hat{P}$, $x_0 \in G$.

1. $(L, W, \rho, M) \leftarrow P(G, K)$.
2. $(L, \hat{L}, W, \hat{\rho}, \hat{M}) \leftarrow \hat{P}(G, K)$.
3. $f \xleftarrow{\approx} M$, $\hat{f} \xleftarrow{\approx} \hat{M}$.
4. $sk \leftarrow (G, L, \hat{L}, f, \hat{f})$.
5. $pk \leftarrow (G, W, L, \hat{L}, \rho, \rho^*(f), \hat{\rho}^*(\hat{f}))$.
6. Initialise decryption oracle $\mathcal{D}_{sk}$.
7. $(m_0, m_1, s) \leftarrow A_1(pk)$, giving A_1 access to $\mathcal{D}_{sk}$.
8. $b \leftarrow \{0, 1\}$.
9. $y_0 \leftarrow f(x_0)$, $e_0 \leftarrow y_0 m_b$, $\hat{y}_0 \leftarrow \hat{f}(x_0, e_0)$.
10. Initialise restricted decryption oracle $\mathcal{D}'_{sk}$.
11. $b' \leftarrow A_2(pk, m_0, m_1, s, x_0, e_0, \hat{y}_0)$, giving A_2 access to $\mathcal{D}'_{sk}$.
12. If $b = b'$, output 1, otherwise output 0.

Output: 0 or 1.

Note that Steps 1–5 do exactly as the key generation algorithm would do.

Let T' be the event that Experiment 1 outputs 1 when the input x_0 is in K. Since Step 9 produces exactly the same result as the encryption algorithm when the input $x_0 \in K$, it is clear that the only difference between Experiment 1 and a real attack is that x_0 has been sampled uniformly from K, and not via the sampling algorithm for W. Since Experiment 1 outputs 1 when the adversary wins, we have that

$$\mathrm{Adv}_A^{CCA2} \leq |\Pr[T'] - 1/2| + \delta, \tag{7.2}$$

since the sampling algorithm for W is δ-close to uniform.

Let T be the event that Experiment 1 outputs 1 when the input x_0 is in $G \setminus K$. It is quite clear that from Experiment 1 we can derive an algorithm A' for distinguishing K from $G \setminus K$ such that

$$|\Pr[T'] - \Pr[T]| \leq \mathrm{Adv}_{A'}^{\mathcal{SM}_{(G,K)}}. \tag{7.3}$$

To analyse the event T, we shall make a series of modifications to Experiment 1. We number the modified experiments as $1'$, $1''$, etc. Note that these modifications need not be efficiently implementable.

First modification We change Step 3 so that f and $\hat{f}$ are sampled from the uniform distribution, and not using the algorithms provided by the hash proof systems.

Let T_1 be the event that Experiment $1'$ outputs 1 when the input x_0 is in $G \setminus K$. Since the algorithms provided by the hash proof systems were δ-close to uniform, we obviously have that

$$|\Pr[T] - \Pr[T_1]| \leq 2\delta. \tag{7.4}$$

Second modification We change the decryption oracles so that they refuse to decrypt a ciphertext $(x, e, \hat{y})$ if $x \notin K$. Let T_2 be the event that Experiment $1''$ outputs 1 when the input x_0 is in $G \setminus K$.

It is clear that this modification only affects the outcome if the adversary produces a valid ciphertext $(x', e', \hat{y}')$ with $x \notin K$, so $|\Pr[T_2] - \Pr[T_1]|$ is upper-bounded by the probability of this happening.

We first consider the decryption queries made by A_1. We condition on a fixed input $\rho^*(f)$ and $\hat{\rho}^*(\hat{f})$, and a fixed random tape. Note that this completely determines the output of A_1. If A_1 makes any decryption query $(x, e, \hat{y})$, with $x \notin K$, then since $\hat{P}$ is ϵ'-universal, we have that $(x, e, \hat{y})$ is valid with probability at most ϵ'.

Next we consider the decryption queries made by A_2. We additionally condition on a fixed value of b, the input x_0, the $\hat{y}_0$ computed in Step 9, along with the random tape of A_2. If A_2 makes any decryption query $(x, e, \hat{y})$, with $x \notin K$, we have two possibilities. If $(x, e) = (x_0, e_0)$, then we must have $\hat{y} \neq \hat{y}_0$, hence the ciphertext must be invalid. If $(x, e) \neq (x_0, e_0)$, then since $\hat{P}$ is ϵ'-universal-2, we have that $(x, e, \hat{y})$ is valid with probability at most ϵ'.

If A_1 and A_2 make Q decryption queries in total, we get that

$$|\Pr[T_2] - \Pr[T_1]| \leq Q\epsilon'. \tag{7.5}$$

Third modification We change Step 9 to be

9. $y' \leftarrow L'$, $y_0 \leftarrow f(x_0)$, $e_0 \leftarrow y_0 m_b y'$, $\hat{y}_0 \leftarrow \hat{f}(x_0, e_0)$.

Let T_3 be the event that Experiment $1'''$ outputs 1 when the input x_0 is in $G \setminus K$.

Since A_1 and A_2 cannot query the decryption oracle with ciphertexts $(x, e, \hat{y})$ where $x \notin K$, their only information about f is $\rho^*(f)$. Since P is ϵ-smooth, we get that

$$|\Pr[T_3] - \Pr[T_2]| \leq \epsilon. \tag{7.6}$$

Fourth modification We change Step 9 to be

9. $y' \leftarrow L \setminus L'$, $y_0 \leftarrow f(x_0)$, $e_0 \leftarrow y_0 m_b y'$, $\hat{y}_0 \leftarrow \hat{f}(x_0, e_0)$.

Let T_4 be the event that Experiment $1''''$ outputs 1 when the input x_0 is in $G \setminus K$.

It is quite clear that if y' had been sampled uniformly from L, then there would be no information about m_b present in the ciphertext, and the probability

that Experiment 1'''' output 1 when the input x_0 was in $G \setminus K$ would be 1/2. Since Experiment 1'''' samples from $L \setminus L'$, we get that

$$|\Pr[T_4] - 1/2| \leq \frac{2|L'|}{|L|} \tag{7.7}$$

by Lemma 2.2.

We need to bound $|\Pr[T_4] - \Pr[T_3]|$. To do this, we introduce another experiment.

Experiment 2.
Input: $A = (A_1, A_2)$, (G, K), P, $\hat{P}$, $y' \in L$.

Steps 1–8 are as in Experiment 1.

9. $x_0 \leftarrow G \setminus K$, $y_0 \leftarrow f(x_0)$, $e_0 \leftarrow y_0 m_b y'$, $\hat{y}_0 \leftarrow \hat{f}(x_0, e_0)$.

Steps 10–12 are as in Experiment 1.

Output: 0 or 1.

It is quite clear that we can repeat the two first modifications to Experiment 1 on Experiment 2, and the analysis remains the same. Let R' be the event that Experiment 2'' outputs 1 when the input y' is in L', and let R be the event that Experiment 2'' outputs 1 when the input y' is in $L \setminus L'$.

If the input y' to Experiment 2'' is in L', then it is quite clear that it behaves exactly as Experiment 1'''. Hence, $\Pr[R'] = \Pr[T_3]$.

If the input y' to Experiment 2'' is in $L \setminus L'$, then it is quite clear that it behaves exactly as Experiment 1''''. Hence, $\Pr[R] = \Pr[T_4]$.

It is also quite clear that we from Experiment 2 can derive an algorithm A'' to distinguish L' from $L \setminus L'$, by sampling x_0 not uniformly from $G \setminus K$, but via the subgroup membership problem's algorithms, and that

$$|\Pr[T_4] - \Pr[T_3]| = |\Pr[R] - \Pr[R']| \leq \mathrm{Adv}_{A''}^{\mathcal{SM}_{(L,L')}} + 2\delta + \delta' + Q\epsilon'. \tag{7.8}$$

Summing up Putting together (7.2)–(7.8), it is clear that we have proved the following theorem.

Theorem 7.1. *Let Π_3 be the cryptosystem described in Figure 7.1, based on a subgroup membership problem $\mathcal{SM}_{(G,K)}$ and hash proof systems P and $\hat{P}$. Let L be the group associated to G by P, and let L' be the subgroup of L. Suppose that P is ϵ-smooth, that $\hat{P}$ is ϵ'-universal-2, that the sampling algorithms for P and $\hat{P}$ are δ-close to uniform, and that the sampling algorithms for the subgroup membership problem are δ'-close to uniform. Then for any chosen ciphertext adversary A against Π_3, we have that*

$$\mathrm{Adv}_A^{CCA2}(\tau) \leq \mathrm{Adv}_{A'}^{\mathcal{SM}_{(G,K)}} + \mathrm{Adv}_{A''}^{\mathcal{SM}_{(L,L')}} + 4\delta + \delta' + 2Q\epsilon' + \epsilon + \frac{2|L'|}{|L|},$$

where A' and A'' are algorithms that invoke each stage of A once, and Q is the number of decryption queries made by A

7.3 Concrete instances

We shall show how to construct hash proof systems for certain symmetric subgroup membership problems. For completeness, we include proofs for this special case, even though the general case was done in [10].

Let $\mathcal{SSM}_{(G,K,H)}$ be a symmetric subgroup membership problem such that G is cyclic, and suppose that a generator g is available for K.

We shall describe a hash proof system P and an extended hash proof system $\hat{P}$ for $\mathcal{SSM}_{(G,K,H)}$. Let $W = \mathbb{Z}_{|K|}$ and $\rho([w]) = g^w$. Let $L = G$ and let $L' = H$. Since G is cyclic, the homomorphism group $\mathrm{Hom}(G,G)$ is isomorphic to $\mathbb{Z}_{|G|}$, and we let $M = \mathrm{Hom}(G,G)$. For any $f \in M$, a useful description of the function $\rho^*(f)$ is the group element $f(g)$, since for any $[w] \in W$, $f(g^w) = f(g)^w$. Then $P(G,K,H) = (G, \mathbb{Z}_{|H|}, \rho, \mathrm{Hom}(G,G))$.

Proposition 7.2. *The hash proof system P described above is ϵ-smooth, for $\epsilon \leq \frac{|H|-\phi(|H|)}{|H|} + \delta$, where the sampling algorithm for M is δ-close to uniform.*

Proof. Apply Lemma 4.12. □

The extended hash proof system $\hat{P}$ is slightly more complicated. The basic idea is to use the same construction as for P, but repeat it several times in parallel, and then combine the output. Let ℓ be the smallest prime factor in $|H|$. We shall suppose that for some sufficiently large l, a 1-1 function $h : G \times G \to \{0, \ldots, \ell-1\}^l$ is available.

Suppose we have a tuple $\vec{f} = (f_0, f_1, \ldots, f_l) \in \mathrm{Hom}(G,G)^{l+1}$. This tuple acts on G^{l+1} in the obvious way. h provides a 1-1 map $G \times G \to \mathbb{Z}_{|G|}^l$. We can extend this to a map $G \times G \to \mathbb{Z}_{|G|}^{l+1}$, given by $(x,e) \mapsto (1, \gamma_1, \ldots, \gamma_l)$. Since $\mathbb{Z}_{|G|} \simeq \mathrm{Hom}(G,G)$, we can let $h(x,e)$ act on G^{l+1}.

Now we compose the map $G \to G^{l+1}$ given by $x \mapsto (x, x, \ldots, x)$ first with $\vec{f}$, then with the action of $h(x,e)$, and finally with the l-fold multiplication map $(z_0, z_1, \ldots, z_l) \mapsto z_0 z_1 \ldots z_l$. This will be our $\hat{f}$. In other words, we consider the set of maps of the form

$$\hat{f}(x,e) = f_0(x) \prod_{i=1}^{l} f_i(x)^{\gamma_i},$$

where $h(x,e) = (\gamma_1, \ldots, \gamma_l)$, and $f_i \in \mathrm{Hom}(G,G)$.

The witness set for $K \times G$ is $\mathbb{Z}_{|K|} \times G$, and the map $\hat{\rho}$ is given by $\hat{\rho}([w],e) = (g^w, e)$, where g is a generator for K. It is clear that

$$\hat{\rho}^*(\hat{f})([w],e) = f_0(g)^w \prod_{i=1}^{l} f_i(g)^{w\gamma_i},$$

where $h(g^w, e) = (\gamma_1, \ldots, \gamma_l)$. So a useful description of the function $\hat{\rho}^*(\hat{f})$ is the tuple $(s_0, s_1, \ldots, s_l) = (f_0(g), f_1(g), \ldots, f_l(g))$.

Proposition 7.3. *The extended hash proof system $\hat{P}$ described above is $1/\ell$-universal-2.*

Proof. We work with the definition (7.1).

Let $(x, e, \hat{y}), (x_0, e_0, \hat{y}_0) \in G \times G \times G$, $(k_0, \ldots, k_l) \in \{0, \ldots, l-1\}$, $h(x, e) = (\gamma_1, \ldots, \gamma_l)$, and $h(x_0, e_0) = (\gamma_{0,1}, \ldots, \gamma_{0,l})$.

Let (x', x''), (x'_0, x''_0), $(\hat{y}', \hat{y}'')$ and $(\hat{y}'_0, \hat{y}''_0)$ be the splittings of x, x_0, $\hat{y}$ and $\hat{y}_0$, respectively. Let $k'_i = k_i \bmod |K|$ and $k''_i = k_i \bmod |H|$, for $0 \leq i \leq l$.

First of all, the knowledge of $s_0, \ldots, s_l$ fixes the values of $(k'_0, \ldots, k'_l)$. We may assume that

$$\hat{y}''_0 = (x''_0)^{k''_0 + \sum_{i=1}^{l} k''_i \gamma_{0,i}}, \tag{7.9}$$

since otherwise, (7.1) is trivially true. We would like to count how many tuples $(k_0, \ldots, k_l)$ satisfy this equation, and the proportion of those that satisfy

$$\hat{y}'' = (x'')^{k''_0 + \sum_{i=1}^{l} k''_i \gamma_i}, \tag{7.10}$$

or alternatively, for some integer a,

$$a \equiv k''_0 + \sum_{i=1}^{l} k''_i \gamma_i \pmod{m}, \tag{7.11}$$

where m is the order of x''. This will give us an upper bound on ϵ.

It is clear that $(k'_0, \ldots, k'_l)$ and $(k''_0, \ldots, k''_l)$ are distributed uniformly and independently when $(k_0, \ldots, k_l)$ is sampled uniformly. Therefore, the only information about $(k''_0, \ldots, k''_l)$ available is in (7.9).

First of all, (7.9) just contains information about $(k''_0, \ldots, k''_l)$ modulo the order of x''_0. Let m_0 be the order of x''_0, and let $m' = \gcd(m, m_0)$ and $m'' = m_0/m'$.

Suppose $m'' > 1$. (7.11) also holds modulo m'', and it is quite clear that for any tuple $(k''_1, \ldots, k''_l)$, there is exactly one value of k''_0 that satisfies the congruence, so at most $1/m$ of all tuples work. Since ℓ was the smallest prime dividing $|H|$, $1/m \leq 1/\ell$, and the result follows.

Suppose $m'' = 1$, and let ℓ' be the smallest prime dividing m. For some integer a_0, we get from (7.9) that

$$a_0 \equiv k''_0 + \sum_{i=1}^{l} k''_i \gamma_{0,i} \pmod{\ell'}. \tag{7.12}$$

The tuple $(k''_0, \ldots, k''_l)$ modulo ℓ' is uniformly distributed in the affine space $\mathbb{F}_{\ell'}^{l+1}$. (7.12) defines a hyperplane W_0 in $\mathbb{F}_{\ell'}^{l+1}$, and (7.11) defines a hyperplane W.

Now each γ_i and $\gamma_{0,i}$ is in $\{0, \ldots, \ell - 1\}$, and since h is 1-1, for at least one i, $\gamma_i \not\equiv \gamma_{0,i} \pmod{\ell'}$. This means that the two vectors $(1, \gamma_1, \ldots, \gamma_l)$ and $(1, \gamma_{0,1}, \ldots, \gamma_{0,l})$ are linearly independent modulo ℓ'. Therefore, the two affine spaces W and W_0 do not coincide and have non-empty intersection, and their intersection has codimension 1. Therefore, at most $1/\ell'$ of all tuples $(k''_0, \ldots, k''_l)$ satisfy both (7.9) and (7.10). Since $1/\ell' < 1/\ell$, the result follows. □

Instantiating the cryptosystem Π_3 with these hash proof systems and the corresponding symmetric subgroup membership problem, we get the cryptosystem described in Figure 7.2. It is quite clear that if the underlying symmetric subgroup membership problem is hard, then by Theorem 7.1, this variant of Π_3 is secure, because the subgroup membership problem $\mathcal{SM}_{(L,L')}$ that appears in Section 7.2 is simply $\mathcal{SM}_{(G,H)}$.

It is also possible to replace the 1-1 function h with a collision resistant hash function, and take $l = 1$. In this case, $\hat{P}$ is no longer ϵ-universal-2, so we need to adapt the proof to show that $\hat{P}$ satisfies a "computational" variant of ϵ-universal-2. We do not do this.

Concretely, this cryptosystem can be instantiated with the symmetric subgroup membership problems $\mathcal{SOU}_n$ (Section 4.5.6) and $\mathcal{GOU}_n$ (Section 4.5.7). Both yield quite practical and efficient cryptosystems, especially if a collision resistant hash function is used.

It is worth noting that if the extended hash proof system $\hat{P}$ is removed, we get the cryptosystem Π_2 from Section 5.3.

We note that for $\mathcal{SOU}_n$, the group order $|G| = n$ is known. Sampling uniformly from $W = \{0, \ldots, |K| - 1\}$ can be done using Lemma 2.3.

For $\mathcal{GOU}_n$, we know that the uniform distribution on $\{0, \ldots, \lfloor n/2 \rfloor\}$ reduced modulo $|G| = \phi(n)/2$ is almost uniform. Sampling uniformly from W can be done using Lemma 2.3. We note that 2 divides the group order, and because of Proposition 7.2 and Proposition 7.3, K must be chosen such that 2 divides $|K|$.

If we consider performance, and compare $\Pi_3/\mathcal{SOU}_n$ with ElGamal over $\mathbb{F}_p^*$, where p is a safe prime, Π_3 requires slightly more than 4 exponentiations in K to encrypt, while ElGamal requires 2 exponentiations in $\mathbb{F}_p^*$.

To decrypt, $\Pi_3/\mathcal{SOU}_n$ requires less than three exponentiations in K (under the assumption that the factorisation of n is known to the secret key holder), while ElGamal requires 1 exponentiation in $\mathbb{F}_p^*$. If the factorisation of n is not known, then decryption requires two exponentiations in G.

This means that $\Pi_3/\mathcal{SOU}_n$ requires roughly the same amount of work as ElGamal to encrypt, and either 1.5 or 2 times as much work to decrypt. And it gives us chosen ciphertext security, not just semantic security as ElGamal does.

When instantiating with $\mathcal{GOU}_n$, it is possible to make K much smaller than H. This leads to an even faster system for encryption, but unless the factorisation of n is known to the secret key holder, decryptions will cost between three and four exponentiations in G. If the factorisation is known, however, Chinese remainder theorem tricks are also available to speed up computations, giving a much faster system, even for decryption.

Compared to the instantiations of the Cramer-Shoup construction given in [10], our two instantiations are significantly faster, except for the elliptic curve variants of Cramer-Shoup. Asymptotically, the elliptic curve variants are significantly faster than our two variants.

Key generation.
Input: $\mathcal{SSM}_\Gamma$, τ.

1. $(G, K, H) \leftarrow \Gamma_\tau$. A generator g for K is assumed given.
2. $(k, k_0, k_1, \ldots, k_l) \stackrel{\approx}{\leftarrow} \{0, \ldots, |G| - 1\}^{l+2}$.
3. $(s, s_0, s_1, \ldots, s_l) \leftarrow (g^k, g^{k_0}, g^{k_1}, \ldots, g^{k_l})$.
4. $pk \leftarrow (G, g, s, s_0, s_1, \ldots, s_l, h)$.
5. $sk \leftarrow (G, k, k_0, k_1, \ldots, k_l, h)$.

Output: (pk, sk).

Encryption.
Input: pk, $m \in G$.

1. $w \stackrel{\approx}{\leftarrow} \{0, \ldots, |K| - 1\}$.
2. $x \leftarrow g^w$.
3. $y \leftarrow s^w$.
4. $e \leftarrow ym$.
5. $(\gamma_1, \ldots, \gamma_l) \leftarrow h(x, e)$.
6. $\hat{y} \leftarrow s_0^w \prod_{i=1}^{l} s_i^{w\gamma_i}$.

Output: $(x, e, \hat{y}) \in G \times G \times G$.

Decryption.
Input: sk, $(x, e, \hat{y}) \in G \times G \times G$.

1. $(\gamma_1, \ldots, \gamma_l) \leftarrow h(x, e)$.
2. $\hat{y}' \leftarrow x_0^k \prod_{i=1}^{l} x^{k_i\gamma_i}$.
3. If $\hat{y} \neq \hat{y}'$, then output $\perp$.
4. $y \leftarrow x^k$.
5. $m \leftarrow ey^{-1}$.

Output: A message $m \in G$ or $\perp$.

Figure 7.2: An instantiation of the cryptosystem Π_3.

When a key encapsulation method is all that is required, the Cramer-Shoup key encapsulation method [11] using a subgroup of a finite field will be significantly faster than our two constructions.

Chapter 8

Concluding remarks

Our presentation of subgroup membership problems, along with the related splitting and subgroup discrete logarithm problems, gives a uniform description and security analysis of a large class of public key cryptosystems, showing that they are all based on very similar principles.

Further, we have designed two key encapsulation mechanism based on a trapdoor splitting problem, and analysed their security in terms of the underlying splitting problem. To get security against chosen ciphertext attacks, we need the random oracle model and a strong assumption on the subgroup membership problem. Unfortunately, the concrete instances of this cryptosystem are in practice of moderate interest.

The most interesting part of this thesis is the study of symmetric subgroup membership problems. Theorems 4.11, 5.7 and 6.1 show how such an indistinguishability structure gives us very useful results.

Following the construction of Cramer and Shoup [10], we have designed a cryptosystem secure in the standard model against chosen ciphertext attacks. If a collision resistant hash function is used, the resulting cryptosystem is quite efficient when instantiated with a symmetric subgroup membership problem, and can compete with other instances of the Cramer-Shoup framework.

One problem is that, except for the Decision Diffie-Hellman problems, all of the subgroup membership problems we have presented depend crucially on factoring. While factoring is in general expected to be a hard problem, this dependency leads to relatively large groups when compared to cryptosystems based on elliptic curves. Finding subgroup membership problems, and especially subgroup discrete logarithm problems, that do not depend on factoring is a very interesting open problem.

Bibliography

[1] M. Abdalla, M. Bellare, and P. Rogaway. DHIES: An encryption scheme based on the Diffie-Hellman problem, 2001. `http://www.cs.ucsd.edu/users/mihir/papers/dhies.html`.

[2] M. Bellare, A. Boldyreva, and A. Palacio. An un-instantiable random-oracle-model scheme for a hybrid-encryption problem. Cryptology ePrint Archive, Report 2003/077, 2003. `http://eprint.iacr.org/`.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Proceedings of CRYPTO '98*, volume 1462 of *LNCS*, pages 26–45. Springer-Verlag, 1998.

[4] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. Cryptology ePrint Archive, Report 2004/008, 2004. `http://eprint.iacr.org/`.

[5] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[6] J. Benaloh. Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*, pages 129–128, 1994.

[7] D. Boneh. The Decision Diffie-Hellman problem. In *Proceedings of the Third Algorithmic Number Theory Symposium*, volume 1423 of *LNCS*, pages 48–63. Springer-Verlag, 1998.

[8] D. Catalano, R. Gennaro, and N. Howgrave-Graham. The bit security of Paillier's encryption scheme and its applications. In B. Pfitzmann, editor, *Proceedings of EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 229–243. Springer-Verlag, 2001.

[9] R. Cramer and V. Shoup. A practical public key cryptosystem secure against adaptive chosen cipher text attacks. In H. Krawczyk, editor, *Proceedings of CRYPTO '98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, 1998.

[10] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *Proceedings of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer-Verlag, 2002.

[11] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing*, 33(1):167–226, 2003.

[12] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Proceedings of CRYPTO '91*, volume 576 of *LNCS*, pages 445–456. Springer-Verlag, 1992.

[13] I. Damgård, J. Groth, and G. Salomonsen. The theory and implementation of an electronic voting system. In D. Gritzalis, editor, *Secure Electronic Voting*. Kluwer Academic Publishers, 2002.

[14] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136. Springer-Verlag, 2001.

[15] Y. Desmedt and A. M. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In H. C. Williams, editor, *Proceedings of CRYPTO '85*, volume 218 of *LNCS*, pages 516–521. Springer-Verlag, 1986.

[16] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[17] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, 1991.

[18] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[19] FIPS 180-1. *Secure hash standard*. Federal Information Processing Standards Publication. National Technical Information Service, Springfield, Virginia, April 1995.

[20] S. D. Galbraith. Elliptic curve Paillier schemes. *Journal of Cryptology*, 15(2):129–138, 2002.

[21] O. Goldreich. *Foundations of Cryptography, Basic tools*. Cambridge University Press, 2001.

[22] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, April 1984.

[23] S. Hada and T. Tanaka. On the existence of 3-Round zero-knowledge protocols. In H. Krawczyk, editor, *Proceedings of CRYPTO '98*, volume 1462 of *LNCS*, pages 408–423. Springer-Verlag, 1998.

[24] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Proceedings of ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.

[25] H. W. Lenstra, Jr. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[26] W. Mao. Fast Monte-Carlo primality evidence shown in the dark. Technical Report HPL-1999-30R1, HP Laboratories, October 1999.

[27] U. Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In Y. Desmedt, editor, *Proceedings of CRYPTO '94*, volume 839 of *LNCS*, pages 271–281. Springer-Verlag, 1994.

[28] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.

[29] D. Naccache and J. Stern. A new public key cryptosystem based on higher residues. In K. Nyberg, editor, *Proceedings of EUROCRYPT '98*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, 1998.

[30] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual Symposium on Theory of Computing*, 1990.

[31] J. M. G. Nieto, C. Boyd, and E. Dawson. A public key cryptosystem based on the subgroup membership problem. In S. Quing, T. Okamoto, and J. Zhou, editors, *Proceedings of ICICS 2001*, volume 2229 of *LNCS*, pages 352–363. Springer-Verlag, 2001.

[32] T. Okamoto and D. Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In K. Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, 2001.

[33] T. Okamoto and D. Pointcheval. REACT: Rapid enchanced-security asymmetric cryptosystem transform. In D. Naccache, editor, *Progress in cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–175. Springer-Verlag, 2001.

[34] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In K. Nyberg, editor, *Proceedings of EUROCRYPT '98*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, 1998.

[35] P. Paillier. Public-key cryptosystems based on composite degree residue classes. In J. Stern, editor, *Proceedings of EUROCRYPT '99*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, 1999.

[36] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Proceedings of CRYPTO '91*, volume 576 of *LNCS*, pages 433–444. Springer-Verlag, 1992.

[37] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[38] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949. `http://www.cs.ucla.edu/~jkong/research/security/shannon.html`.

[39] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Proceedings of EUROCRYPT '97*, volume 1233 of *LNCS*, pages 256–266. Springer-Verlag, 1997. Revised version available from `http://www.shoup.net`.

[40] J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 105 of *Graduate Texts in Mathematics*. Springer-Verlag, 1986.

[41] A. Yamamura and T. Saito. Private information retrieval based on the subgroup membership problem. In V. Varadharajan and Y. Mu, editors, *Proceedings of ACISP 2001*, volume 2119 of *LNCS*, pages 206–220. Springer-Verlag, 2001.

Index