Non-Constructivity in Security Proofs

by

Priya Soundararajan

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

Waterloo, Ontario, Canada, 2018

Author's Declaration

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

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

Abstract

In the field of cryptography, one generally obtains assurances for the security of a cryptographic protocol by giving a reductionist security proof, which is comprised of a reduction from breaking a mathematical problem (that is well-studied and widely believed to be intractable) to the breaking of the cryptographic protocol. While such reductions are generally constructive, some authors give non-constructive reductions (also called non-uniform reductions) in order to reduce the tightness gap of the reduction. However, in order to assess the concrete security that the proof provides, one also needs to assess the intractability of the underlying mathematical problem against non-constructive attacks. Unfortunately, there has been very little work in the literature on non-constructive attacks on these problems, and sometimes non-constructive attacks are found that are much faster than their constructive counterparts. Thus, it is sometimes very difficult to obtain meaningful security assurances about a cryptographic protocol from a non-constructive reductionist security proof.

In this thesis, we examine three instances of non-constructive security proofs for cryptographic protocols in the literature: (1) a password-based key derivation function; (2) an HMAC-related message authentication code scheme; and (3) a round-optimal blind signature scheme.

Acknowledgements

I would like to thank my advisor Alfred Menezes for motivating this thesis topic, patiently guiding me through its development, and painstakingly reading several drafts of it. I would like to thank my thesis readers Douglas Stebila and David Jao for their thoughtful suggestions that helped improve the presentation of this thesis significantly.

Thanks to Melissa and Carol for cheerfully helping me whenever I had questions, and making sure the thesis submission was hassle-free. Thanks to the Crypto group in the C&O department for organizing weekly seminars. Thanks to all my friends here at Waterloo and back home, for their support, good company and everything else; I owe it to you guys. To Prof. Somitra Sanadhya, for being a huge source of encouragement. Finally, thanks to my parents for their selfless support and love, without which this would not have been possible.

Table of Contents

Chapter 1

Introduction

Suppose that we have a cryptosystem designed for a specific purpose, and we would like to assess its "security". First, we need to provide a definition of security that makes sense in practice. Typically, a definition would state that any attacker with certain capabilities, trying to accomplish some goal, can succeed with at most a negligibly small probability. The definition requires a description of the attacker's capabilities. This could involve, for example, interactions with certain parties, or the ability to query an "oracle" and receive responses. The attacker's resources include the attacker's computational capabilities, the maximum number of queries it can make to an oracle, the number of interactions it can have with a party, etc. The attacker's goals are also suitably defined to capture security properties desired of the cryptosystem when deployed in real-world applications. For example, for encryption schemes, we can define the attacker's goal as retrieving the private key or decrypting a ciphertext.

Proving that a cryptographic scheme is secure generally takes the following approach, (1) Describing the attacker's capabilities and resources. (2) Defining a suitable notion of security with respect to the above attacker. (3) Stating assumptions that are backed by extensive research. (4) Proving that the cryptographic scheme satisfies the security definition under the stated assumptions.

1.1 Reductions in security proofs

Let us denote the problem of attacking a cryptosystem, as described in its security definition, as problem A. One way to obtain some assurance about the security of the cryptosystem is to see if it withstands extensive cryptanalysis. Another way is to provide a proof that for all attackers with certain capabilities and resources, it is "hard" to solve A. This is typically proved using a reduction from a problem B that is widely believed to be "hard", to problem A. In simple terms, a reduction algorithm from problem B to problem A is an algorithm that solves B, using an algorithm that solves A. If the algorithm for solving B is not allowed to view or modify the algorithm for solving A, but can only call the function to solve A, then it is called a *blackbox reduction*. If the reduction can view or modify the solution to A, then it is called a *non-blackbox reduction*. Assuming the conjecture that problem B is hard for adversaries with certain capabilities and resources, the reduction establishes a class of adversaries (with some capabilities and resources) for which problem A is hard. This is the basis of the proof-based security approach that is central to cryptography.

1.2 Types of proofs

Information-theoretic. Information-theoretic security refers to the setting where, roughly speaking, one is only concerned with the amount of information that an adversary can learn in its interaction with the cryptosystem, irrespective of the time required to execute such an adversary. The adversary is assumed to have unbounded computational power, but its other resources might be constrained, such as being allowed to only make a fixed number of queries to an oracle.

Complexity-theoretic. In complexity-theoretic security, one is concerned with the running time of the adversary. A cryptosystem is described in terms of a security parameter λ. The running time of the adversary and its advantage (which can be interpreted as the success probability of the adversary) are functions of λ and are meant to be interpreted in the language of complexity theory, where one is concerned with the asymptotic behavior of functions.

An *adversary* can be described as a Turing machine (possibly suitably modified to allow interactions) that takes (among other things) the security parameter λ as

input, performs some computation, and halts. The running time of such an adversary is the number of steps executed by the machine until it halts. An adversary is said to be *efficient* if its running time is *probabilistic polynomial time* (PPT) in λ. The advantage of the adversary in accomplishing its goal is provided in the security definition, and is a function of λ. We require this function to be "negligible". A function $f(n)$ is said to be negligible if, for every polynomial function $poly(n)$, there exists a positive integer N such that $f(n) \leq 1/poly(n)$ for all $n \geq N$. In other words, the function f is smaller than every inverse polynomial, for large enough n. So, in the security definition of a cryptosystem, we would like all efficient adversaries with certain capabilities to have negligible advantage.

A reduction from problem B to problem A is an adversary with input the security parameter λ that solves B, with the additional capability to make function calls to a "solver" for problem A. Each function call is counted as one time step. We would like to have an efficient reduction, i.e., the running time of the reduction must be PPT in λ. This would prove that if there exists a PPT solution for A, then there exists a PPT solution for B.

This approach of using complexity theory makes the analysis easier because the running times are not strictly bounded by a constant t, unlike the case for concrete security. However, this approach cannot be used directly to compute concrete parameters.

Concrete setting. In this setting, the running time of the adversary and its advantage are concrete values. A (t, ϵ)-adversary attacking a cryptosystem is an algorithm that runs in time at most t and succeeds in breaking the cryptosystem with advantage greater than ϵ. A problem or cryptographic scheme is said to be (t, ϵ)-hard (or (t, ϵ)-secure respectively) if the advantage of any adversary running in time t is at most ϵ.

Security reductions are typically described and analyzed in the Random Access Machine (RAM) model of computation whereby the cost of an algorithm is its running time in the RAM model plus its length. The length of the program is included in the cost, for otherwise one could include large tables in the program. This is also the metric considered in this thesis. The problem with the RAM metric is that it only considers the cost of running an attack, but not the cost of finding it. This underestimates the actual cost of an attack, whereby attacks with large precomputation can cost significantly less in the RAM metric. The impact of this observation

on security reductions, and possible ways to fix this are discussed in detail in [7]. Analyzing other cost metrics that avert this problem is not the focus of this thesis, however.

1.2.1 Tightness gaps

In the concrete setting, one can define the *tightness gap* of a security reduction. Consider a reduction from problem B (the underlying hard problem) to problem A (the security definition of the cryptosystem). For a (t, ϵ)-adversary against problem A, suppose that the reduction solves B in time t' and advantage ϵ'. The reduction proves that if there is no (t', ϵ')-adversary against problem B, then there cannot be a (t, ϵ)-adversary against problem A. We would like the reduction to have t'/ϵ' as small as possible for a given t, ϵ (it must be at least t/ϵ). The tightness gap of the reduction is defined as $t'\epsilon/t\epsilon'$. Security reductions aim for a small tightness gap, since this would mean that the computational effort required to attack the cryptosystem is almost the same as the effort required to solve the underlying hard problem.

1.2.2 Calculating parameters

The security parameter of a scheme is evaluated using a reductionist security proof in the following way. A typical reduction uses an attacker against the cryptosystem A that takes time t and succeeds with advantage ϵ, to construct an attacker for the underlying hard problem B that takes time t' and succeeds with advantage ϵ'. If the hypothesis of the security theorem is that B is (t', ϵ')-hard, then it implies that the scheme is (t, ϵ)-secure. Parameters are then chosen for the scheme assuming that the hypothesis is true. Therefore, it is necessary to assess the concrete hardness of the underlying problem B.

1.3 Idealized assumptions

In this section, we consider two idealized assumptions that are commonly used in security proofs and are relevant to the schemes considered in this thesis. If no idealized assumption is used in the analysis, the security proof is said to be in the *standard model*.

Random Oracle Model (ROM). The first idealized assumption we consider is called the random oracle model, or ROM for short. This represents a function that for any input, outputs a value in its range uniformly at random. If the same input is queried to the oracle twice, it must output the same value. Cryptographic hash functions are usually modeled as random oracles in security proofs.

There has been a difference of opinion in the cryptography community about the use of ROMs. From a theoretical perspective, it was shown by Canetti, Goldreich and Halevi [9] that there exists a cryptographic scheme that is secure in the random oracle model, but the scheme is insecure for any instantiation of the oracle with a concrete function. Koblitz and Menezes [23] argue that most of the negative theoretical results on the random oracle model (including the aforementioned one) require constructions that do not comply with standard cryptographic practice, and hence do not pose a real threat to the security of protocols that are evaluated in the random oracle model. Cryptographic schemes with proofs in the ROM are usually more efficient than schemes with proofs in the standard model.

Generic Group Model. The generic group model refers to the setting where a group $\mathbb{G}$ is represented by a random encoding (encoding here refers to a particular representation of the group elements). Further, an adversary can only perform group operations using an oracle, which takes as input the encodings of two group elements, and outputs the encoding of the group element obtained by applying the group operation to the two inputs. Such an adversary is called a *generic adversary*. The generic group model was introduced by Shoup [30] to prove a lower bound on the number of group operations required to solve the discrete logarithm problem for generic adversaries. In [22], it is argued that the generic group assumption is stronger than the random oracle assumption from a practical point of view.

1.4 Constructive vs. non-constructive adversaries

There are two types of computational adversaries we consider: *constructive* and *non-constructive*. Informally, a constructive adversary/algorithm is one for which an efficient way to describe it is known. A non-constructive adversary/algorithm is one which is known to exist, but for which no efficient method for describing it is known. [29] contains one of the earliest attempts to formalize constructivity in the context of

collision resistance for unkeyed hash functions. Attempts to formalize constructive adversaries in general can be found in [7].

Non-constructive adversaries can render some problems completely trivial. Consider the problem of finding a collision in the SHA-256 hash function. Since the input length is arbitrary, there must exist two inputs x_1, x_2 that have the same hash value.The *existence* of a fast algorithm that prints a collision for SHA-256 is known, namely one that has x_1, x_2 hardcoded, and which outputs the collision (x_1, x_2) for SHA-256 with probability 1. However, we do not know an efficient way to find such a collision. This type of an attack works for any hash function that is not injective.

In the complexity-theoretic setting, adversaries can also be modeled as *uniform* or *non-uniform*. A non-uniform adversary is one that receives a polynomial sized *advice string* depending on its input security parameter. Thus, for each security parameter, the adversary has a different algorithm to solve the problem, depending on the advice string. In constrast, a uniform adversary is a single algorithm that must work for all input values (or security parameter, in this case). The non-uniform adversary is very powerful since there is no requirement that the advice string can be generated efficiently. Therefore, a truly non-uniform adversary (one for which there is no known efficient way to generate the advice string from the security parameter) is essentially non-constructive. Under the same assumptions, proving that a cryptosystem is secure against all non-uniform adversaries yields stronger assurances than proving security against all uniform adversaries. Now, consider the following theorem statement.

Statement I. Under non-uniform assumptions (i.e. the underlying problem is hard for non-uniform adversaries), the cryptosystem is secure against non-uniform adversaries.

Statement II. Under uniform assumptions (i.e. the underlying problem is hard for uniform adversaries), the cryptosystem is secure against uniform adversaries.

The above two statements cannot be compared directly. We need to assess the hardness of the underlying problem against non-uniform adversaries to make sense of statement I. This is often not well studied, and in many cases in the literature, the hardness is only evaluated against uniform adversaries.

1.5 Our contributions

In this thesis we re-evaluate the security of two cryptographic schemes that use non-constructive security reductions: an NMAC-related scheme called 2-Lane NMAC, and a blind signature scheme. The security parameters we obtain from the analysis are larger than what is originally claimed in the two works, which evaluate security with respect to constructive attacks only on the underlying primitive. Further, since non-constructive attacks have not been sufficiently well studied in the cryptanalytic literature, we cannot claim with much confidence that the security parameters we compute are sufficient for the 128-bit security level, since there could exist much faster non-constructive attacks than the ones we consider. Cryptanalysts do not have sufficient motivation to investigate non-constructive attacks, since the cost of finding an attack matters in practice. This makes it challenging to satisfactorily evaluate the security of schemes that use non-constructive arguments.

In addition to the above schemes, we consider the use of an argument called coin-fixing for game-based security proofs, described by Bellare and Rogaway [6]. This is a non-constructive technique used to upper bound the advantage of any adversary with fixed computational resources. Coin-fixing is mainly used to eliminate adversarial adaptivity. It is not always applicable in every game-based proof, but when applicable, it helps simplify the analysis. However, it may result in a worse (i.e. larger) upper bound for the adversary's advantage when compared to a security theorem obtained without the use of coin-fixing. This is to be expected, since to obtain the upper bound, one considers the most powerful adversary that somehow "knows" how to fix the coins used in its randomness to maximize its probability of winning the security game. We show that this is the case in the security proof of a password-based key derivation function (PBKDF) scheme considered by Zhou et al. [34].

1.6 Outline of thesis

In Chapter 2, we consider a Password Based Key Derivation Function (PBKDF) scheme that was presented and analyzed by Zhou et al. [34]. The security proof of this scheme uses games, and a coin-fixing technique that was defined formally by Bellare and Rogaway [6]. However, we found that the application of this technique to the PBKDF scheme was incorrect. We provide a new security proof of the same result that does not use coin-fixing, based on the proof of a very similar KDF scheme

considered by Yao and Yin [32]. We then rework the security result of the PBKDF with a correct application of the coin-fixing technique. This results in a greater upper bound on the adversary's advantage against the PBKDF scheme (and thus a weaker security result) compared to the result obtained without coin-fixing. The adversary considered in the security definition is information-theoretic with unbounded computation, but is allowed a fixed number of queries to the random oracle. Therefore, there is no issue of constructivity vs non-constructivity in this case.

In Chapter 3, we describe 2-Lane NMAC, a variant of the MAC scheme NMAC that uses non-constructive arguments in its proof. The security of 2-Lane NMAC (in terms of the number of queries up to which the security result has any meaning) is evaluated in light of the fastest known non-constructive attack on the underlying primitives, and compared with the security level offered assuming the fastest known constructive attack. Not unexpectedly, the security level offered in the former is much lower than the latter. All but one non-constructive theorem can be converted into a constructive theorem, and wherever it is evident, we restate the security result and recalculate parameters in the constructive model. The constructive security result for 2-Lane NMAC we obtain is similar to the constructive theorem for GNMAC (a generalized version of NMAC). We do not know if the originally claimed security result of 2-Lane NMAC in [33] can be restored in the constructive setting.

In Chapter 4, we take a closer look at the efficiency claims made in a *round-optimal blind signature scheme* by Garg and Gupta [17]. We examine uses of non-constructivity in the security proofs, and re-calculate security parameters and efficiency by evaluating the underlying hard problems against the fastest known non-constructive attacks, wherever applicable. This results in significantly larger security parameters, and therefore lowered efficiencies than originally claimed.

In Chapter 5, we provide some concluding remarks and directions for future work.

Chapter 2

A Password Based Key Derivation Function

This chapter is about a Password Based Key Derivation Function (PBKDF) scheme that was presented and analyzed by Zhou et al. [34]. The security proof of this scheme uses games, and a coin-fixing technique that was defined formally by Bellare and Rogaway [6]. However, we found that the application of this technique to the PBKDF scheme was incorrect. We provide a new security proof of the same result that does not use coin-fixing, based on the proof of a very similar KDF scheme considered by Yao and Yin [32]. We then rework the security result of the PBKDF with a correct application of the coin-fixing technique. This results in a greater upper bound on the adversary's advantage against the PBKDF scheme compared to the result obtained without coin-fixing. The adversary considered in the security definition is information-theoretic with unbounded computation, but is only permitted a fixed number of queries to the oracle. Therefore, there is no issue of constructivity vs. non-constructivity in this case.

The organization of this chapter is as follows: The PBKDF of Zhou et al. is presented in §2.1. In §2.2, we give a brief introduction to the game-playing technique as described by Bellare and Rogaway [6]. We state the coin-fixing theorem for games in §2.2.1. In §2.3, we give a proof of the "weakly secure KDF" property of the PBKDF that doesn't use coin-fixing. In §2.4, we use the coin-fixing technique to give a slightly different result, and compare the theorems obtained.

2.1 Description

Password Based Key Derivation Functions (PBKDFs) are used to generate cryptographic keys selected from passwords. This is usually done by choosing an element from a password space PW at random (or at least, with sufficiently high entropy), and repeatedly applying an n-bit pseudorandom function a constant number of times. Iteratively applying the pseudorandom function increases the attacker's cost in brute forcing the key, and is known as *key stretching*.

The PBKDF, denoted by $F_p(s, c)$, has the following parameters:

- $H : \{0,1\}^* \to \{0,1\}^n$, a pseudorandom function (PRF).
- c, the number of times the PRF is iteratively applied, and is appropriately chosen to increase the time taken for brute force attacks on the PBKDF.
- PW, a set of "passwords" from which a password p is chosen and passed as an input to the PRF. It is assumed that all passwords in PW have the same length.
- salt s, which is a public value that changes with each use of the PBKDF, and is chosen from a set S of valid salts. It is assumed that all salts have the same length.
- n, the length of the output of the PBKDF.

The key generated can be used in other cryptographic applications. The purpose of adding a salt is to prevent an adversary from using a single precomputed table of inputs (passwords from PW) and outputs of the PBKDF to brute force a given n-bit key generated from the PBKDF.

The PBKDF considered in this chapter is of the type described above, and instantiates the PRF with a cryptographic hash function. It is assumed that each password is chosen from PW with equal probability. Further, it is assumed that $p||s$ is of bitlength at most n, where $p \in PW$ and $s \in S$. The scheme is formally described in Algorithm 2.1.

We prove the security of the PBKDF scheme described in Algorithm 2.1 with respect to the "weakly secure KDF" property as defined in [32]. We note that a very similar PBKDF was considered by Yao and Yin in [32], the only difference being that the index i was not appended to u_i when passed to H.

Algorithm 2.1 PBKDF scheme

$p \xleftarrow{\$} PW$
$s \leftarrow S$, where s is public
$u_0 \leftarrow p||s$
for $i = 0$ to $c - 1$ **do**
    $u_{i+1} \leftarrow H(u_i||i)$
end for
Return u_c

Definition 2.2 (weakly secure KDF)**.** *Let $F_p(s, c)$ be a PBKDF. Let $\mathcal{A}(t)$ be an adversary making at most t queries to H. We consider the experiment E that is described in Algorithm 2.3.*

Algorithm 2.3 Experiment E

1: $p \xleftarrow{\$} PW$
2: $s \leftarrow S$ (s is public)
3: $H \xleftarrow{\$} \{h : \{0,1\}^* \rightarrow \{0,1\}^n\}$
4: $b \xleftarrow{\$} \{0,1\}$
5: **if** $b = 0$ **then**
6:     $y_0 \leftarrow F_p(s, c)$
7: **else**
8:     $y_0 \xleftarrow{\$} \{0,1\}^n$
9: **end if**
10: $\mathcal{A}$ takes as input y_0.
11: **for** $i = 1$ to t **do**
12:     On $\mathcal{A}$'s query x_i to H, return $H(x_i)$
13: **end for**
14: $\mathcal{A}$ outputs either 0 or 1

The advantage of the adversary is given by

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{wKDF}}(n) = |\Pr\left[\mathcal{A}^E = 1|b = 1\right] - \Pr\left[\mathcal{A}^E = 1|b = 0\right]|.$$

Then $F_p(s, c)$ is considered to be a (t, ϵ)-weakly secure KDF if $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{wKDF}}(n)$ is at most ϵ for all adversaries $\mathcal{A}$ making at most t queries to H.

In the above definition, the case $b = 0$ corresponds to a string output from the KDF, and the case $b = 1$ corresponds to a random string.

2.2 Game-playing technique

The manuscript [6] was specifically written for the purpose of formalizing game-playing which had until then appeared in different works, but without much formalism. Since a comprehensive, rigorous definition of this technique will be too lengthy, we will give an informal description which will hopefully be enough to understand how it works.

A *program* is defined to be a finite, valid sequence of statements in some programming language. This is for the purpose of formalism, however, in our case we can assume that programs are written in pseudocode. A *game* is a program consisting of a collection of *procedures*. A game can contain three kinds of procedures, an *initialization* procedure, a finite sequence of procedures $P_1, \ldots, P_n$ $(n \geq 0)$ called *oracles*, and a *finalization* procedure. The initialization or finalization procedure may be absent in a game. For the sake of simplicity, we shall assume in this chapter that all games are comprised of an initialization phase P_0, a single oracle P, and a finalization phase P_1.

An *adversary* is a probabilistic polynomial time algorithm written as a program consisting of a single procedure, with the ability to query oracles. We can run a game G with an adversary $\mathcal{A}$, and this is denoted by $G^{\mathcal{A}}$. First, the initialization procedure of G possibly produces an output which is passed as an input to $\mathcal{A}$. Next, $\mathcal{A}$ executes its procedure, while making at most t queries to P. $\mathcal{A}$ may generate an output which is passed as an input to the finalization procedure, which then produces the final output of the game.

Suppose that we have an adversary $\mathcal{A}$ whose goal is to distinguish between two cryptographic systems A and B. Each system is described as a game. The task of distinguishing the two systems A and B is broken down into a finite sequence of suitably chosen games $A = G_0, G_1, \ldots, G_{n-1}, B = G_n$. The advantage of distinguishing between A and B is now upper bounded by the sum of the advantages of distinguishing (A and G_1), (G_1 and G_2), $\ldots$, (G_{n-1} and B). Typically, the intermediate games $G_1, \ldots, G_{n-1}$ are written in such a way that for some two consecutive games G_i and G_{i+1}, where $0 \leq i \leq n-1$, $G_i = S||\ bad \leftarrow \text{true}||X$ and $G_{i+1} = S||bad \leftarrow \text{true}||Y$ for some pseudocode S, X, Y, with X possibly different from Y. The variable *bad* is a flag that is set to true from its initial state of false, to indicate that the code following it is possibly different for the two games G_i and G_{i+1}, but the code preceding it is the same for the two games. In this chapter, we will only need to consider the case when *bad* is set to true once inside the oracle. The games G_i and G_{i+1} are

called *identical-until-bad-is-set* games. The following lemma can be used to prove Theorem 2.5.

Lemma 2.4 (after *bad* is set, nothing matters, Proposition 3, [6])**.** *Let G and H be identical-until-bad-is-set games, and let $\mathcal{A}$ be an adversary. Then* $\Pr\left[G^{\mathcal{A}} \text{ sets } bad\ \right] = \Pr\left[H^{\mathcal{A}} \text{ sets } bad\ \right]$.

The following is the main result of game-playing.

Theorem 2.5 (fundamental lemma of game-playing, Lemma 2, [6])**.** *Let G and H be identical-until-bad-is-set games, and let $\mathcal{A}$ be an adversary. Then*

$$|\Pr\left[G^{\mathcal{A}} \to 1\right] - \Pr\left[H^{\mathcal{A}} \to 1\right]| \leq \Pr\left[G^{\mathcal{A}} \text{ sets } bad\right]. \tag{2.1}$$

The quantity on the left hand side of inequality (2.1) is the advantage of the adversary in distinguishing the games G and H. Depending on how we define the games G, H, the probability of *bad* being set could vary. For the goal of distinguishing two systems, two differently designed pairs of games could give different upper bounds. Therefore, the prover's task is to design the games in such a way that the probability of bad being set is as small as possible, and in particular is not trivially 1.

Next, we will describe coin-fixing, which is one of the methods used to compute the probability of *bad* being set.

2.2.1 Coin-fixing

This section describes the coin-fixing argument used in game-based security proofs. The purpose of this proof technique is to reduce or eliminate adversarial adaptivity, i.e., to avoid the complexity of computing the probability of *bad* being set when taking into account the various possibilities for the adversary's input to the oracle, each input depending on the outputs of the previous queries. Roughly, what the coin-fixing theorem states is this: the probability of *bad* being set in a game can be upper bounded by the probability of *bad* being set in a modified game where the inputs to the oracle and its outputs are certain constants fixed beforehand, provided some conditions on the game are satisfied. We formally define this technique next.

The following description of the technique has been adapted from [6] to the case where there is only one oracle in the game. Let G be a game with an oracle P. Let

$\mathcal{A}$ be an adversary that makes at most t queries to P, where each query lies in the domain of P. We would like to modify $G^{\mathcal{A}}$ so that no oracles are used.

Eliminable oracles. We first describe the conditions under which the technique applies. Let P be of the following form:

Procedure $P(X)$
$i \leftarrow i + 1$
$X_i \leftarrow X$
$Y_i \leftarrow D_{X_1,\ldots,X_i,Y_1,\ldots,Y_{i-1}}$
S
Return Y_i

Here, i is an integer variable that is initialized to 0 before $\mathcal{A}$ makes its first query to P. The variables X_i and Y_i where $1 \leq i \leq t$, must appear in the game G as *l-values* (which means on the left hand side of an assignment/random assignment statement) only in the statements in P shown above. $D_{X_1,\ldots,X_i,Y_1,\ldots,Y_{i-1}}$ denotes a set that only depends on the variables $X_1, \ldots, X_i, Y_1, \ldots, Y_{i-1}$, by which we mean that D represents a piece of code that computes a set which only uses the variables $X_1, \ldots, X_i, Y_1, \ldots, Y_{i-1}$. S denotes an arbitrary compound statement. If P is of the above form, then it is said to *eliminable*.

Coin-fixing technique. Let P be an eliminable oracle. We define $\mathcal{C}$, the set of *valid query-responses* for P, in the following way. If there exists an adversary $\mathcal{A}$ that makes at most t queries to P for which there is a non-zero probability that $\mathsf{X}_1, \ldots, \mathsf{X}_t$ and $\mathsf{Y}_1, \ldots, \mathsf{Y}_t$ occur as queries and responses, respectively, in an execution of $G^{\mathcal{A}}$, then $(\mathsf{X}_1, \ldots \mathsf{X}_t, \mathsf{Y}_1, \ldots, \mathsf{Y}_t)$ is in $\mathcal{C}$. For a query-response $\mathsf{C} = (\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}_1, \ldots, \mathsf{Y}_t) \in \mathcal{C}$, we can define a modified game G_{C} called the *coin-fixed* game as follows. Starting from $G^{\mathcal{A}}$, replace every occurrence of X_i by X_i. Similarly, replace every occurrence of Y_i by Y_i. Eliminate the procedure for oracle P. At the beginning of the Finalize procedure (add a Finalize procedure if there is none), execute the statement "**for** $i = 1$ to t, S". Here i represents the index of the variables X_i and Y_i. Note that all occurrences of X_i and Y_i in S have been replaced by X_i and Y_i, respectively. Also note that the game G_{C} no longer depends on $\mathcal{A}$. We can now state the coin fixing lemma.

Lemma 2.6 (coin-fixing)**.** *Let G be a game with eliminable oracle P and a flag* bad*. Let $\mathcal{A}$ denote an adversary that makes at most t queries to P. Let $\mathcal{C}$ denote the set of*

valid query-responses for P, and for $\mathsf{C} \in \mathcal{C}$ let G_{C} be the coin-fixed game as defined above. We have that

$$\max_{\mathcal{A}} \Pr\left[G^{\mathcal{A}} \text{ sets } bad\right] \leq \max_{\mathsf{C} \in \mathcal{C}} \Pr\left[G_{\mathsf{C}} \text{ sets } bad\right].$$

2.3 Proof without coin-fixing

In order to model experiment E (Algorithm 2.3), we design two games G_0 and G_1 which correspond to the case where $b = 0$ and $b = 1$ in the experiment, respectively. Algorithm 2.7 describes the two games, with G_0 containing the underlined statement but G_1 omitting it. They are very similar to the games K and R described in [32] (albeit for a slightly different KDF scheme). Note that the *collision* flag records whether a collision for H has occured, whereas the *bad* flag records whether $H^c(u_0||0)$ has been computed. The collision flag is used to simplify the analysis.

There are a few points to note regarding the two games. In trying to simulate the hash function H, we assume that its output cannot be distinguished from a random string of the same length. Therefore, we are justified in first choosing a random string y of the correct length, and then setting that as the output for a given input x to the hash function. We also treat the function H as a black-box, in that the adversary cannot exploit its internal code. Hence, the proof that is provided in this section is in the *random oracle model*. If an adversary queries an input that has been previously queried to the oracle, we need to make sure that the output is the same as before. However, we can safely assume that a real-life adversary will make distinct queries to the oracle since a hash function is deterministic and will return the same output for the same input.

The advantage of the adversary can be rewritten as

$$\mathrm{Adv}^{\mathrm{wKDF}}_{\mathcal{A}(t)}(n) = |\Pr\left[\mathcal{A}^{G_0} = 1\right] - \Pr\left[\mathcal{A}^{G_1} = 1\right]|. \tag{2.2}$$

We can now state the main theorem of this section.

Theorem 2.8. *For an adversary $\mathcal{A}(t)$, the PBKDF scheme described in Algorithm 2.1 is a (t, ϵ)-weakly secure KDF where $\epsilon = (t/c)/|PW| + t^2/2^n$.*

Proof. Note that games G_0 and G_1 are identical until line 16, and are therefore *identical-until-bad-is-set* games. We can apply the fundamental lemma of game playing (Theorem 2.5) to obtain

$$|\Pr\left[\mathcal{A}^{G_0} = 1\right] - \Pr\left[\mathcal{A}^{G_1} = 1\right]| \leq \Pr\left[G_1^{\mathcal{A}} \text{ sets } bad\right].$$

Algorithm 2.7 Distinguishing games G_0 and G_1

1: $p \xleftarrow{\$} PW$
2: $s \leftarrow S$, where s is public
3: $y_0 \xleftarrow{\$} \{0,1\}^n$, y_0 is given to the adversary $\mathcal{A}$
4: $i \leftarrow 0$, $u_0 \leftarrow p||s$, $Y \leftarrow \{u_0, y_0\}$
5: $collision \leftarrow 0$, $bad \leftarrow 0$
6: **function** ORACLE $H(x)$
7: $\quad y \xleftarrow{\$} \{0,1\}^n$
8: $\quad$ **if** $y \notin Y$ **then**
9: $\quad\quad Y \leftarrow Y \cup \{y\}$
10: $\quad$ **else**
11: $\quad\quad collision \leftarrow 1$
12: $\quad$ **end if**
13: $\quad$ **if** $i < c-1$ and $x = u_i||i$ **then**
14: $\quad\quad i \leftarrow i+1$ and $u_i \leftarrow y$
15: $\quad$ **else if** $i = c-1$ and $x = u_{c-1}||c-1$ **then**
16: $\quad\quad bad \leftarrow 1$, $\underline{y \leftarrow y_0}$
17: $\quad$ **end if**
18: $\quad$ Set $H(x) \leftarrow y$ and return y.
19: **end function**
20: After at most t queries to H, $\mathcal{A}(t)$ outputs 0 or 1.

The goal now is to calculate $\Pr\left[G_1^{\mathcal{A}} \text{ sets } bad\right]$. Similar to what was done in [32], we can define a directed graph that represents $\mathcal{A}$'s queries and its responses in the following way. Let U be the set $\{0,1\}^n \cup \{p||s : \ p \in PW\}$. Consider a graph with vertex set $V = U \times \{0, 1, 2, \ldots, c\}$. The directed edge set of the graph is defined in the following way: there is an edge from (x, i) to $(y, i+1)$ if and only if $H(x||i) = y$, where $x, y \in U$ and $0 \leq i \leq c-1$. We shall assume that the adversary with the highest advantage only queries H with messages that are in the subset of vertices $W = U \times \{0, \ldots, c-1\}$ of V. (Since we have modelled H as a random function, the output of H on some $X \notin W$ is independently, randomly distributed from each $H(Y)$ for $Y \in W$. Furthermore, *bad* is set if and only if $i = c-1$ and some query of the adversary is $u_{c-1}||c-1$, i.e., the query is in W. Therefore, querying $X \notin W$ does not give any information about any of the u_i, $0 \leq i \leq c-1$, and the adversary may as well not make such a query.)

We can define a subgraph of the original graph based on the H-queries made by the adversary in the following way. If the message queried is $m||i$ where $m \in U$ and $0 \leq i \leq c-1$, then there is an edge from the vertex (m, i) to the vertex $(H(m||i), i+1)$. We shall refer to this subgraph as the "query graph" of the adversary. We can see that

$$\begin{aligned}
\mathrm{Adv}_{\mathcal{A}}^{\mathrm{wKDF}}(n) &\leq \Pr[bad = 1] \\
&= \Pr[bad = 1 | collision = 0] \cdot \Pr[collision = 0] \\
&\quad + \Pr[bad = 1 | collision = 1] \cdot \Pr[collision = 1] \\
&\leq \Pr[bad = 1 | collision = 0] + \Pr[collision = 1] \,.
\end{aligned}$$

The above probabilities are assessed with respect to the game $G_1^{\mathcal{A}}$. We first calculate $\Pr[collision = 1]$. This is the probability that there is a collision among $\{u_0, y_0\}$ and the outputs $y_1, \ldots, y_t$ to the t queries. The probability that the output to any of the t queries is equal to u_0 is at most $t/2^n$, since each output is randomly chosen. The probability that there is a collision among y_0 and the outputs to the t queries is at most $\binom{t+1}{2}/2^n$, since there are $\binom{t+1}{2}$ unordered pairs $\{y_i, y_j\}$, $0 \leq i \neq j \leq t$, among which a collision can occur, and each collision occurs with a probability of $1/2^n$ since all the y_i's are chosen randomly. Thus, we have

$$\Pr[collision = 1] \leq \binom{t+1}{2}/2^n + t/2^n = \frac{t^2 + 3t}{2^{n+1}}.$$

Assuming that $3t < t^2$, we have

$$\Pr[collision = 1] \leq t^2/2^n.$$

To calculate $\Pr[bad = 1|collision = 0]$, note that if $collision = 0$ the query graph must consist of distinct paths in which $(u_0, 0)$ can only occur as the first vertex in some path. Now, $\Pr[bad = 1|collision = 0]$ is the probability that there is a path of length c in the query graph starting at $(u_0, 0)$. There are a maximum of t/c paths of length c in the query graph, so we need to calculate the probability that at least one of them starts at $(u_0, 0)$. Since p is chosen randomly from PW, the probability that the starting vertex of a path is equal to $(u_0, 0)$ is $1/|PW|$. Given that there are at most t/c such paths, the probability that at least one of them starts at $(u_0, 0)$ can be upper bounded by the union bound to get

$$\Pr[bad = 1|collision = 0] \leq \frac{t/c}{|PW|}.$$

Therefore, we have

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{wKDF}}(n) \leq \frac{t/c}{|PW|} + \frac{t^2}{2^n}. \tag{2.3}$$

□

We can give an attack that almost matches the upper bound in (2.3). Consider the adversary A defined in Algorithm 2.9. The adversary randomly selects an element $\tilde{p}$ from the password space, affixes the salt s and an index $j = 0$ to it, and iteratively queries it to the hash function H c times, each time incrementing the index j. It checks if the final output equals the challenge string y_0. If it is not equal, the adversary tries a new $\tilde{p}$.

Let us calculate $\Pr\left[A^E = 1|b = 0\right]$. The adversary returns 1 if $\tilde{p}$ is equal to p at some iteration i of the outer loop (in which case $H(u_{c-1}||c-1) = y_0$), or if for some iteration i we have $p \neq \tilde{p}$ but still $H(u_{c-1}||c-1) = y_0$. The probability of the former is $(t/c)/|PW|$ and that of the latter is $(t/c)/2^n$. Next, we calculate $\Pr\left[A^E = 1|b = 1\right]$. The probability that $u_c = y_0$ is $1/2^n$, and the probability that A outputs 1 is $(t/c)/2^n$. So in this case, the advantage is $(t/c)/|PW|$, which shows that the upper bound in (2.3) is essentially tight, if $t^2 \ll 2^n$.

Let us compare Theorem 2.8 to the result for the weakly secure KDF property of the PBKDF defined in [32]. The upper bounds are exactly the same, suggesting that affixing the index while iteratively applying the hash function does not offer any security advantage in the random oracle model.

Algorithm 2.9 $A(y_0)$

```
Tried_passwords = ∅
for i = 1 to t/c do
    p̃ ←$ PW − Tried_passwords
    Tried_passwords ← Tried_passwords ∪ {p̃}
    u_0 ← p̃||s
    for j from 0 to c − 1 do
        u_{j+1} ← H(u_j||j)
    end for
    if u_c = y_0 then
        Return 1
    end if
end for
Return 0
```

2.4 Proof using coin-fixing

We now state a result for the weakly secure KDF property of the PBKDF that uses coin-fixing. We begin by noting that the two distinguishing games from the work by Zhou et al. [34] are exactly the same as the games G_0 and G_1 in Algorithm 2.7. We can now state the theorem.

Theorem 2.10. *For an adversary $\mathcal{A}(t)$ making at most t queries to H, the PBKDF scheme described in Algorithm 2.1 is a (t, ϵ)-weakly secure KDF where $\epsilon = (t - c + 1)/|PW|$.*

Proof. As in the proof of Theorem 2.8, the advantage of $\mathcal{A}$ can be bounded by the probability of *bad* being set in one of the games G_0 or G_1 since they are *identical-until-bad-is-set* games and Theorem 2.5 can be applied. We have

$$|\Pr\left[\mathcal{A}^{G_0} = 1\right] - \Pr\left[\mathcal{A}^{G_1} = 1\right]| \leq \Pr\left[G_1^{\mathcal{A}} \text{ sets } bad\right].$$

Let us compute the right hand side of the above inequality. One can observe that the oracle H in the game G_1 is eliminable, as per the definition of eliminable oracles in §2.2.1. Applying the coin-fixing lemma to game G_1 we get that

$$\Pr\left[G_1^{\mathcal{A}} \text{ sets } bad\right] \leq \max_{\mathsf{C} \in \mathcal{C}} \Pr[G_{1,\mathsf{C}} \text{ sets } bad]$$

where $G_{1,\mathsf{C}}$ is the non-interactive game defined in Algorithm 2.11. Let $\mathsf{C} \in \mathcal{C}$ and let the queries corresponding to C be $(\mathsf{X}_1, \ldots, \mathsf{X}_t)$ and the corresponding responses be $(\mathsf{Y}_1, \ldots, \mathsf{Y}_t)$. We would like to upper bound $\Pr[G_{1,\mathsf{C}} \text{ sets } bad]$. For the flag *bad* to

Algorithm 2.11 Coin-fixed game $G_{1,\mathsf{C}}$ with $\mathsf{C} = (\mathsf{X}_1, \ldots, \mathsf{X}_t, \mathsf{Y}_1, \ldots, \mathsf{Y}_t)$

$p \xleftarrow{\$} PW$
$s \leftarrow S$, s is public
$y_0 \xleftarrow{\$} \{0,1\}^n$, y_0 is an input to the adversary
$i \leftarrow 0$, $u_0 \leftarrow p||s$, $Y \leftarrow \{u_0, y_0\}$
$collision \leftarrow 0$, $bad \leftarrow 0$
for $j = 1$ to t **do**
    if $\mathsf{Y}_j \notin Y$ **then**
        $Y \leftarrow Y \cup \{\mathsf{Y}_j\}$
    else
        $collision \leftarrow 1$
    end if
    if $i < c-1$ and $\mathsf{X}_j = u_i||i$ **then**
        $i \leftarrow i+1$ and $u_i \leftarrow \mathsf{Y}_j$.
    else if $i = c-1$ and $\mathsf{X}_j = u_{c-1}||c-1$ **then**
        $bad \leftarrow 1$
    end if
end for

be set to true in $G_{1,\mathsf{C}}$, C must have the following sequence of queries (not necessarily consecutive, but in the given order):

$$p||s||0,\ H(p||s||0)||1,\ H(H(p||s||0)||1)||2,\ \ldots,\ \underbrace{H(\ldots H(H(}_{c-1 \text{ times}} p||s||0)||1)\ldots)||c-1.$$

Define the set $U = \{p||s\ :\ p \in PW\} \cup \{0,1\}^n$. We say that a valid query-response C contains a *query chain* Q_i where $1 \le i \le t-c+1$ if there exists a query $\mathsf{X}_i = \mathsf{y}||0$ for some $\mathsf{y} \in U$ and C contains the sequence of queries

$$\mathsf{X}_i = \mathsf{y}||0,\ H(\mathsf{y}||0)||1,\ H(H(\mathsf{y}||0)||1)||2,\ \ldots,\ \underbrace{H(\ldots H(H(}_{c-1 \text{ times}} \mathsf{y}||0)||1)\ldots)||c-1$$

that occur in this order but not necessarily consecutively. Therefore, we can see that *bad* is set in $G_{1,\mathsf{C}}$ if and only if the following event occurs:

$$\bigcup_{1 \le i \le t-c+1} \{\mathsf{C} \text{ contains } Q_i \text{ and } \mathsf{X}_i = p||s||0\}.$$

The probability β of the above event occurring can be upper bounded using the union bound:

$$\begin{aligned}\beta &\leq \sum_{i=1}^{t-c+1} \Pr[\mathsf{C} \text{ contains } Q_i \text{ and } \mathsf{X}_i = p||s||0] \\ &= \sum_{i=1}^{t-c+1} \Pr[\mathsf{C} \text{ contains } Q_i |\ \mathsf{X}_i = p||s||0] \cdot \Pr[\mathsf{X}_i = p||s||0] \\ &\leq \sum_{i=1}^{t-c+1} \Pr[\mathsf{X}_i = p||s||0]\,.\end{aligned}$$

Since p is chosen randomly, $\Pr[\mathsf{X}_i = p||s||0]$ is at most $1/|PW|$ for any $1 \leq i \leq t-c+1$. Thus, for any $\mathsf{C} \in \mathcal{C}$ we have

$$\Pr[G_{1,\mathsf{C}} \text{ sets } bad] \leq (t-c+1)/|PW|.$$

What remains to show is a valid query-response that achieves this upper bound. Let $\mathbf{C}$ be the concatenation of the following queries

$$(\mathsf{x}_1||s||0,\ \mathsf{x}_2||s||0,\ \ldots,\ \mathsf{x}_{t-c+1}||s||0,\ \mathsf{y}_1||1,\ \mathsf{y}_2||2,\ \ldots, \mathsf{y}_{c-1}||c-1)$$

and the corresponding responses

$$(\underbrace{\mathsf{y}_1, \ldots, \mathsf{y}_1}_{t-c+1 \text{ times}},\ \mathsf{y}_2,\ \mathsf{y}_3,\ \ldots,\ \mathsf{y}_c)$$

where x_i's are chosen randomly from PW and are all distinct (this part of the query can be thought of as trial-and-error to hit the value of p used in the game), the y_i's for $1 \leq i \leq c-1$ are chosen randomly from $\{0,1\}^n$, and $y_c = y_0$. We will show that $\mathbf{C}$ maximizes $\Pr[G_{1,\mathsf{C}} \text{ sets } bad]$. First, $\mathbf{C}$ is indeed a valid query-response for the oracle H since there exists an adversary that makes the above queries and the oracle outputs the above responses with a non-zero probability, since the outputs are chosen randomly. Next, we compute the probability of *bad* being set. Note that $\mathbf{C}$ contains a query chain, where the first query in the chain can be any of the first $t-c+1$ queries, and the last $c-1$ queries of the chain are the last $c-1$ queries of $\mathbf{C}$. Since all the first $t-c+1$ queries map to y_1, *bad* is set if and only if any of them is equal to $p||s||0$. The probability that any of the first $t-c+1$ queries equals $p||s||0$ is $(t-c+1)/|PW|$, since p is chosen randomly. This completes the argument. $\square$

Note that provided $t^2 \ll 2^n$, the upper bound in Theorem 2.8 is better than the upper bound in Theorem 2.10 by a factor of approximately c. In [34], the same result as Theorem 2.8 is obtained, however the attempt to use coin-fixing was incorrect. In order to use it correctly, one must consider a query/response set that, when fixed in the game G_1, maximizes the probability of *bad* being set in the coin-fixed game. Since the authors of [34] did not consider the query/response set $\mathbf{C}$ that we described in the above proof, they obtained an (incorrect) upper bound that was significanty lower, namely $(t/c)/|PW|$.

Coin-fixing is a technique used in games primarily to eliminate complications that may arise in upper bounding the probability of *bad* being set when an adversary can query an oracle adaptively. Informally, it involves fixing certain coins in the game and the adversary that maximize the probability of the variable *bad* being set to 1 in the coin-fixed game. The coin-fixed adversary may be non-constructive, and hence, the probability of setting *bad* in the coin-fixed game is usually higher than what is obtained in the non-coin-fixed game (this is to be expected). However, in the PBKDF scheme considered here, we cannot make the distinction of a constructive/non-constructive adversary, since we are not concerned with its running time.

In conclusion, we investigated the use of coin-fixing in the security proof of a PBKDF scheme considered by Zhou et al. [34]. We found the use of coin-fixing to be incorrect and provided a revised proof. Further, we observed that the result obtained without the use of coin-fixing (Theorem 2.8) was stronger than the result obtained using coin-fixing (Theorem 2.10). In the PBKDF scheme considered in this chapter, since it is possible to obtain an upper bound without the use of coin-fixing that is almost tight with a matching attack, the use of coin-fixing is not necessary in the proof, and results in a weaker security result than what can be obtained without its use.

Chapter 3

2-Lane NMAC

In this chapter, we describe 2-Lane NMAC, a variant of NMAC that uses non-constructive arguments in its proof. The security of 2-Lane NMAC (in terms of the number of queries up to which the security result has any meaning) is evaluated in light of the fastest known non-constructive attack on the underlying primitives, and compared with the security level offered assuming the fastest known constructive attack. Not unexpectedly, the security level offered in the former is much lower than the latter. All but one non-constructive theorem in this scheme can be converted to a constructive theorem, and wherever it is evident, we restate the security result and recalculate parameters in the constructive model. The constructive security result for 2-Lane NMAC we obtain is similar to the constructive theorem for GNMAC. We do not know if the originally claimed security result of 2-Lane NMAC in [33] can be restored in the constructive setting.

The chapter is organized as follows. In §3.1 we provide some cryptographic definitions that are used in the rest of the chapter. In §3.2, we state two theorems on the prf security of NMAC provided by Bellare [5]. In §3.3 we discuss a scheme called L-Lane NMAC by Yasuda [33], based on Bellare's NMAC construction. In §3.4 we state the theorem on prf security of 2-Lane NMAC from [33], and discuss the use of non-constructivity in the proofs. In §3.5 we provide a constructive proof of the prf-security of 2-Lane NMAC and compare the constructive and non-constructive theorems. In §3.6, we calculate parameters for 2-Lane NMAC under different assumptions, and compare the values obtained for the different cases.

3.1 Preliminaries

3.1.1 MAC schemes

MAC stands for *Message Authentication Code*, which is a cryptographic scheme used to certify the authenticity and integrity of data transmitted between two communicating parties. The following three steps describe how a MAC works:

1. Key generation. $\mathsf{KeyGen}(1^n) = k$: On input 1^n, a key k is generated and shared between the two parties.
2. Tag generation. $\mathsf{MAC}_k(m) = t$: The sender, who wishes to communicate a message m, generates a tag t using the shared key k.
3. Verification. $\mathsf{SignVer}_k(m', t') =$ true or false: The receiver, upon receiving a message m' and tag t', regenerates a tag $t'' = \mathsf{MAC}_k(m')$, and checks if $t' = t''$ whereupon it accepts the message m', otherwise, rejects it.

MACs are designed to guarantee two properties: *data origin authentication*, which is to certify that the received message came from a trusted party (in terms of the protocol above, came from a party who knows the secret key k), and *data integrity* which is to certify that the message m transmitted by the sender is exactly the same as the message m' obtained by the receiver. Note that MAC schemes differ from digital signature schemes in that they do not provide *non-repudiation*, which is the property wherein the party that "signed" a message cannot deny that it indeed "signed" it.

In order to describe the security of a MAC scheme, the following definition is often used. It must be *computationally infeasible* for an adversary, given access to a *MAC oracle*, denoted by MAC_k, to produce a *MAC forgery*. There are three types of access the adversary can have to MAC_k.

- *Known-message*: The adversary is given a set of message-tag pairs beforehand, and is not allowed access to MAC_k.
- *Chosen-message*: The adversary is allowed to query MAC_k a sequence of messages $m_1, \ldots, m_s$ to get their corresponding tags $t_1, \ldots, t_s$.
- *Adaptive chosen-message*: The adversary is allowed to query MAC_k with a sequence of messages $m_1, \ldots, m_s$ to receive their corresponding tags $t_1, \ldots, t_s$, where each message m_i can be chosen by the adversary based on its previous messages $m_1, \ldots, m_{i-1}$ and tags $t_1, \ldots, t_{i-1}$.

There are two types of MAC-forgeries one can consider:

- *Existential*: The adversary has to produce a message m and a corresponding tag t such that $\mathsf{SignVer}_k(m,t) = 1$, where m has not been queried to MAC_k before.
- *Universal*: The adversary is required to produce a message-tag pair (m,t), for an m that is provided to it, such that $\mathsf{SignVer}_k(m,t) = 1$. Again, the adversary is not allowed to query m to MAC_k.

The strongest definition of security that one can describe for a MAC scheme is existential unforgeability against an adaptive, chosen-message adversary. In the remainder of this chapter, we will only consider this notion of security.

3.1.2 Cryptographic ingredients

Notation. We will use the following notation throughout this section. Let $B = \{0,1\}^b$, and let B^+ denote the set of binary strings which are of length some multiple of b bits. Let D denote the set of all strings up to a fixed, maximum length. Let *pad* be a fixed, public function, known as the length *padding* function, which converts any string $s \in \{0,1\}^*$ to $s^* = s||pad(|s|) \in B^+$, where $|s|$ denotes the bitlength of s. Let $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ be a *compression* function, and M be a message $m_1 \dots m_\ell$ where each $m_i, 1 \le i \le \ell$, is a b-bit message block. Let T_h denote the time taken for one compression function computation. Let $h^* : \{0,1\}^c \times B^+ \to \{0,1\}^c$ be the iterated application of h on M defined as follows: $h^*(K,M) = H_\ell$, where $H_0 = K$ and $H_i = h(H_{i-1}, m_i)$ for $1 \le i \le \ell$.

Computationally almost universal (cAU) functions. Let h be a family of keyed functions $h : \{0,1\}^c \times \mathit{Domain} \to \mathit{Range}$. A (t,μ)-cAU adversary of h is an algorithm that runs in time t and outputs two messages $M_1, M_2 \in \mathit{Domain}$ each of bitlength at most μ. The advantage of such an adversary $\mathcal{A}$ is

$$\mathrm{Adv}_h^{\mathrm{cAU}}(\mathcal{A}) = \Pr\left[h(K,M_1) = h(K,M_2) \;\middle|\; M_1, M_2 \leftarrow \mathcal{A}, K \xleftarrow{\$} \{0,1\}^c\right].$$

The function family h is (t,μ,ϵ)-cAU if $\mathrm{Adv}_h^{\mathrm{cAU}}(\mathcal{A}) \le \epsilon$ for all (t,μ)-cAU adversaries $\mathcal{A}$ of h. We write $\mathrm{Adv}_h^{\mathrm{cAU}}(t,\mu) = \epsilon$ if the maximum advantage of all (t,μ)-cAU adversaries of h is ϵ.

In the definition of a cAU adversary used in the L-Lane NMAC proof in [33], time is not taken into account since the author assumes that one is working in the non-uniform computational model. Thus, the parameter t is omitted and, for example, one writes $\mathrm{Adv}_h^{\mathrm{au}}(\mu)$ instead of $\mathrm{Adv}_h^{\mathrm{cAU}}(t, \mu)$.

Pseudorandom function (prf) function. Let $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}^c$ be a keyed family of functions, where $\{0,1\}^c$ is the key space. A (t, q)-prf adversary $\mathcal{A}$ of h is an algorithm that runs in time t and makes at most q queries to an "oracle". The oracle with probability $\frac{1}{2}$ is a randomly chosen function from b bits to c bits (i.e. $R(\cdot)$, where $R \xleftarrow{\$} \{\{0,1\}^b \to \{0,1\}^c\}$) or with probability $\frac{1}{2}$ is $h(K, \cdot)$ for a randomly chosen K. The adversary $\mathcal{A}$ outputs either 0 or 1, and its advantage is defined as

$$\mathrm{Adv}_h^{\mathrm{prf}}(\mathcal{A}) = |\Pr\left[\mathcal{A}^{R(\cdot)} \to 1\right] - \Pr\left[\mathcal{A}^{h(K,\cdot)} \to 1\right]|.$$

A function family h is a (t, q, ϵ)-secure prf if $\mathrm{Adv}_h^{\mathrm{prf}}(\mathcal{A}) \leq \epsilon$ for all (t, q)-prf adversaries $\mathcal{A}$ of h. Each query to the oracle is counted as one time-step. We write $\mathrm{Adv}_h^{\mathrm{prf}}(t, q) = \epsilon$ if the maximum advantage of all (t, q)-prf adversaries of h is ϵ.

One can also define the notion of a secure prf for a function family $h : \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^c$. In this case, there is an additional parameter μ which is the maximum of the bitlength of a message. Thus, one considers (t, q, μ)-prf adversaries, and writes $\mathrm{Adv}_h^{\mathrm{prf}}(t, q, \mu)$.

Iterated Merkle-Damgård hash function. The collision resistance properties of this construction for a hash function was proved independently by Ralph Merkle [26] and Ivan Damgård [13]. Such hash functions $H : \{0,1\}^c \times \{0,1\}^* \to \{0,1\}^c$ are defined as

$$H(IV, M) = h^*(IV, M||pad(|M|)).$$

For the constructions in this chapter, we can assume that the IV is a c-bit string chosen randomly, and is fixed and public. Hash functions implemented in practice have a hard-coded IV as their c-bit input, and hence it is not possible to change this value. There also exist hash functions in practice for which the final output is strictly smaller than c bits; examples include SHA384 which is SHA512 with its 512 bit output truncated to 384 bits and using a different IV.

Nested message authentication code (NMAC^H) ([5]). This is a function from $\{0,1\}^{2c} \times \{0,1\}^* \to \{0,1\}^c$ that uses an iterated Merkle-Damgård hash function H.

The first input of NMAC^H is a concatenation of two c-bit keys K_1 and K_2, and the second is a message M of variable length. It is defined as

$$\text{NMAC}^H(K_1||K_2, M) = H(K_1, H(K_2, M)).$$

Hash-based message authentication code (HMAC^H) ([5]). This MAC construction overcomes NMAC's problem of having to key the hash function with a variable IV. Let ipad and opad be fixed, public b-bit strings. Let K be a b-bit key and M a message of variable length. The function $\text{HMAC}^H : \{0,1\}^b \times \{0,1\}^* \to \{0,1\}^c$ is defined as

$$\text{HMAC}^H(K, M) = H(IV, (K \oplus \text{opad})||(H(IV, (K \oplus \text{ipad})||M))). \tag{3.1}$$

This construction for HMAC^H has been taken from [5] where it was proved that HMAC^H is a prf when keyed through the b-bit input. However, the version of HMAC standardized in [25] has a minimal recommended key length of c bits. Here, the c-bit key K is converted to a b-bit string by appending $b - c$ zeroes, which is then used in the HMAC^H construction shown. The standardized version also allows the bitlength of K to exceed b bits, in which case the key $K' = H(IV, K)||0^{b-c}$ is used in place of K in HMAC^H. A discussion of the security issues that arise from the different key sizes considered in Bellare's HMAC proof [5] can be found in [24].

3.2 Bellare's NMAC proofs

Bellare [5] proved that NMAC is a pseudorandom function assuming that the compression function h is a prf. The proof establishes that h^*, the iterated compression function defined in §3.1.2, is computationally almost universal. This can be proved assuming that h is a prf. The following two lemmas describe these two results; there are two versions for each, a blackbox version and a non-blackbox version.

Lemma 3.1 (Lemma 3.1, [5]). *Let $B = \{0,1\}^b$, let $h : \{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions, and let $n \geq 1$ be an integer. Let $\mathcal{A}^*$ be a cAU-adversary against h^* that has time complexity at most t. Assume that the two messages output by $\mathcal{A}^*$ are at most n_1, n_2 blocks long respectively, where $1 \leq n_1, n_2 \leq n$. Then, there exists a prf-adversary $\mathcal{A}$ against h such that*

$$\text{Adv}^{\text{cAU}}_{h^*}(\mathcal{A}^*) \leq (n_1 + n_2 - 1) \cdot \text{Adv}^{\text{prf}}_h(\mathcal{A}) + \frac{1}{2^c}. \tag{3.2}$$

$\mathcal{A}$ makes at most two oracle queries, has time complexity t under a blackbox reduction, and $O(nT_h)$ under a non-blackbox reduction where T_h is the time for one evaluation for h.

Lemma 3.2 (Lemma 3.2, [5])**.** *Let $B = \{0,1\}^b$. Let $h : \{0,1\}^c \times B \to \{0,1\}^c$ and $F : \{0,1\}^k \times D \to B$ be families of functions, and let $hF : \{0,1\}^{c+k} \times D \to \{0,1\}^c$ be defined by*

$$hF(K_{out}||K_{in}, M) = h(K_{out}, F(K_{in}, M))$$

for $K_{out} \in \{0,1\}^c, K_{in} \in \{0,1\}^k$ and $M \in D$. Let $\mathcal{A}_{hF}$ be a prf-adversary against hF that makes at most $q \geq 2$ oracle queries, each of length at most n, and has time complexity at most t. Then, there exists a prf-adversary $\mathcal{A}_h$ against h and a cAU-adversary $\mathcal{A}_F$ against F such that

$$\mathrm{Adv}^{\mathrm{prf}}_{hF}(\mathcal{A}_{hF}) \leq \mathrm{Adv}^{\mathrm{prf}}_{h}(\mathcal{A}_h) + \binom{q}{2}\mathrm{Adv}^{\mathrm{cAU}}_{F}(\mathcal{A}_F).$$

$\mathcal{A}_h$ makes at most q oracle queries, has time complexity at most t, and is obtained via a blackbox reduction. The two messages that $\mathcal{A}_F$ outputs have length at most n. The time complexity of $\mathcal{A}_F$ is t under a blackbox reduction and $O(T_F(n))$ under a non-blackbox reduction, where $T_F(n)$ is the time taken to compute F on an n-bit input.

The next theorem combines the previous two lemmas for a construction known as GNMAC (Generalized NMAC).

Theorem 3.3 (Theorem 3.3, [5])**.** *Assume $b \geq c$ and let $B = \{0,1\}^b$. Let $h : \{0,1\}^c \times B \to \{0,1\}^c$ be a family of functions and let* fpad $\in \{0,1\}^{b-c}$ *be a fixed padding string. Let $GNMAC\colon \{0,1\}^{2c} \times B^+ \to \{0,1\}^c$ be defined by*

$$GNMAC(K_{out}||K_{in}, M) = h(K_{out}, h^*(K_{in}, M)||\mathrm{fpad})$$

for all $K_{out}, K_{in} \in \{0,1\}^c$ and $M \in B^+$. Let $\mathcal{A}$ be a prf-adversary against GNMAC that makes at most $q \geq 2$ oracle queries, each of at most ℓ blocks, and has time complexity at most t. Then, there exist prf adversaries $\mathcal{A}_1, \mathcal{A}_2$ against h such that

$$\mathrm{Adv}^{\mathrm{prf}}_{GNMAC}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{prf}}_{h}(\mathcal{A}_1) + \binom{q}{2}\left[2\ell \cdot \mathrm{Adv}^{\mathrm{prf}}_{h}(\mathcal{A}_2) + \frac{1}{2^c}\right]. \tag{3.3}$$

$\mathcal{A}_1$ makes at most q oracle queries, has time complexity at most t, and is obtained via a blackbox reduction. $\mathcal{A}_2$ makes at most two oracle queries. The time complexity of $\mathcal{A}_2$ is t under a blackbox reduction and $O(\ell T_h)$ under a non-blackbox reduction, where T_h is the time for one computation of h.

There is a well known forgery attack on iterated MACs by Preneel and van Oorschot [28]. This attack is applied on MACs of the following form:

$$H_0 = IV, \quad H_{i+1} = h(H_i, x_i) \quad 0 \leq i \leq t, \quad output = g(H_t).$$

Here, h is the compression function with c-bit chaining value, $x_0, \ldots, x_t$ is the block representation of a suitably padded message, and g is the output transformation that converts the c-bit chaining value into an m-bit final MAC output. The secret key can be introduced in the IV, compression function h, or in the output transformation g. Note that GNMAC falls into this class of iterated MACs, with the keys being introduced in the IV and the function g.

The attack described in [28] is based on finding internal collisions in the iterated MAC. According to Proposition 3 and Corollary 2 in [28], an existential MAC forgery can be obtained using an expected number $2^{c/2+1}/\sqrt{s+1}$ of known message-tag pairs where each message has the same substring of s trailing blocks, and which works with probability approximately $1/(1+2^{c-m}/(s+1))$ provided that $s \geq 2^{c-m+6}$. For GNMAC, s can be at most the maximum block length ℓ, and can be assumed to be greater than 2^{c-m+6}. Therefore, the attack requires approximately $2^{c/2}/\sqrt{\ell}$ queries to the MAC function.

The non-blackbox version of Theorem 3.3 has content only up to $2^{c/2}/\ell$ queries for GNMAC, assuming only constructive adversaries for $\mathcal{A}_1$ and $\mathcal{A}_2$ (for a detailed analysis see §3.6). Since $\mathcal{A}_2$ in the non-blackbox version of Theorem 3.3 is non-constructive, strictly speaking, it must be evaluated with respect to non-constructive adversaries, and this gives a different query bound of $2^{c/4}/\sqrt{\ell}$ (see [24]). The blackbox version of Theorem 3.3 has content only up to $2^{c/3}/\ell^{1/3}$ queries for GNMAC (see [5]). In comparison, the best known forgery attack on GNMAC which is the one described above requires $2^{c/2}/\sqrt{\ell}$ queries to the MAC function. We see that there is a gap between the security offered (in terms of the number of queries upto which the theorem has content) by the blackbox version of Theorem 3.3 and the best known constructive attack. Even using the non-blackbox version of Theorem 3.3 does not help shorten this gap significantly.

3.3 Description of 2-Lane NMAC

L-Lane NMAC, a variant of NMAC, was introduced by Yasuda [33] for the following reason. From the attack on iterated MACs we saw in §3.2, the security

of NMAC/HMAC can be at most $O(2^{n/2})$ for an n-bit chaining variable. L-Lane NMAC was introduced to restore n-bit security for NMAC. The variable "L" in L-Lane NMAC represents an integer at least two; the L=2 and $L \geq 3$ cases are treated differently. We first describe the ingredients used in L-Lane NMAC before describing the construction of 2-Lane NMAC.

Doubly Injective Lengthening (DIL). Let $\mathcal{M}, X$ be finite subsets of $\{0,1\}^*$. Fix an integer $L \geq 2$. Let

$$\phi = \{\varphi_i\}_{1 \leq i \leq L}$$

be a family of functions $\varphi_i : \mathcal{M} \to X$. We say that ϕ is *doubly injective* if $M \neq M'$ $(M, M' \in \mathcal{M})$ implies $\varphi_i(M) \neq \varphi_i(M')$ and $\varphi_j(M) \neq \varphi_j(M')$ for some $1 \leq i < j \leq L$. Given a DIL $\phi = \{\varphi_i\}_{1 \leq i \leq L}$, define

$$\rho_\phi(\mu) = \max_{M,i} |\varphi_i(M)|,$$

where M ranges over messages in $\mathcal{M}$ of length at most μ.

DIL-cAU-PRF Construction. Given a **DIL** $\phi = \{\varphi_i\}_{1 \leq i \leq L}$ with $\varphi_i : \mathcal{M} \to X$, a cAU function $H : K \times X \to Y$, and a PRF $G : K' \times Y^L \to T$, we construct their compositions $H^L \circ \phi : K^L \times \mathcal{M} \to Y^L$ and $G \circ H^L \circ \phi : K^L \times K' \times \mathcal{M} \to T$ as described in Algorithm 3.4 and Algorithm 3.5 (see also Figure 3.1).

Algorithm 3.4 Function $\left(H^L \circ \phi\right)_{k_1,\ldots,k_L}(M)$

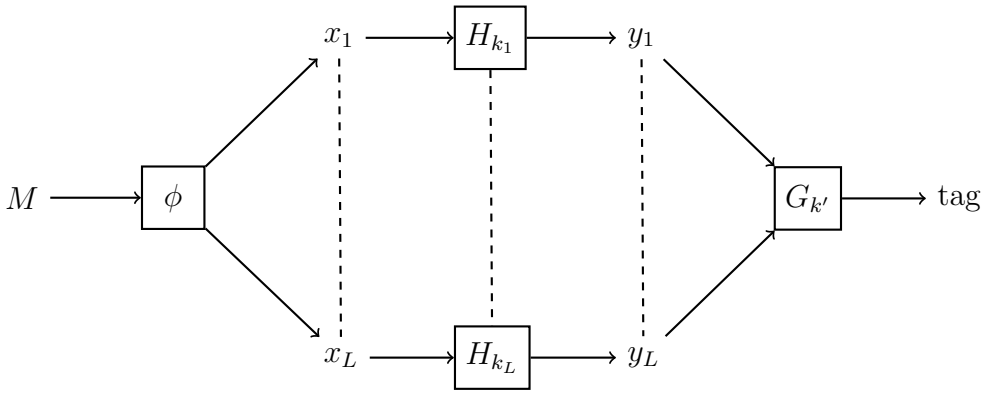
$x_i \leftarrow \varphi_i(M)$ for $i = 1, \ldots, L$
$y_i \leftarrow H_{k_i}(x_i)$ for $i = 1, \ldots, L$
Output $(y_1, \ldots, y_L)$

Algorithm 3.5 Function $(G \circ H^L \circ \phi)_{k_1,\ldots,k_L,k'}(M)$

$(y_1, \ldots, y_L) \leftarrow \left(H^L \circ \phi\right)_{k_1,\ldots,k_L}(M)$
$tag \leftarrow G_{k'}(y_1, \ldots, y_L)$
Output tag.

2-Lane NMAC. The 2-Lane NMAC is essentially a DIL-caU-PRF function. We describe the instantiation of each component of the DIL-cAU-PRF construction as seen in 2-Lane NMAC, next.

Figure 3.1: Yasuda's L-Lane NMAC construction.



1. Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ denote a compression function, where $m \geq 2n$.

2. Trivial instantiation of DIL ϕ. For the DIL φ_1, φ_2, the trivial instantiation $\varphi_1(M) = \varphi_2(M) = M$ where $M \in \mathcal{M}$ is used.

3. Merkle-Damgård iteration for the cAU H. For a compression function f, use the Merkle-Damgård iterated hash function as described in §3.1.2 to instantiate H.

4. Prf G. For $k' \in \{0,1\}^n$, $y_1, y_2 \in \{0,1\}^n$, the function G is defined as follows

$$G_{k'}(y_1, y_2) = f(k', y_1||y_2||0^{m-2n}).$$

   Note that this makes sense since we have assumed $m \geq 2n$.

3.4 Non-constructive proof

In this section, we reproduce Yasuda's proof of the prf property of 2-Lane NMAC [33]. This proof is in the non-constructive setting. All the results stated in this section were proved in [33] for the general case of L-Lane NMAC, where $L \geq 2$. To enable an easy analysis in the constructive setting, we consider the case $L = 2$.

Theorem 3.6. *An upper bound on the advantage of any (t, q, μ)-prf adversary of 2-Lane NMAC is given by*

$$\mathrm{Adv}^{\mathrm{prf}}_{\text{2-Lane-NMAC}}(t, q, \mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f}(t, q) + 2q^2 \cdot \left(\ell \cdot \mathrm{Adv}^{\mathrm{prf}}_{f}(t', 2) + 2^{-n-1}\right)^2$$

where $\ell = \lceil \mu/m \rceil$, $t' = 4\ell \cdot T_f$, $\mathrm{Adv}^{\mathrm{prf}}_{f}(t, q)$ is assessed against all constructive adversaries, and $\mathrm{Adv}^{\mathrm{prf}}_{f}(t', 2)$ is assessed against all non-constructive adversaries.

The proof of Theorem 3.6 follows by substituting the cAU advantage of the adversary in Theorem 3.7 with the right hand side of (3.2) in Lemma 3.1 where $n_1 = n_2 = \ell$.

Theorem 3.7. *An upper bound on the advantage of any (t, q, μ)-prf adversary of $G \circ H^2 \circ \phi$ is given by*

$$\mathrm{Adv}^{\mathrm{prf}}_{G \circ H^2 \circ \phi}(t, q, \mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{G}(t, q) + \binom{q}{2} \cdot \left(\mathrm{Adv}^{\mathrm{au}}_{H}(\mu')\right)^2,$$

where $q \geq 2$, $\mu' = \rho_\phi(\mu) = \mu$, and $\mathrm{Adv}^{\mathrm{prf}}_{G}(t, q)$ is assessed against constructive adversaries. (Note that in the instantiation for 2-lane NMAC, φ_1, φ_2 are identity maps, and hence $\mu' = \rho_\phi(\mu) = \max_{M,i} |\varphi_i(M)| = \mu$.)

Theorem 3.7 follows by applying the result of Lemma 3.8 to upper bound the cAU advantage term in Lemma 3.2. Bellare's Lemmas 3.1 and 3.2 each has a constructive as well as a non-constructive version. The main use of non-constructivity in the security result of 2-Lane NMAC stems from Lemma 3.8. There is no obvious way to convert it to the constructive setting.

Lemma 3.8 (Lemma 2 with L=2, [33])**.** *We have*

$$\mathrm{Adv}^{\mathrm{au}}_{H^2 \circ \phi}(\mu) \leq \left(\mathrm{Adv}^{\mathrm{au}}_{H}(\mu')\right)^2$$

where $\mu' = \rho_\phi(\mu) = \mu$.

Proof. Let $\mathcal{A}$ be a au-adversary attacking $H^2 \circ \phi$ with the maximum advantage, and that outputs a pair of messages each at most μ bits. Without loss of generality we assume that $\mathcal{A}$ always outputs a fixed pair of messages M, M' with $M, M' \in \mathcal{M}$ and $M \neq M'$.

Remark 3.9. Note that in the definition of AU used in [33], the time taken by the adversary is not taken into account. This gives an adversary unbounded computational power and makes the proof non-constructive. The strongest au-adversary for a function H is one that outputs two messages $M, M' \in \mathcal{M}$ that maximizes the number of keys $k \in \{0,1\}^n$ for which $H(k, M) = H(k, M')$. Note that there cannot exist an adversary with an advantage greater than the one we have described. Therefore, we can assume that an adversary attacking the AU property of H with maximum advantage will output a fixed pair of messages M, M' with probability 1.

Recall that the DIL functions φ_1, φ_2 described in the construction for 2-Lane NMAC are the identity functions. Therefore

$$\begin{aligned}\mathrm{Adv}^{\mathrm{au}}_{H^2 \circ \phi}(\mu) &= \Pr[H(k_1, \varphi_1(M)) = H(k_1, \varphi_1(M')) \wedge H(k_2, \varphi_2(M)) = H(k_2, \varphi_2(M')) \\ &\qquad |k_1, k_2 \leftarrow \{0,1\}^n], \\ &= \Pr[H(k_1, M) = H(k_1, M') \mid k_1 \leftarrow \{0,1\}^n] \\ &\qquad \cdot \Pr[H(k_2, M) = H(k_2, M') \mid k_2 \leftarrow \{0,1\}^n].\end{aligned}$$

Note that $\Pr[H(k_1, M) = H(k_1, M') \mid k_1 \leftarrow \{0,1\}^n] \leq \mathrm{Adv}^{\mathrm{au}}_H(\mu)$. The lemma follows from this. □

3.5 Constructive proof

In this section, we provide a proof of the prf property of 2-Lane NMAC in the constructive setting. This is obtained by converting the proof of each result in Section §3.4 to a constructive proof. We emphasize that all adversaries in this section are constructive, and that prf-advantages and cAU-advantages are assessed over constructive adversaries. Note that the result is similar to Theorem 3.3 obtained for GNMAC.

Theorem 3.10. *We have*

$$\mathrm{Adv}^{\mathrm{prf}}_{\text{2-Lane-NMAC}}(t, q, \mu) \leq \mathrm{Adv}^{\mathrm{prf}}_f(t, q) + q^2 \cdot (\ell \cdot \mathrm{Adv}^{\mathrm{prf}}_f(t, 2) + 2^{-n-1})$$

where $\ell = \lceil \mu/m \rceil$.

The proof of the above theorem follows from using the upper bound for the cAU advantage of H from Lemma 3.11 in Theorem 3.12. Lemma 3.11 is obtained from the constructive version of Lemma 3.1 using $n_1 = n_2 = \ell$.

Lemma 3.11. *If f is a PRF, then H, constructed from f via the Merkle-Damgård iteration, is cAU. More concretely, we have*

$$\mathrm{Adv}^{\mathrm{cAU}}_{H}(t,\mu) \leq (2\ell-1)\cdot \mathrm{Adv}^{\mathrm{prf}}_{f}(t,2) + \frac{1}{2^n}$$

where $\ell = \lceil \mu/m \rceil$.

Theorem 3.12. *We have*

$$\mathrm{Adv}^{\mathrm{prf}}_{G\circ H^2\circ\phi}(t,q,\mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{G}(t,q) + \binom{q}{2}\cdot \mathrm{Adv}^{\mathrm{cAU}}_{H}(t,\mu')$$

where $q \geq 2$ and $\mu' = \rho_\phi(\mu) = \mu$.

Theorem 3.12 is obtained by using the upper bound for the cAU-advantage of H^2 from Lemma 3.14 in Lemma 3.13. Lemma 3.13 follows from the constructive version of Bellare's Lemma 3.2.

Lemma 3.13. *We have*

$$\mathrm{Adv}^{\mathrm{prf}}_{G\circ H^2\circ\phi}(t,q,\mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{G}(t,q) + \binom{q}{2}\cdot \mathrm{Adv}^{\mathrm{cAU}}_{H^2\circ\phi}(t,\mu)$$

where $q \geq 2$.

In Lemma 3.14, we lose the square factor in the upper bound for $\mathrm{Adv}^{\mathrm{au}}_{H^2\circ\phi}$ compared to the corresponding Lemma 3.8 we had in the non-constructive setting[1]. We show that it is not possible to obtain the upper bound from Lemma 3.8 in the constructive setting, unless the adversary $\mathcal{A}^{\mathrm{cAU}}_{H^2\circ\phi}$ satisfies a very strong condition.

Next, we state Lemma 3.14 and provide a proof.

Lemma 3.14. *We have*

$$\mathrm{Adv}^{\mathrm{cAU}}_{H^2\circ\phi}(t,\mu) \leq \mathrm{Adv}^{\mathrm{cAU}}_{H}(t,\mu')$$

where $\mu' = \rho_\phi(\mu) = \mu$.

[1]Note that [33] was published in Indocrypt 2007, when the version of [5] had all its results in the non-constructive model. The earliest version of [5] on eprint which has the revised Lemmas in the constructive model appears on 9^{th} April 2014. Therefore, it is understandable that [33] would use non-constructivity in their proofs, in accordance with the lemmas in [5].

Proof. Let $\mathcal{B}$ be a (t, μ)-cAU adversary of $H^2 \circ \phi$ such that $\mathrm{Adv}^{\mathrm{cAU}}_{H^2 \circ \phi}(t, \mu) = \mathrm{Adv}^{\mathrm{cAU}}_{H^2 \circ \phi}(\mathcal{B})$. In the rest of the proof, M, M' will represent messages that are each at most μ bits long. We have

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{cAU}}_{H^2 \circ \phi}(\mathcal{B}) &= \sum_{M, M' \in \mathcal{M}} \Pr\left[H_{k_1}(M) = H_{k_1}(M') \wedge H_{k_2}(M) = H_{k_2}(M') \;\middle|\; k_1, k_2 \xleftarrow{\$} K \right] \\
&\qquad \cdot \Pr[\{M, M'\} \leftarrow \mathcal{B}] \\
&= \sum_{M, M' \in \mathcal{M}} \Pr\left[H_k(M) = H_k(M') \;\middle|\; k \xleftarrow{\$} K \right]^2 \cdot \Pr[\{M, M'\} \leftarrow \mathcal{B}] \\
&\leq \sum_{M, M' \in \mathcal{M}} \Pr\left[H_k(M) = H_k(M') \;\middle|\; k \xleftarrow{\$} K \right] \cdot \Pr[\{M, M'\} \leftarrow \mathcal{B}] \\
&= \mathrm{Adv}^{\mathrm{cAU}}_{H}(\mathcal{B}) \\
&\leq \mathrm{Adv}^{\mathrm{cAU}}_{H}(t, \mu).
\end{aligned}$$

This completes the proof. □

Continuing the discussion from the above proof, let us use $P(M, M')$ to denote $\Pr\left[H_k(M) = H_k(M') \;\middle|\; k \xleftarrow{\$} K \right]$. For any (t, μ)-cAU adversary $\mathcal{B}$ against $H^2 \circ \phi$, we can construct a (t, μ)-cAU adversary $\mathcal{A}$ against H that just runs $\mathcal{B}$ and outputs the two messages that $\mathcal{B}$ outputs. Then, the cAU advantage of $\mathcal{A}$ is given by

$$\sum_{M, M' \in \mathcal{M}} \Pr\left[H_k(M) = H_k(M') \;\middle|\; k \xleftarrow{\$} K \right] \cdot \Pr[\{M, M'\} \leftarrow \mathcal{B}] .$$

We also have the inequality

$$E_{M,M'}[P^2(M, M')] \geq E^2_{M,M'}[P(M, M')], \tag{3.4}$$

where the expectation is taken over the coins of $\mathcal{B}$ as it outputs the messages M, M'. Therefore

$$\mathrm{Adv}^{\mathrm{cAU}}_{H^2}(t, \mu) \geq \mathrm{Adv}^{\mathrm{cAU}}_{H^2}(\mathcal{B}) \geq \left(\mathrm{Adv}^{\mathrm{cAU}}_{H}(\mathcal{A})\right)^2 = \left(\mathrm{Adv}^{\mathrm{cAU}}_{H}(t, \mu)\right)^2 .$$

Equality holds if and only if equation 3.4 is tight. This happens if with probability 1 $\mathcal{B}$ outputs a pair of messages $(M, M') \in S$, where S is a set such that for any $(M_1, M_1'), (M_2, M_2') \in S$, we have $P(M_1, M_1') = P(M_2, M_2')$. Note that from the definition of the adversary $\mathcal{B}$, it has the maximum advantage among all adversaries with the same time and bound on message length. This seems like a strong condition

that is required to be satisfied by a constructive adversary for any hash function H and time t. If $t = 2\mu T_f$, then one cannot expect the adversary to obtain information about this set S and output the two messages within time t, given any hash function H.

3.6 Concrete security

To evaluate the security of 2-Lane NMAC, we consider the dominant term in the upper bound of the prf-advantage in Theorem 3.10 and Theorem 3.6, which are in the constructive and non-constructive setting respectively. We examine two claims made by Yasuda in [33], the first regarding the security of NMAC, and the second regarding the security of L-Lane NMAC, both in the non-constructive model. There appears to be an error in both cases, even assuming that in the non-constructive model the fastest prf-attack is brute force; we correct them here.

The first claim is that the security of NMAC is upper bounded by a term of order $O(\ell q^2/2^n)$, obtained from the non-blackbox version of Theorem 3.3. By observation, the dominating term on the right hand side of the inequality (3.3) is $\binom{q}{2} 2\ell \cdot \mathrm{Adv}_h^{\mathrm{prf}}(\mathcal{A}_2)$. One can assume that for a constructive adversary $\mathcal{A}_2$ against the prf-property of h that runs in time t and makes at most two queries to the oracle, the highest advantage is $(t/T_h)/2^n$. This is because for a good compression function h, the assumption is that the best $\mathcal{A}_2$ can do is the following brute force attack. It queries two messages M_1, M_2 to the oracle and stores the output X_1, X_2 respectively. It tries as many keys K as possible in time t and computes $h(K, M_1)$. If for any key K, $h(K, M_1) = X_1$, it computes $h(K, M_2)$, and if this equals X_2, the adversary outputs 1, else 0. It can be seen that the advantage in this case is $(t/T_h)/2^n$, where t/T_h is the number of keys tried, and with a probability of $1/2^n$ a tried key is correct. Note that even if the non-blackbox version of Theorem 3.3 is evaluated against all constructive adversaries $\mathcal{A}_2$ against h, the dominant term in the upper bound must be $O(\ell^2 q^2/2^n)$ by setting $t = O(\ell \cdot T_h)$ in the brute force attack described above.

The second claim is that L-Lane NMAC has $O(\ell^2 q^2/2^{2n})$ security. The following is the theorem on prf security of L-Lane NMAC. We don't discuss the L-Lane construction in detail in this chapter, however.

Theorem 3.15 ([33]). *The upper bound on advantage for any (t, q, μ)-prf adversary*

against L-Lane NMAC is given by

$$\mathrm{Adv}^{\mathrm{prf}}_{\textit{L-Lane-NMAC}}(t,q,\mu) \leq \mathrm{Adv}^{\mathrm{prf}}_{f}(t,q) + 4q^2\left(\lambda \cdot \mathrm{Adv}^{\mathrm{prf}}_{f}(t',2) + 2^{-n-1}\right)^2$$

where $\lambda = max\{\lceil \mu/m(L-1)\rceil, \lceil Ln/m\rceil\}$ *and* $t' = 4\cdot\lceil\mu/(m(L-1))\rceil \cdot T_f$. *(Note that for the construction to work, we need* $m \geq 2n$*, which is satisfied in practice).*

Assuming that the dominant term is $4q^2 \cdot \lambda^2 \cdot (\mathrm{Adv}^{\mathrm{prf}}_{f}(t',2))^2$, we see that it is of the order of $O(q^2\ell^4/2^{2n})$, and not $O(\ell^2q^2/2^{2n})$.

Now we can analyze the security of 2-Lane NMAC under different assumptions. In Table 3.1, q_1 denotes the number of queries up to which Theorem 3.6 for the 2-Lane NMAC (which is non-constructive) is meaningful, assuming only constructive attacks to evaluate the term $\mathrm{Adv}^{\mathrm{prf}}_{f}(t',2)$. The quantity q_2 is computed similar to q_1, but against all known non-constructive attacks when evaluating $\mathrm{Adv}^{\mathrm{prf}}_{f}(t',2)$. The number of queries up to which Theorem 3.10 for 2-Lane NMAC (which is constructive) is meaningful, assuming only constructive adversaries, is denoted by q_3. The quantity q_4 is the number of queries up to which the constructive version of Theorem 3.3 for the GNMAC construction is meaningful. The last two columns of Table 3.1 can be used to compare the actual security benefits offered by 2-Lane NMAC over GNMAC. We do not consider the computational cost of using 2-Lane/L-Lane NMAC over GNMAC. For this, one can refer to the original paper [33].

Table 3.1: Query bounds

Hash function.	n	μ	m	ℓ	q_1	q_2	q_3	q_4
SHA-1	160	2^{50}	512	2^{41}	2^{75}	2^{39}	2^{40}	2^{40}
SHA-224/SHA-256	256	2^{50}	512	2^{41}	2^{171}	2^{87}	2^{72}	2^{72}
SHA-384/SHA-512	512	2^{50}	1024	2^{40}	2^{429}	2^{216}	2^{158}	2^{158}

NIST recommends an upper bound of $2^{64} - 8\cdot 64$ bits for a message input to HMAC instantiated with SHA256. We make the assumption that a reasonable upper bound on the bitlength of a query to HMAC with SHA-256 is 2^{50}. To give an estimate on how large this value is, assuming that a fast implementation of SHA256 requires 1.6 cycles per byte (see [12]) and working with a 4 GHz processor, it would take 15 hours to print the SHA256 hash output of a 2^{50}-bit message. We fix the same upper bound on the bitlength of messages queried to HMAC instantiated with

SHA-1 and SHA-512.

Computing $\mathbf{q_1}$. Consider Theorem 3.6. For simplicity, we set $T_f = 1$. Set $\mathrm{Adv}_f^{\mathrm{prf}}(t, q)$ to be $t/2^n$ and $\mathrm{Adv}_f^{\mathrm{prf}}(t', 2)$ to be $4\ell/2^n$, assuming the brute force attack described above to be the fastest constructive attack. This gives

$$\mathrm{Adv}^{\mathrm{prf}}_{\text{2-Lane-NMAC}}(t, q, \mu) \leq \frac{t}{2^n} + 2^5 \cdot \left(\frac{q\ell^2}{2^n}\right)^2.$$

To compute q_1, we set the right hand side of the above expression to 1, and make the assumption that the first term in the above expression is negligibly small in comparison to the second term when $q = q_1$. For example, from Table 3.1, if SHA256 is used as the hash function, then $\ell = 2^{50}/512 = 2^{41}$. Assuming $t = 2^{128}$, we need $q_1 \gg 2^{106}$ for the second term to be dominant. To compute q_1, set

$$2^6 \cdot \left(\frac{q_1\ell^2}{2^n}\right)^2 = 1$$

and thus

$$q_1 = \frac{2^{n-3}}{\ell^2}.$$

Computing $\mathbf{q_2}$. To compute q_2, assume that the fastest non-constructive attack against the compression function f is the one described by Koblitz and Menezes [24], which has advantage $1/2^{n/2}$, much larger than the brute force attack considered earlier. Substituting this in Theorem 3.6, we have

$$\mathrm{Adv}^{\mathrm{prf}}_{\text{2-Lane-NMAC}}(t, q, \mu) \leq \frac{t}{2^n} + 2 \cdot \left(\frac{q\ell}{2^{n/2}}\right)^2.$$

We make the assumption that the first term is negligibly small compared to the second term in the above expression when $q = q_2$. Setting $2 \cdot (q_2 l/2^{n/2})^2 = 1$ we have

$$q_2 = \frac{2^{n/2}}{\ell}.$$

Computing $\mathbf{q_3}$. To compute q_3, we use Theorem 3.10. Assuming that brute force is the fastest prf attack, we get

$$\mathrm{Adv}^{\mathrm{prf}}_{\text{2-Lane-NMAC}}(t, q, \mu) \leq q^2 \ell t / 2^n.$$

Assuming that each query to the oracle takes one time step, the prf-adversary against 2-Lane-NMAC can make at most t queries. Therefore, $q \leq t$. This gives

$$q_3 = \left(\frac{2^n}{\ell}\right)^{1/3}.$$

Computing q_4. Since the constructive theorem for 2-Lane NMAC is the same as for GNMAC, we have the same upper bound on the number of queries q_4 for GNMAC.

One can see from Table 3.1 that the actual security offered by Yasuda's non-constructive theorem, i.e. Theorem 3.6 (column q_2) is much closer to the security offered by its constructive version, i.e. Theorem 3.10 (column q_3) (which is really the same as the security offered by GNMAC (q_4)), as opposed to what is claimed in the paper (q_1).

Chapter 4

A Round-Optimal Blind Signature Scheme

In this chapter, we take a closer look at the efficiency claims made in a *round-optimal blind signature scheme* by Garg and Gupta [17]. We examine uses of non-constructivity in the security proofs, and recalculate security parameters and efficiency by evaluating the underlying hard problems against the fastest known non-constructive attacks, wherever applicable. This results in significantly larger security parameters and lowered efficiencies than originally claimed.

The security proof of the Garg-Gupta scheme is in the Common Reference String (CRS) model, which refers to a setup where every party in a cryptographic protocol has access to a *common reference string* generated by a trusted party. This is typically used to eliminate interactions between parties, for example, to create a non-interactive proof of knowledge system.

The chapter is organized as follows: In §4.1, we define blind signature schemes and state the two main security properties, i.e. blindness and unforgeability. We describe the framework of the Garg-Gupta round-optimal blind signature scheme in §4.2. In §4.3 we describe the hardness assumptions used, and the fastest known constructive and non-constructive attacks against them. In §4.4, we highlight the use of complexity leveraging and non-constructivity used in the security results for unforgeability and blindness, respectively. In §4.5 we instantiate the Garg-Gupta scheme as in [17], and in §4.6 we compute its security parameters and efficiency for the 80-bit and 128-bit security levels.

4.1 Blind signature schemes

To describe a blind signature scheme simply, it involves two parties, a user and a signer, where the user wants to get a signature s of some message m from the signer, who has a secret key sk and a corresponding public key vk. The user does not want to reveal m to the signer, but needs a signature that verifies under vk; this property is called *blindness*. Further, the user must be unable to forge a signature s' on any message m' such that s' verifies correctly under vk (provided it has not obtained the signature for m' from the signer previously) even if the user is allowed to ask for signatures of a certain number of messages from the signer previously. This property is referred to as *unforgeability*. We formally define a blind signature scheme, and state the two security properties of blindness and unforgeability.

A blind signature scheme can be described in terms of three algorithms which are all probabilistic polynomial time. However, for the blind signature scheme under consideration in this chapter, we will assume that the third algorithm, $\mathsf{SignVer}$ is deterministic. In the following, λ denotes a security parameter, and $\mathcal{U}, \mathcal{S}$ represent the user and signer respectively, which are interactive algorithms. It is assumed that if either of $\mathcal{U}, \mathcal{S}$ aborts, it outputs the null string $\perp$.

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$. sk is the secret key known only to $\mathcal{S}$, and vk is the public key.
- $\mathsf{SignGen}(\mathsf{sk}, m) \to \sigma$. m represents the message which has to be signed, and σ is its signature. $\mathsf{SignGen}$ is an interactive protocol between $\mathcal{U}$ and $\mathcal{S}$, where $\mathcal{U}$ has input m, vk and outputs either σ or aborts. $\mathcal{S}$ has input sk and either generates no output, or aborts.
- $\mathsf{SignVer}(\mathsf{vk}, m, \sigma) \to \{0, 1\}$. If the output is 1, it means that the signature σ verified correctly for the message m, and 0 means that the verification failed.

4.1.1 Completeness

For perfect completeness, the following should hold true:

$$\Pr\left[\mathsf{SignVer}(\mathsf{vk}, m, \sigma) = 1 \;\middle|\; \mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk});\ \mathsf{SignGen}(\mathsf{sk}, m) \to \sigma\right] = 1$$

for all messages m.

4.1.2 Blindness

The signer is assumed to be adversarial, and tries to learn something about the messages that the user wants to get signed. Further, since the signer is also responsible for generating the keys in a blind signature scheme, we assume that it can generate keys adversarially. Let $\mathcal{A}$ be a PPT adversary that can operate in three modes: *generate*, *protocol* and *guess*. We can describe the game played between the adversary and the user in three steps.

1. In the first step, $\mathcal{A}$ operating in the "generate" mode takes as input a security parameter λ and outputs two messages m_0, m_1, a public key vk, a secret key sk, and a string st_1.

2. In the second step, the user with public key vk chooses a bit b at random and executes the blind signature protocol to sign the messages $m_b, m_{\bar{b}}$ in that order, with $\mathcal{A}$ operating in the "protocol" mode acting as the signer. $\mathcal{A}$'s initial input at the start of this step is st_1, sk. If in any of the signing protocols for either of the messages m_0, m_1 the user aborts, the user outputs $\perp$ for the other message too, even if it generates a signature for it successfully without aborting. Let the user's output at the end of the first execution of the signing protocol be denoted as σ_0, and the second one as σ_1. Thus, the user's output at the end of this step is (σ_0, σ_1), where either both of σ_0, σ_1 are $\perp$, or neither is $\perp$ and each is a successfully generated signature. The adversary outputs a string st_2.

3. In the third step, the adversary, operating in the "guess" mode, takes as input st_2, (σ_0, σ_1) and outputs a bit b'.

The advantage of $\mathcal{A}$ is defined to be

$$\mathrm{Adv}^{\mathrm{Blindness}}_{\mathcal{A},BS}(\lambda) = |\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]|.$$

If the advantage is negligibly small in λ for all probabilistic polynomial time adversaries $\mathcal{A}$, then the scheme is said to satisfy the blindness property.

4.1.3 Unforgeability

In this case, the user acts as the adversary $\mathcal{A}$. The game is as follows.

1. $\mathsf{KeyGen}(1^\lambda)$ is run to generate $(\mathsf{sk}, \mathsf{vk})$, where vk is public and only the signer $\mathcal{S}$ is given access to sk.

2. $\mathcal{A}$ interacts with $\mathcal{S}(\mathsf{sk})$ in at most k executions of the signing protocol. If $\mathcal{A}$ outputs distinct messages $m_1, \ldots, m_k, m_{k+1}$ and their corresponding signatures $\sigma_1, \ldots, \sigma_{k+1}$ such that $\mathsf{SignVer}(\mathsf{vk}, m_i, \sigma_i) = 1$ for all $1 \leq i \leq k+1$, then $\mathcal{A}$ is said to succeed.

The advantage of $\mathcal{A}$ in the unforgeability game, denoted by $\mathrm{Adv}^{\mathrm{unforge}}_{\mathcal{A},BS}(\lambda)$, is defined as the probability that $\mathcal{A}$ succeeds in the above game. For a fixed value of k, the blind signature scheme is said to be unforgeable if the advantage is negligible in the security parameter λ for all probabilistic polynomial time adversaries $\mathcal{A}$.

4.2 The Garg-Gupta blind signature scheme

We give the framework of the Garg-Gupta blind signature scheme [17] next. It uses the following cryptographic components.

1. A group based commitment scheme $\mathsf{Com}_{\mathbb{G}}$. This is a commitment scheme where the message is an element in a group $\mathbb{G}$ and the commitment is comprised of group elements in $\mathbb{G}$. The scheme must be computationally hiding, by which we mean it is computationally hard for any adversary to distinguish between commitments of two messages, even if it is allowed to choose the messages. It must also be perfectly binding, by which we mean that a commitment must open to a unique message.

2. A structure preserving signature scheme on groups. It consists of three algorithms: Key generation $\mathsf{SPKeyGen}(1^\lambda) \rightarrow (\mathsf{sk}_{SP}, \mathsf{vk}_{SP})$, where sk_{SP} is the secret key and vk_{SP} is the public key; Signing $\mathsf{SPSign}(\mathsf{sk}_{SP}, m) \rightarrow s$; and signature verification $\mathsf{SPVer}(\mathsf{vk}_{SP}, m, s) \rightarrow \{0, 1\}$. Public keys, messages and signatures are group elements, and verification involves evaluating pairing product equations.

3. A 2-CRS NIZK (*2-common reference string non-interactive zero knowledge proof*) on groups. Let R be an efficiently computable binary relation, and let $(x, w) \in R$. We shall refer to x as the statement and w as the corresponding witness. The 2-CRS NIZK consists of the following components.

    - A PPT crs generator $K(1^\lambda) \rightarrow (\mathsf{crs}, \tau)$, where crs represents a common reference string and τ is called the extraction key.

- Two PPT deterministic algorithms $\mathsf{Shift}, \mathsf{Shift}^{-1}$ that take a crs as input, and output a "shifted" crs.
- A PPT prover $\mathcal{P}$. Let $K(1^\lambda) \to (\mathsf{crs}, \tau)$. $\mathcal{P}$ takes as input one of $\{\mathsf{crs}, \mathsf{Shift}(\mathsf{crs}), \mathsf{Shift}^{-1}(\mathsf{crs})\}$ denoted by crs', a proof statement x, and a witness w, and outputs a proof π.
- A PPT verifier $\mathcal{V}$. Let $K(1^\lambda) \to (\mathsf{crs}, \tau)$. $\mathcal{V}$ takes as input one of $\{\mathsf{crs}, \mathsf{Shift}(\mathsf{crs}), \mathsf{Shift}^{-1}(\mathsf{crs})\}$ denoted by crs', a proof statement x, and a proof π. The verifier $\mathcal{V}(\mathsf{crs}', x, \pi)$ outputs 1 or 0, which corresponds to the case where it accepts or rejects the proof π, respectively.
- A PPT extractor ε that takes as input crs, the extraction key τ, a statement x, and a proof π. It produces a witness $w \leftarrow \varepsilon(\mathsf{crs}, \tau, x, \pi)$ such that $(x, w) \in R$.

The 2-CRS NIZK must satisfy the following properties

- **Perfect completeness.** If a prover produces a proof honestly, then the verifier must accept it. That is,

$$\Pr[\mathcal{V}(\mathsf{crs}, x, \pi) = 1 \mid \mathcal{P}(\mathsf{crs}, x, w) \to \pi] = 1,$$

  where crs can be the output of either $K, \mathsf{Shift} \circ K$ or $\mathsf{Shift}^{-1} \circ K$.
- **Perfect soundness.** For a crs $\mathsf{crs} \leftarrow K(1^\lambda)$, a statement x, and a proof π such that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$, there must exist a witness w such that $(x, w) \in R$. Perfect knowledge extraction implies perfect soundness.
- **Perfect knowledge extraction.** There must exist an efficient knowledge extractor ε that can extract the witness from a zero-knowledge proof that verifies correctly under a crs generated from K. In other words, for any statement x, $crs \leftarrow K(1^\lambda)$, and proof π such that $\mathcal{V}(\mathsf{crs}, x, \pi) = 1$, we require

$$\Pr[(x, \varepsilon(\mathsf{crs}, \tau, x, \pi)) \in R] = 1.$$

- **Perfect zero-knowledge.** It must be computationally hard to extract any information about the witness used in a proof, from the proof. We do not describe this property formally here since it is not required for our discussion. It is enough to note that this property must hold under the shifted crs's, i.e. $\mathsf{Shift}(\mathsf{crs})$ and $\mathsf{Shift}^{-1}(\mathsf{crs})$ where $\mathsf{crs} \leftarrow K(1^\lambda)$.

- **CRS-indistinguishability.** Informally, this property requires that the following two distributions be computationally indistinguishable : $(\mathsf{crs}', \mathsf{crs})$ and $(\mathsf{crs}, \mathsf{crs}'')$, where $\mathsf{crs} \leftarrow K(1^\lambda)$, $\mathsf{crs}' = \mathsf{Shift}^{-1}(\mathsf{crs})$ and $\mathsf{crs}'' = \mathsf{Shift}(\mathsf{crs})$.

This completes the description of the cryptographic components. We describe the framework of the Garg-Gupta scheme next.

1. Key generation:
   - Let λ be a security parameter. Execute a pairing generation algorithm $\mathcal{G}(1^\lambda)$ to obtain $(n, \mathbb{G}, g, \mathbb{G}_T, e)$ where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are groups of prime order n, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a symmetric bilinear pairing. Select $0 < q < n$. All algorithms described in this protocol take $(n, \mathbb{G}, g, \mathbb{G}_T, e)$ as implicit input.
   - Execute $\mathsf{SPKeyGen}(1^\lambda) \to (\mathsf{sk}_{SP}, \mathsf{vk}_{SP})$.
   - Sample $(\mathsf{crs}_1, \tau) \leftarrow K(1^\lambda)$, and its shift $\mathsf{crs}_2 \leftarrow \mathsf{Shift}(\mathsf{crs}_1)$.
   - Set $\mathsf{vk} = (\mathsf{vk}_{SP}, \mathsf{crs}_1, \mathsf{crs}_2, q)$ and $\mathsf{sk} = (\mathsf{sk}_{SP}, \tau)$.
2. Signing protocol: The user $\mathcal{U}$ wants to get a message $m \in \mathbb{G}$ signed by the signer $\mathcal{S}$.
   (a) Round 1 (User):
      - Abort if $\mathsf{Shift}(\mathsf{crs}_1) \neq \mathsf{crs}_2$.
      - $\mathsf{Com}_{\mathbb{G}}(m; r) \to m_{blind}$.
      - Sample a uniformly random c such that $0 < c < q$, and set $C = g^c$.
      - Generate a proof $\pi \leftarrow \mathcal{P}(\mathsf{crs}_1, x, c)$ of the statement x:

        $$\text{There exists } c,\ 0 < c < q,\ \text{such that } g^c = C.$$

      - Send (m_{blind}, C, π) to $\mathcal{S}$.

   (b) Round 2 (Signer):
      - Verify that $\mathcal{V}(\mathsf{crs}_1, x, \pi) = 1$, else abort.
      - Extract $c = \varepsilon(\mathsf{crs}_1, \tau, x, \pi)$.
      - Sample $\pi' \leftarrow \mathcal{P}(\mathsf{crs}_2, x, c)$.
      - Generate $\sigma_{SP} \leftarrow \mathsf{SPSign}(\mathsf{sk}_{SP}, m_{blind})$.
      - Send (σ_{SP}, π') to $\mathcal{U}$.

(c) Round 3 (User):

- Check if $\mathcal{V}(\mathsf{crs}_2, x, \pi') = 1$ and if $\mathsf{SPVer}(\mathsf{vk}_{SP}, m_{blind}, \sigma_{SP}) = 1$. If either is false then abort.
- Let Φ be the following statement:

$$\exists(m_{blind}, r, \sigma_{SP}) | m_{blind} = \mathsf{Com}_{\mathbb{G}}(m; r) \wedge \mathsf{SPVer}(\mathsf{vk}_{SP}, m_{blind}, \sigma_{SP}) = 1.$$

$\mathcal{U}$ outputs the signature on m as σ, where σ is a proof of Φ under crs_2.

3. Verification: To verify a signed message (m, σ), check if $\mathcal{V}(\mathsf{crs}_2, \Phi, \sigma) = 1$.

4.3 Hardness assumptions

In this section, we describe various hardness assumptions that need to be examined for the blind signature scheme. To evaluate the assumptions, we consider constructive as well as non-constructive attacks for each problem. First, we describe a suitable choice of an elliptic curve to instantiate the symmetric bilinear pairing.

4.3.1 Symmetric bilinear pairings

Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be groups of prime order n. A symmetric bilinear pairing

$$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$$

is a non-degenerate bilinear pairing. Such a pairing is cryptographically useful if e and the group operations of $\mathbb{G}, \mathbb{G}_T$ are efficiently computable, and the discrete logarithm problem in $\mathbb{G}$ is intractable. Since the pairing e can be used to efficiently reduce the discrete logarithm problem in $\mathbb{G}$ to the discrete logarithm problem in $\mathbb{G}_T$, we also require that the latter problem be intractable. The known constructions for cryptographically useful symmetric bilinear pairings arise from supersingular elliptic curves over finite fields.

Let $\mathbb{F}_q$ be a finite field of characteristic p, and let E be a supersingular elliptic curve defined over $\mathbb{F}_q$ (so $p \mid (q + 1 - \#E(\mathbb{F}_q))$. Let n be a prime divisor of $\#E(\mathbb{F}_q)$ with $n^2 \nmid \#E(\mathbb{F}_q)$ and $\gcd(n, q) = 1$, and let $\mathbb{G}$ be the order-n subgroup of $E(\mathbb{F}_q)$. The embedding degree of E (with respect to $\mathbb{G}$) is the smallest positive integer k

such that $n \mid (q^k - 1)$. The Weil and Tate pairings can be used to construct a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}_T$ is the order-n subgroup of the multiplicative group $\mathbb{F}^*_{q^k}$ of the extension field $\mathbb{F}_{q^k}$.

Supersingular elliptic curves having embedding degrees 2, 4 and 6, corresponding to the cases $q = p$, $q = 2^m$ and $q = 3^m$, respectively, have been widely implemented. However, after the discovery of quasi-polynomial time algorithms for computing discrete logarithms problems in finite fields of small characteristic (see [4], [18], [2]), the elliptic curves of embedding degrees 4 and 6 are considered insecure for cryptographic applications. Thus, in the remainder of this chapter we will instantiate symmetric bilinear pairings using supersingular elliptic curves of embedding degree 2.

The following is one construction for these curves (see [21]). Select a prime n, and a prime $p = nh - 1$ where $4 \mid h$ and $3 \nmid h$. Consider the elliptic curve

$$E/\mathbb{F}_p : Y^2 = X^3 - 3X.$$

Then $\#E(F_p) = p+1 = nh$, and so E is a supersingular elliptic curve with embedding degree $k = 2$. The Weil and Tate pairings yield a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ is the order-n subgroup of $E(\mathbb{F}_p)$ and $\mathbb{G}_T$ is the order-n subgroup of $\mathbb{F}^*_{p^2}$. Observe that an element in $\mathbb{G}$ is a point in $E(\mathbb{F}_p)$, and so can be represented using two elements of $\mathbb{F}_p$, or a single element of $\mathbb{F}_p$ (plus a single bit) if point compression is used. Note also that the discrete logarithm problem in $\mathbb{G}$ can be solved using Pollard's rho algorithm in time approximately $n^{1/2}$ [27], or by using the number field sieve (NFS) to solve the discrete logarithm problem in $\mathbb{F}^*_{p^2}$ in time $L_{p^2}[\frac{1}{3}, 1.923]$ [3].

4.3.2 Discrete logarithm in a symmetric bilinear group

Definition 4.1 (Discrete logarithm in a symmetric bilinear group assumption)**.** *Let $\mathbb{G} = \langle g \rangle$ be a symmetric bilinear group of prime order n and security parameter λ. For a PPT algorithm $\mathcal{A}$ and the group $\mathbb{G}$, the following probability should be negligible in the security parameter λ:*

$$\mathrm{Adv}^{\mathrm{dlog}}_{\mathbb{G},\mathcal{A}}(1^\lambda) = \Pr\left[c = c' \;\middle|\; c \xleftarrow{\$} \mathbb{Z}_n;\ C = g^c;\ c' \leftarrow \mathcal{A}(C)\right].$$

Note that $\mathbb{G}$ and g are known to the attacker before it sees C. In the non-constructive setting, the attacker can perform unbounded precomputation using g, and this may offer faster run times as compared to attacks in the constructive setting.

First, we assess the fastest known constructive attacks for discrete logarithms in $\mathbb{G}$. As noted in §4.3.1, the discrete logarithm problem in $\mathbb{G}$ can be solved using Pollard's rho algorithm in time approximately $n^{1/2}$ and minimal space requirements [27], or by using the number field sieve (NFS) to solve the discrete logarithm problem in $\mathbb{F}_{p^2}^*$ in time $L_{p^2}[\frac{1}{3}, 1.923]$. For the former attack to be infeasible, n should be at least 160 bits for the 80-bit security level, and 256 bits for the 128-bit security level. For the latter attack to be infeasible, p should be at least 512 bits for the 80-bit security level, and 1536 bits for the 128-bit security level.

Next, we consider two non-constructive attacks on the discrete logarithm problem in $\mathbb{G}$. First, we mention an attack due to Bernstein and Lange [7] on the discrete logarithm problem over the NIST P-256 elliptic curve. Let P be a generator of the NIST P-256 curve, an elliptic curve group of a 256-bit prime order n over a 256-bit prime field $\mathbb{F}_p$. The discrete logarithm problem is the following: Given a point Q on the curve, find the unique integer k modulo n such that $Q = kP$. Bernstein and Lange's attack requires approximately $2n^{1/3}$ curve additions, and succeeds with a considerable probability of at least 0.23. The space requirements are $n^{1/3}$ "distinguished" elliptic curve points, but the precomputation to find them is of the order of $n^{2/3}$ curve additions. The Bernstein-Lange attack is not restricted to the P-256 curve, which was only used for concreteness and practical relevance. It can also be applied to compute discrete logarithms in the group $\mathbb{G}$ with similar computational and space requirements.

The second non-constructive attack is the speedup for NFS with precomputation on $\mathbb{F}_p$ due to Commeine et al. [10]. We make the heuristic assumption that these speedups apply for discrete logarithms in the finite field $\mathbb{F}_{p^2}$ as well. Therefore, the running time of this attack on $\mathbb{F}_{p^2}$ is $L_{p^2}[1/3, 3^{1/3}]$, with a precomputation table of size $L_{p^2}[1/3, .951]$.

4.3.3 Discrete logarithm in an interval

Definition 4.2 (Discrete logarithm in an interval (DLOG-q) assumption)**.** *Let $\mathbb{G} = \langle g \rangle$ be a symmetric bilinear group of prime order n and security parameter λ, and let $0 < q < n$. The* DLOG-q *assumption is that for all PPT adversaries $\mathcal{A}$, the advantage*

$$\mathrm{Adv}^{\mathrm{dlog}}_{\mathcal{A},\mathbb{G},q}(\lambda) = \Pr\left[c = c' \;\middle|\; c \xleftarrow{\$} \mathbb{Z}_q;\ C = g^c;\ c' \leftarrow \mathcal{A}(C)\right]$$

is negligible in the security parameter λ.

To determine the fastest known constructive attack on DLOG-q in $\mathbb{G}$, we consider two attacks. The first one is the "kangaroo method" due to Pollard [27]. The kangaroo method has expected running time is about $2n^{1/2}$ group operations (ignoring optimizations in the constant factor), and requires minimal storage. The second attack is the number field sieve for computing discrete logarithms in $\mathbb{F}_{p^2}$, and has running time $L_{p^2}[1/3, 1.923]$.

We consider the following two non-constructive attacks against discrete logarithm in an interval in $\mathbb{G}$: computing discrete logarithms in $\mathbb{F}_{p^2}$ using the number field sieve with precomputation having running time $L_{p^2}[1/3, 3^{1/3}]$ (see §4.3.2), and Pollard's kangaroo method with precomputation, requiring approximately $n^{1/3}$ elliptic curve additions and a storage of $n^{1/3}$ precomputed "distinguished points". The precomputation effort is of the order of $n^{2/3}$ curve additions. We describe the latter attack in detail next.

Procedure. Let $S = \{s_1, \ldots, s_r\}$ be a set of *jump distances*, and $J = \{g^{s_1}, \ldots, g^{s_r}\}$ be the set of *jumps*. We require the mean of the set S to be $\alpha = q^{2/3}$. A walk starting at the point $x_0 \in \mathbb{G}$ is defined as a sequence of points x_i, $i \geq 0$, such that $x_{i+1} = x_i g^{s_{d(x_i)}}$ where $d : \mathbb{G} \to \{1, \ldots, r\}$. A step is defined to be the process of moving from x_i to x_{i+1} for some $i \geq 0$. The function d should be chosen such that the sizes of the preimages of i, for each $1 \leq i \leq r$, are roughly equal, and the function is efficiently computable. An analysis of the choice of parameters for Pollard's Kangaroo method is presented in [31]. Based on these considerations, we choose $r = 20$, and the jump distances in S to be randomly chosen between 1 and 2α with $\gcd(s_1, \ldots, s_r) = 1$. The jumps can be precomputed and stored. We define the "distance" between two group elements, g^x and g^y, as $|x - y|$ assuming that $0 \leq x, y < n$.

Define a distinguishing criterion such that an element in the group is distinguished with probability $1/(\beta q^{1/3})$ where β is a constant (we shall choose an appropriate value of β later on). This is typically done using an easily-verifiable property of the representation of a group element, such as checking if some x bits of the binary representation of the group element are all zeroes, and marking it as distinguished if true. In the above example, a fraction of $1/2^x$ points are distinguished. The purpose of using distinguishing points is to offer a time-memory trade-off.

In the *precomputation* phase, start $m = q^{1/3}$ walks (tame kangaroos) at the points

$g^{q/2+iv}$, $0 \leq i \leq m-1$, and keep track of the discrete logarithm of the current position of each walk. Here, v is a constant chosen to be α/m. Stop a walk if it hits a distinguished point. Note that the distance between the "first" and "last" walk is $(m-1) \cdot v$, which is approximately $q^{2/3}$. If any of the walks reaches a distinguished point before $\beta q^{1/3}$ steps, restart the walk from a new point a small distance away from the previous and repeat the process. It is unfavourable for our case if two of these walks collide at the same distinguished point (one can think of this situation as one where no "new" information is provided by the colliding walk). Therefore, repeat one of the walks starting from a point slightly displaced from its original starting point, until a different distinguished point is obtained. Store the m distinguished points, and their discrete logarithms in a table.

In the *online* phase, after receiving $C = g^c$, launch one wild kangaroo from C and perform a walk for at most $q/2\alpha + (\beta+1)q^{1/3}$ steps, recording only the distance between C and the current point in the walk. Record the point reached after the $(q/2\alpha)^{th}$ step (call it x), and the discrete logarithm of $C^{-1}x$. At each step, check if a distinguished point is obtained, and if it coincides with a distinguished point in the precomputed table. Suppose that the wild kangaroo hits a distinguished point Cg^b which is also stored in the table as (Cg^b, a); then $C = g^{a-b}$ and $(a-b) \bmod n$ is the discrete logarithm of C. If the walk fails to reach a distinguished point in the table, restart the wild kangaroo at a small distance from the $(q/2\alpha)^{th}$ step.

Analysis. At the start of the online phase, the wild kangaroo can be anywhere between g^0 and g^{q-1}. Once it covers $q/2\alpha$ steps, assuming an average distance of α for each step, it is somewhere betwen $g^{q/2}$ and $g^{3q/2}$. The starting points of the precomputed walks are between $g^{q/2}$ and $g^{q/2+q^{2/3}}$. Therefore, once the wild kangaroo covers $q/2\alpha$ steps, we can assume that it is in the region of the precomputed walks. The probability that in one step in this region, the wild kangaroo collides with one of the m walks in the precomputation is $m/\alpha = q^{1/3}/q^{2/3} = 1/q^{1/3}$. The probability that there is at least one collision in the next $q^{1/3}$ steps of the wild kangaroo with any of the precomputed walks is roughly

$$1 - \left(1 - \frac{1}{q^{1/3}}\right)^{q^{1/3}} \approx 1 - e^{-1}.$$

If there is a collision between a precomputed walk and a wild kangaroo, both will hit the same distinguished point, which can be detected. We have to ensure that after the wild kangaroo has completed $q/2\alpha + q^{1/3}$ steps, it is still in the region of precomputed walks for the above mentioned collision to occur. For this, we need

each precomputed walk to travel at least $q^{1/3}$ steps before it hits a distinguished point. Note that after the wild kangaroo has taken $q/2\alpha$ steps, it is at most $q/2\alpha$ steps from g^q, and hence at most q/α steps from $g^{q/2}$, the starting point of the "first" precomputed walk corresponding to $i = 0$. Therefore, we need each precomputed walk to cover at least $q/\alpha + q^{1/3}$ steps before a distinguished point is reached. We need β to satisfy the relation $\beta q^{1/3} = q/\alpha + q^{1/3} = 2q^{1/3}$, whence $\beta = 2$. Once a collision between the wild kangaroo and a precomputed walk has ocurred, the wild kangaroo needs to travel at most $\beta q^{1/3} = 2q^{1/3}$ steps to hit a distinguished point, which should match with some entry in the table. Therefore, the total running time of the wild kangaroo in terms of steps is $q/2\alpha + q^{1/3} + \beta q^{1/3}$, where the collision occurs with probability $1 - e^{-1}$. The expected running time is then

$$q^{1/3}\left(\frac{1}{2} + 3\frac{1}{1-e^{-1}}\right) \approx 5q^{1/3}.$$

4.3.4 Decision linear problem

Definition 4.3 (Decision Linear Assumption)**.** *Let $\mathbb{G} = \langle g \rangle$ be a bilinear group of prime order n with security parameter λ. For a non-uniform PPT adversary $\mathcal{A}$, its advantage* $\mathrm{Adv}^{\mathrm{DLIN}}_{\mathcal{A},\mathbb{G}}(\lambda)$ *is defined as:*

$$\mathrm{Adv}^{\mathrm{DLIN}}_{\mathcal{A},\mathbb{G}}(\lambda) = \mathsf{Pr}\left[b = b' \;\middle|\; \begin{array}{l} g_1, g_2 \xleftarrow{\$} \mathbb{G}; x, y, z \xleftarrow{\$} \mathbb{Z}_n \\ b \xleftarrow{\$} \{0,1\};\ \textit{if } b\textit{=0, set } W = g^{x+y} \textit{ else set } W = g^z \\ b' \leftarrow \mathcal{A}(g_1, g_2, g_1^x, g_2^y, W) \end{array}\right] - 1.$$

The DLIN assumption is said to hold for the bilinear group $\mathbb{G}$ if for all PPT non-uniform algorithms $\mathcal{A}$, $\mathrm{Adv}^{\mathrm{DLIN}}_{\mathcal{A},\mathbb{G}}(\lambda)$ *is negligible in λ. More generally we say that the T-DLIN assumption holds in the group $\mathbb{G}$ if for every $T \cdot poly(\lambda)$ time non-uniform algorithm $\mathcal{A}$,* $\mathrm{Adv}^{\mathrm{DLIN}}_{\mathcal{A},\mathbb{G}}(\lambda)$ *is negligible in λ.*

It can be shown that if one can solve the DLIN problem in $\mathbb{G}$, then one can solve the decisional Diffie-Hellman problem (DDH) in $\mathbb{G}$, implying that DLIN is at least as hard as DDH. Boneh et al. [8] proved that a generic algorithm that solves the DLIN problem has a lower bound complexity of $O(n^{1/2})$ for a generic bilinear group of order n. The fastest known attack to solve DLIN in $\mathbb{G}$ is to solve discrete logs in $\mathbb{G}$. This is done by computing discrete logs of g_1, g_1^x, g_2 with base g, thereby computing x, Wg^{-x}, and determining if $(Wg^{-x})^{\log_g g_2} = g_2^y$. Therefore, the fastest known non-constructive attack for the DLIN problem is the fastest non-constructive

attack for computing discrete logs in $\mathbb{G}$. Any generic algorithm to solve the latter problem has a lower bound of $n^{1/3}$, as was shown recently in [11].

4.4 Security proofs

In this section, we will highlight the uses of non-constructivity and complexity leveraging used in the two proofs of blindness and unforgeability, respectively.

4.4.1 Proof of blindness

Theorem 4.4 (Theorem 3, [17]). *For any PPT malicious signer $\mathcal{S}^*$ that plays the blindness game of §4.1.2, the following holds:*

$$\mathrm{Adv}^{\mathrm{Blindness}}_{\mathcal{S}^*,BS}(\lambda) < 2 \cdot \mathrm{Adv}^{\mathrm{hide}}_{\mathcal{A},\mathsf{Com}_{\mathbb{G}}}(\lambda) + \mathrm{Adv}^{\mathrm{dlog}}_{\mathcal{B},\mathbb{G},q}(\lambda)$$

where $\mathcal{A}$ is an adversary against the non-uniform hiding property of $\mathsf{Com}_{\mathbb{G}}$ such that $\mathbf{T}(\mathcal{A}) = \mathbf{T}(\mathcal{S}^*) + poly(\lambda)$, and $\mathcal{B}$ is an adversary against the non-uniform discrete log problem in $\mathbb{G}$ when exponents are chosen uniformly randomly in $\mathbb{Z}_q$ such that $\mathbf{T}(\mathcal{B}) = \mathbf{T}(\mathcal{S}^*) + poly(\lambda)$.

The use of non-uniformity in Theorem 4.4 stems from the proof of Lemma 4.5. We describe the proof of Theorem 4.4 leading up to Lemma 4.5, providing only enough details to understand the setting in which Lemma 4.5 is relevant. This proof uses a sequence of games, starting from the original blindness game Game_0 between the challenger $\mathcal{U}$ (honest user) and the adversary $\mathcal{S}^*$ (malicious signer). For our purpose, we will need to consider what is denoted as Game_1 in [17] and is described below.

Game_1. $\mathcal{S}^*$ outputs a public key vk and challenge messages m_0, m_1. The challenger may run in unbounded time to test if crs_2 is in the range of K. If the challenger concludes that crs_2 is in the range of K, it can compute and store an extraction key τ corresponding to crs_2. $\mathcal{S}^*$ expects the incoming blind messages $m_{blind,b}, m_{blind,1-b}$ from $\mathcal{U}$ corresponding to the messages m_b, m_{1-b}, where b is a random bit. After receiving the two messages, $\mathcal{S}^*$ outputs its responses to the challenger. The challenger $\mathcal{U}$ outputs the signatures on m_0, m_1 obtained from the two executions of the signing protocol with the adversary $\mathcal{S}^*$. If the following two events occur (i) crs_2 is in the range of K and (ii) the first instance of the signing protocol completes successfully, the challenger $\mathcal{U}$ additionally outputs DL-Abort. The adversary $\mathcal{S}^*$ then outputs a

bit b^*. If $b = b^*$, then the adversary is said to succeed, and $\mathrm{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda)$ is defined as $|2\Pr[b = b^*] - 1|$.

Game_0 and Game_1 are identical, except when the challenger outputs DL-Abort. Thus, they are identical-until-DL-Abort-is-set, and

$$\begin{aligned} &|\Pr[\mathcal{A} \text{ succeeds in } \mathsf{Game}_0] - \Pr[\mathcal{A} \text{ succeeds in } \mathsf{Game}_1]| \\ &\qquad \leq \Pr[\mathcal{U} \text{ outputs DL-Abort in } \mathsf{Game}_1]. \end{aligned} \tag{4.1}$$

The expression on the left hand side of (4.1) is $1/2 \cdot |\mathrm{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) - \mathrm{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda)|$. Note that $\mathrm{Adv}^{\mathsf{Game}_0}_{\mathcal{A}}(\lambda) = \mathrm{Adv}^{\mathrm{Blindness}}_{\mathcal{A},BS}(\lambda)$. Therefore, we have

$$\mathrm{Adv}^{\mathrm{Blindness}}_{\mathcal{A},BS}(\lambda) \leq \mathrm{Adv}^{\mathsf{Game}_1}_{\mathcal{A}}(\lambda) + \Pr[\mathcal{U} \text{ outputs DL-Abort in } \mathsf{Game}_1].$$

We are concerned with the probability that $\mathcal{U}$ outputs DL-Abort. This is given by the following lemma.

Lemma 4.5 (Lemma 1, [17])**.** *The probability of DL-Abort happening is bounded above by* $\mathrm{Adv}^{\mathrm{dlog}}_{\mathcal{B},\mathbb{G},q}(\lambda)$ *with* $T(\mathcal{B}) = T(\mathcal{S}^*) + poly(\lambda)$*, where* $\mathcal{B}$ *is an adversary against the non-uniform discrete log problem in* $\mathbb{G}$ *when exponents are chosen uniformly at random from* $\mathbb{Z}_q$*.*

The proof of Lemma 4.5 uses the following argument. Using an adversary $\mathcal{S}^*$ that makes the challenger output DL-Abort in Game_1, one can "construct" a non-uniform adversary $\mathcal{B}$ that solves the discrete logarithm in an interval problem in $\mathbb{G}$, where the exponent is restricted to q. The adversary $\mathcal{B}$ in the proof of Lemma 4.5 uses unbounded precomputation, and a polynomial sized advice string. We explain the use of non-constructivity in $\mathcal{B}$ next.

The adversary $\mathcal{B}$ participates in Game_1 as the challenger against $\mathcal{S}^*$. For an adversary $\mathcal{S}^*$ there exist coins c such that, upon running $\mathcal{S}^*$ with these coins, the challenger in Game_1 outputs DL-Abort with the maximum probability. Let $\mathcal{B}$ be a non-uniform adversary, which for an adversary $\mathcal{S}^*$, obtains the aforementioned coins c as an advice string and hard codes $\mathcal{S}^*$ with these coins. Further, $\mathcal{B}$ upon obtaining the public key vk from $\mathcal{S}^*$, executes in unbounded time to test if crs_2 is in the range of K or not. Therefore, the adversary $\mathcal{B}$ is non-uniform and non-constructive.

4.4.2 Proof of unforgeability

Suppose that $\mathcal{A}$ is an adversary for problem A, and $\mathcal{B}$ is an adversary for problem B constructed from $\mathcal{A}$ using a reduction from B to A. Complexity leveraging is

the technique where the running time of $\mathcal{B}$ so constructed is much larger than the running time of $\mathcal{A}$. Since B must now be hard for adversaries that have much larger running times, this technique is often not desirable in a security proof, and authors try to minimize or eliminate the need for complexity leveraging. The Garg-Gupta protocol is more efficient compared to previous constructions of round-optimal blind signature schemes in the standard model due to its reduced use of complexity leveraging. We describe the use of complexity leveraging in [17], which appears in the proof of unforgeability.

Theorem 4.6 (Theorem 1, [17])**.** *For any PPT malicious user $\mathcal{U}^*$ for the unforgeability game against the blind signature scheme, the following holds*

$$\mathrm{Adv}^{\text{unforge}}_{\mathcal{U}^*,BS}(\lambda) \leq \mathrm{Adv}^{\text{crs-distinguish}}_{\mathcal{B}}(\lambda) + \mathrm{Adv}^{\text{unforge}}_{\widehat{\mathcal{U}^*},SPSign}(\lambda) \tag{4.2}$$

where $\mathcal{B}$ is an adversary against the CRS indistinguishability property of the 2-CRS NIZK proof system with run time $T(\mathcal{B}) = k \cdot T^{\text{dlog}}_{\mathbb{G},q} + T(\mathcal{U}^*) + poly(\lambda)$ and $\widehat{\mathcal{U}^*}$ is an adversary against the unforgeability of the underlying structure preserving signature scheme such that $T(\widehat{\mathcal{U}^*}) = k \cdot T^{\text{dlog}}_{\mathbb{G},q} + T(\mathcal{U}^*) + poly(\lambda)$. Also, $\mathcal{U}^*$ and $\widehat{\mathcal{U}^*}$ make at most k signing queries.

We now describe the use of complexity leveraging in the proof of Theorem 4.6.

Consider the unforgeability game for the blind signature scheme described in §4.1.3, and denote it as Game_0. The signer $\mathcal{S}$ is the challenger, and the user $\mathcal{U}^*$ is the malicious adversary. We can modify Game_0 to obtain Game_1, where the challenger extracts the witness c of the statement "There exists c, $0 < c < q$, such that $g^c = C$" from the proof π by computing the discrete logarithm of C instead of using the extractor ε. If this value is greater than q, it aborts. The challenger requires time $T^{\text{dlog}}_{\mathbb{G},q}$ for this step. Since the 2-CRS NIZK proof π used with crs_1 is perfectly sound, the value c computed by the challenger must equal the value otherwise obtained using the knowledge extractor ε. Thus, $\mathcal{U}^*$ sees no difference between Game_0 and Game_1.

We can modify Game_1 to obtain Game_2, where the challenger generates crs_1 and crs_2 in the opposite way, i.e. $\mathsf{crs}_2 \leftarrow K(1^\lambda)$, and $\mathsf{crs}_1 = \mathsf{Shift}^{-1}(\mathsf{crs}_2)$. From the crs-indistinguishability property of the 2-CRS NIZK, it is hard to distinguish between $(\mathsf{crs}_1, \mathsf{crs}_2)$ and $(\mathsf{crs}'_1, \mathsf{crs}'_2)$, where $\mathsf{crs}'_1 \leftarrow K(1^\lambda)$ and $\mathsf{crs}'_2 = \mathsf{Shift}(\mathsf{crs}'_1)$ (the way crs's are generated in the blind signature scheme). Any adversary $\mathcal{U}^*$ that distinguishes between Game_1 and Game_2 can be used to construct an adversary $\mathcal{B}$ that runs in time at least $k \cdot T^{\text{dlog}}_{\mathbb{G},q} + T(\mathcal{U}^*)$, and that distinguishes between the two crs pairs mentioned

above. ($\mathcal{B}$ does this by running as the challenger in the distinguishing game between Game_1 and Game_2, and setting $(\mathsf{crs}_1, \mathsf{crs}_2)$ as the challenge pair that it receives in the crs-distinguishing game.) This is one use of complexity leveraging in the proof of unforgeability, as the adversary $\mathcal{B}$ in the reduction must solve DLOG-q multiple times (at most the query bound k of $\mathcal{U}^*$).

Next, it is shown that any adversary $\mathcal{U}^*$ that succeeds in Game_2 can be used to construct an adversary $\widehat{\mathcal{U}^*}$ that outputs an existential-forgery for the underlying structure preserving signature scheme. $\widehat{\mathcal{U}^*}$ does this by running as the challenger in Game_2. Since the challenger in Game_2 must solve at most k instances of DLOG-q, the running time of $\widehat{\mathcal{U}^*}$ must be $k \cdot T^{\text{dlog}}_{\mathbb{G},q} + T(\mathcal{U}^*)$ in addition to the time taken for the remainder of the reduction.

4.5 Instantiating the Garg-Gupta scheme

In this section, we will describe the instantiations used in [17] for each component of the framework of the GG scheme provided in §4.2.

- A **symmetric bilinear pairing** $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are groups of order n. This is instantiated using a suitable elliptic curve $E(\mathbb{F}_p)$ and the Weil/Tate pairing, as described in §4.3.1.

- **Commitment scheme on groups**. The commitment scheme $\mathsf{Com}_{\mathbb{G}} : \mathbb{G} \times \mathbb{Z}_n^4 \to \mathbb{G}^6$ is defined as

$$\mathsf{Com}_{\mathbb{G}}(m; a, b, x, y) = (g, g^a, g^b, g^{ax}, g^{by}, m \cdot g^{x+y}).$$

  To generate a commitment on a message m, choose $a, b, x, y \xleftarrow{\$} \mathbb{Z}_n^4$ and output $\mathsf{Com}_{\mathbb{G}}(m; a, b, x, y)$. To reveal the commitment, output (m, a, b, x, y). The scheme is computationally hiding under the DLIN assumption, and perfectly binding.

- **Structure preserving signature scheme**. A constant size, structure preserving signature scheme SIG1 instantiated under the DLIN assumption, by Chase et al. [1] is used. The scheme employs the symmetric bilinear pairing e. If the message contains k elements of $\mathbb{G}$, the public key consists of $2k + 25$ elements of $\mathbb{G}$, the signature is comprised of 17 elements of $\mathbb{G}$, and verification involves evaluating 9 pairing product equations.

- **2-CRS NIZK proofs on groups**. This is instantiated using Groth-Sahai (GS) NIZKs [19]. GS NIZKs are used to provide proofs of knowledge for certain kinds of equations. Before describing the type of statements that allow a GS NIZK, we describe its ingredients.

  – A crs which is sampled from one of two distributions, depending on whether the proof is in the *binding* mode or *hiding* mode. If the crs is binding, then there is also an extraction key τ, and a PPT knowledge extractor ε. In particular, if the crs in the GS setup is a DLIN tuple, it is in the binding mode, and in this mode, GS satisfies perfect knowledge extraction. If the crs in the GS setup is a non-DLIN tuple, wherein $\mathsf{crs} = (g, g^a, g^b, g^{ax}, g^{by}, g^z)$ with $z \neq x+y$, then the GS setup is in the hiding mode. In this mode, the perfect zero knowledge property is satisfied. Both modes support perfect completeness.

  – A prover $\mathcal{P}$ and a verifier $\mathcal{V}$ which are PPT, and operate in both the modes just described.

  A 2-CRS NIZK can be constructed from a GS NIZK in the following way

  – For the prover, verifier, and knowledge extractor of the 2-CRS NIZK, use the corresponding algorithms available for GS NIZK.

  – Set the crs generator to output a DLIN tuple:

  $$K(1^\lambda) \to \mathsf{crs} = (g, g^a, g^b, g^{ax}, g^{by}, g^{x+y}).$$

  The extraction key τ is (a, b, x, y). Note that under this crs, the GS NIZK elements $\mathcal{P}, \mathcal{V}, \varepsilon$ operate in the binding mode.

  – Set

  $$\mathsf{Shift}(\mathsf{crs}) \to (g, g_1, g_2, g_1^x, g_2^y, g^{x+y+1})$$

  and

  $$\mathsf{Shift}^{-1}(\mathsf{crs}) \to (g, g_1, g_2, g_1^x, g_2^y, g^{x+y-1}).$$

  Note that both the shifted crs's are non-DLIN tuples, and hence the GS NIZK operates in the hiding mode.

  A GS NIZK is used to provide proof of knowledge of statements that are *pairing product equations*, *multiscalar multiplication equations* and *quadratic equations*. However, we also need to provide a proof of knowledge of a different type of equation called a *range equation*, in the protocol.

1. **Pairing product equations**. A pairing product equation (PPE) is of the following form, where $X_1, \ldots, X_m \in \mathbb{G}$ are variables, $A_i \in \mathbb{G}$ and $\gamma_{1,j} \in \mathbb{Z}_n$ are constants:

$$\prod_{i=1}^{m} e(A_i, X_i) \prod_{i=1, j \geq i}^{m,m} e(X_i, X_j)^{\gamma_{i,j}} = 1.$$

2. **Multiscalar multiplication equations**. Let $X_1, \ldots, X_m \in \mathbb{G}$ and $y_1, \ldots, y_m \in \{0, 1\}$ be variables. The equation must be of the form

$$\prod_{i=1}^{m} A_i^{y_i} \prod_{i=1}^{t} X_i^{b_i} \prod_{i=1}^{t} \prod_{j=1}^{m} X_i^{y_j \gamma_{i,j}} = \tau,$$

   where $A_i \in \mathbb{G}$, $b_i, \gamma_{i,j} \in \mathbb{Z}_p$ and $\tau \in \mathbb{G}$ are constants.

3. **Quadratic equations over $\mathbb{Z}_n$**. Let $y_1, \ldots, y_m$ be variables in $\mathbb{Z}_n$. Let $a_i, \gamma_{i,j}$ and t, shown in the equation below, be constants in $\mathbb{Z}_n$. The quadratic equation must be of the following form

$$\sum_{i=1}^{m} a_i y_i + \sum_{1 \leq i \neq j \leq m} \gamma_{i,j} y_i y_j = t.$$

4. **Range equation**. Let $c \in \mathbb{Z}_n$ and $C \in \mathbb{G}$ be variables. Let $q \in \mathbb{Z}_n$ be a constant. The equation must be of the form

$$0 < c < q \ \wedge \ g^c = C.$$

   For any $0 < c < q$, let $d = \lfloor \log_2 q \rfloor + 1$, and let $x_0, \ldots, x_{d-1}$ be the bit representation of c. We can write the range equation in the following way:

$$\prod_{i=0}^{d-1} g_i^{x_i} = C,$$
$$x_i(1 - x_i) = 0, \text{ where } 0 \leq i \leq d - 1,$$

   where $x_0, \ldots, x_{d-1}$ are variables in $\mathbb{Z}_n$, C and $g_i = g^{2^i}$ are constants in $\mathbb{G}$. This is a combination of a multiscalar multiplication equation and quadratic equations, for which it is known how to construct GS-NIZK.

From Table 4 in [19], a GS-NIZK proof of a statement containing m variables, each either in $\mathbb{Z}_n$ or $\mathbb{G}$, ℓ pairing product equations/multiscalar multiplication equations, is comprised of $3m + 9\ell$ group elements of $\mathbb{G}$.

4.6 Concrete security

In this section, we will discuss some concerns with the selection of security parameters for the Garg-Gupta scheme as performed in [17]. The parameters are computed using Theorems 4.4 and 4.6 on blindness and unforgeability respectively, and efficiency is measured in terms of public key size, signature size, and communication size. We recalculate parameters and efficiency for the 80-bit and 128-bit security levels, taking into consideration the aforementioned concerns.

The security parameters of a scheme are selected using a reductionist security proof in the following way. A typical reduction uses an attacker against the scheme that takes time t and succeeds with probability ϵ, to construct an attacker for the underlying hard problem A that takes time t' and succeeds with probability ϵ'. If the hypothesis of the security theorem is that A is (t', ϵ')-hard, then it implies that the scheme is (t, ϵ)-secure. Parameters are then chosen for the scheme assuming that the hypothesis is true. Therefore, it becomes necessary to assess the hardness of the underlying problem A. If the reduction is non-constructive, i.e., there exists (in the mathematical sense) an attacker for A but there is no efficient way known to construct it, then the hypothesis must hold against all non-constructive adversaries, making it a stronger assumption. Non-constructive attacks against commonly used hard problems in cryptography are not as well studied as constructive attacks, making it difficult to base our confidence on such strong hypotheses. Indeed, in practice for any cryptographic problem one is mostly concerned with attacks that can actually be constructed, as opposed to fast attacks that *exist* but cannot be found efficiently.

There has been some interest recently in studying non-constructive attacks and their implications to provable security including the non-contructive attack on pseudorandom functions by Koblitz and Menezes [24], non-constructive attacks on AES-128, NIST P-256, DSA-3072 and RSA-3072 by Bernstein and Lange [7], and recent work by Corrigan-Gibbs and Kogan [11] on lower bounds for generic discrete logarithm attacks with precomputation. It was shown in [11] that any generic algorithm for the discrete logarithm problem in a group of order N that allows unbounded preprocessing, uses an S-bit advice string, runs in online time T, and succeeds with probability ϵ, must satisfy $ST^2 = \widetilde{\Omega}(\epsilon N)$. The Bernstein-Lange attack with $T = N^{1/3}$ and $S = N^{1/3}$ matches this lower bound. A faster attack in any group would require one to exploit the group representation.

Returning to the Garg-Gupta scheme, we have two main objections to their analysis

of security parameters. The first is that the authors assume the hypothesis of Corollary 4.7 holds in the constructive setting, whereas the reduction in their proof makes use of non-constructive adversaries. They evaluate the underlying hard problems, DLIN and DLOG-q, with respect to constructive attacks only. As seen in §4.3, there exist much faster non-constructive attacks for the two problems. The result is that they assume that their theorem is stronger than it actually is, and obtain smaller security parameters.

The second issue is their treatment of the order-n group $\mathbb{G}$ as a generic group. Since it is a symmetric bilinear group, it is implemented as a subgroup of a suitably chosen elliptic curve $E/\mathbb{F}_p$. Therefore, there are three parameters under consideration for this scheme: n, q and p. The choice of p was not considered in [17].

A group element in $\mathbb{G}$ is an elliptic curve point, represented by an element of $\mathbb{F}_p$ with one additional bit (see §4.3.1). If $n \ll p$, then storing a point requires significantly more than $\log n$ bits as was assumed in [17]. For a comparison, from line 1 of Table 4.1 where q, n and p are 240, 380 and 3270-bit integers respectively, storing one element of $\mathbb{G}$ requires 3271 bits as opposed to 380 bits. This has an impact on all three measures of efficiency, since they all depend on the size of a group element.

Further, when evaluating the hardness of discrete logarithm and related assumptions in $\mathbb{G}$, the authors of [17] do not consider the existence of a faster discrete logarithm attack on $\mathbb{G}$ that utilizes pairings to reduce the problem to computing discrete logarithms in $\mathbb{F}_{p^2}$, for which there exist much faster subexponential attacks. They assume that the fastest attack against discrete logarithm and DLIN in $\mathbb{G}$ is Pollard's Rho, and the fastest attack for DLOG-q takes time $q^{1/2}$. The pairing-based attack has to be taken into account when considering attacks against discrete logarithm, DLOG-q and DLIN, since all three can be solved by computing discrete logarithms in $\mathbb{F}_{p^2}$. This further enlarges the security parameters than what was originally claimed. Compounding the two issues is the fact that when considering attacks against DLIN and DLOG-q in the non-constructive setting, we also need to consider the $L_p[1/3, 3^{1/3}]$-NFS with precomputation by Commeine et al. [10] apart from generic group attacks with precomputation.

Next, we compute the security parameters for the 80-bit and 128-bit security levels. Table 4.1 presents the security parameters and public key sizes that we computed, whereas Table 4.2 presents the corresponding values computed using the methodology in [17].

Table 4.1: Security parameters (q, n, p), public key size (vk), signature size (σ), and communication complexity (CC) of the Garg-Gupta blind signature scheme for different values of k and t, with the underlying hard problems evaluated with respect to the appropriate computational model of the adversary.

	k	t	q	p	n	vk (in KB)	σ (in KB)	CC (in MB)
80-bit	2^{20}	2^{30}	240	3270	380	17.6	74.8	1.8
	2^{20}	2^{40}	240	3700	400	19.9	84.6	2
	2^{30}	2^{30}	240	4170	420	22.4	95.4	2.3
	2^{30}	2^{40}	240	4670	440	25.1	106.8	2.5
128-bit	2^{20}	2^{30}	384	12322	652	66.2	281.9	10.7
	2^{20}	2^{40}	384	13286	672	71.4	303.9	11.6
	2^{30}	2^{30}	384	14294	692	76.8	326.9	12.4
	2^{30}	2^{40}	384	15348	712	82.5	351.1	13.3

4.6.1 Selecting parameters

In order to measure the computational effort of an adversary $\mathcal{A}$ in attacking a problem with running time $T(\mathcal{A})$ and advantage ϵ, we use the concept of a work factor (as was done in [17]) which is defined as $T(\mathcal{A})/\epsilon$. First, we compute parameters for the 80-bit security level, followed by parameters for 128-bit security.

Since the hiding property of the commitment scheme $\mathsf{Com}_{\mathbb{G}}$ is based on the DLIN assumption, we have the following corollary of Theorem 4.4.

Corollary 4.7 (Corollary 2, [17])**.** *For any PPT malicious signer $\mathcal{S}^*$ that plays the blindness game of §4.1.2, the following holds:*

$$\mathrm{Adv}^{\mathrm{Blindness}}_{\mathcal{S}^*,BS}(\lambda) < 4 \cdot \mathrm{Adv}^{\mathrm{DLIN}}_{\mathcal{C},\mathbb{G}}(\lambda) + \mathrm{Adv}^{\mathrm{dlog}}_{\mathcal{B},\mathbb{G},q}(\lambda),$$

where $\mathcal{C}$ is an adversary against the non-uniform DLIN assumption in $\mathbb{G}$ such that $\mathbf{T}(\mathcal{C}) = \mathbf{T}(\mathcal{S}^*) + poly(\lambda)$, and $\mathcal{B}$ is an adversary against the non-uniform discrete log problem in $\mathbb{G}$ when exponents are chosen uniformly randomly in $\mathbb{Z}_q$ such that $\mathbf{T}(\mathcal{B}) = \mathbf{T}(\mathcal{S}^*) + poly(\lambda)$.

For the 80-bit security level, an adversary $\mathcal{S}^*$ playing the blindness game should have $\mathrm{Adv}^{\mathrm{Blindness}}_{\mathcal{S}^*,BS}(\lambda) = 1$ only when $T(\mathcal{S}^*) \geq 2^{80}$. Suppose that there exists such an

Table 4.2: Security parameters (q, n), public key size (vk), signature size (σ), and communication complexity (CC) of the Garg-Gupta blind signature scheme for different values of k and t, evaluated in the original work [17].

	k	t	q_{GG}	n_{GG}	vk_{GG}(in KB)	σ_{GG}(in KB)	CC_{GG}(in KB)
80-bit	2^{20}	2^{30}	155	291	1.6	6.6	102.9
	2^{20}	2^{40}	155	311	1.7	7.1	110
	2^{30}	2^{30}	155	331	1.8	7.8	117.1
	2^{30}	2^{40}	155	351	1.9	8.0	124.4
128-bit	2^{20}	2^{30}	248	385	2.1	8.8	216.8
	2^{20}	2^{40}	248	405	2.2	9.3	228.1
	2^{30}	2^{30}	248	424	2.3	9.7	238.8
	2^{30}	2^{40}	248	444	2.4	10.2	250

adversary $\mathcal{S}^*$ that runs in time 2^{80} and succeeds in the blindness game with advantage 1. Corollary 4.7 tells us that there exists an adversary $\mathcal{B}$ that solves DLOG-q with probability 1 and has running time $T(\mathcal{B}) \approx 2^{80}$. (This is assuming that $\mathrm{Adv}^{\mathrm{DLIN}}_{\mathcal{C},\mathbb{G}}(\lambda) \ll \mathrm{Adv}^{\mathrm{dlog}}_{\mathcal{B},\mathbb{G},q}(\lambda)$, given that $\mathcal{C}$ and $\mathcal{B}$ take approximately the same time.) Corollary 4.7 has some content only if the adversary $\mathcal{B}$ constructed through the reductionist security proof performs better than the fastest known non-constructive attack against DLOG-q, for otherwise we cannot use the result to conjecture that there exist no attacks against the blind signature scheme that take time at most 2^{80} and succeed with probability 1. To assess the fastest known non-constructive attack for DLOG-q in $\mathbb{G}$, we consider three attacks: (i) Pollard's kangaroo method with precomputation in $\mathbb{G}$ with running time $q^{1/3}$ (see §4.3.3); (ii) Bernstein-Lange's Pollard's Rho method with precomputation in $\mathbb{G}$ with running time $n^{1/3}$; and (iii) non-constructive NFS for discrete logarithms in $\mathbb{F}_{p^2}$ with running time $L_{p^2}[1/3, 3^{1/3}]$ [10]. We need the following inequalities to hold:

$$T(\mathcal{B})/\mathrm{Adv}^{\mathrm{dlog}}_{\mathcal{B},\mathbb{G},q}(\lambda) = 2^{80} \leq q^{1/3}, \tag{4.3}$$

$$2^{80} \leq n^{1/3}, \tag{4.4}$$

$$2^{80} \leq L_{p^2}[1/3, 3^{1/3}], \tag{4.5}$$

which imply that q must be at least 240 bits, n must be at least 240 bits, and p must be at least bits 825 bits for the 80-bit security level.

We now consider the theorem on unforgeability for the blind signature scheme. We have the following existential unforgeability result for the constant size, structure preserving signature scheme (SIG1) by Chase et al. [1] mentioned in §4.5.

Theorem 4.8 (Theorem 30, [1])**.** *SIG1 is a structure-preserving signature scheme that yields constant-size signatures, and is UF-CMA (existentially UnForgeable under Chosen Message Attack) under the DLIN assumption. In particular, for any PPT adversary $\mathcal{A}$ for SIG1 making at most q_s signing queries, there exists a PPT algorithm $\mathcal{B}$ such that*

$$\mathrm{Adv}^{\mathrm{uf-cma}}_{SIG1,\mathcal{A}}(\lambda) \leq (q_s + 3)\mathrm{Adv}^{\mathrm{DLIN}}_{\mathbb{G},\mathcal{B}}(\lambda) + \frac{1}{n}. \tag{4.6}$$

The CRS-distinguishing property holds under the DLIN assumption. Using this fact, and substituting (4.6) in (4.2), we have the following corollary.

Corollary 4.9. *For any PPT malicious user $\mathcal{U}^*$ playing the unforgeability game described in §4.1.3, the following holds*

$$\mathrm{Adv}^{\mathrm{unforge}}_{\mathcal{U}^*,BS}(\lambda) \leq (k + 5)\mathrm{Adv}^{\mathrm{DLIN}}_{\mathbb{G},\mathcal{A}}(\lambda) + \frac{1}{n}$$

where $\mathcal{A}$ is an adversary against the DLIN assumption in $\mathbb{G}$ with run time $T(\mathcal{A}) = k \cdot T^{\mathrm{dlog}}_{\mathbb{G},q} + T(\mathcal{U}^*) + poly(\lambda)$, and k is the maximum number of signing queries made by $\mathcal{U}^*$.

For the 80-bit security level, we require that for any adversary $\mathcal{U}^*$ playing the unforgeability game, $\mathrm{Adv}^{\mathrm{unforge}}_{\mathcal{U}^*,BS}(\lambda) = 1$ only if $T(\mathcal{U}^*)$ is at at least 2^{80}. Suppose that there exists an adversary $\mathcal{U}^*$ such that $T(\mathcal{U}^*) = 2^{80}$ and $\mathrm{Adv}^{\mathrm{unforge}}_{\mathcal{U}^*,BS}(\lambda) = 1$, i.e., it breaks the unforgeability property of the blind signature scheme. Corollary 4.9 tells us that there exists an adversary $\mathcal{A}$ against the DLIN problem in $\mathbb{G}$ that takes time $k \cdot T^{\mathrm{dlog}}_{\mathbb{G},q} + T(\mathcal{U}^*)$ and has advantage at least $1/(k + 5)$.

Remark 4.10. The following simplifying assumption is made regarding the running time of $\mathcal{A}$ in [17]. It is assumed that the running time of $\mathcal{A}$ is $t \cdot k \cdot T^{\mathrm{dlog}}_{\mathbb{G},q}$, for a suitably chosen constant t. The fastest known constructive attack to solve DLOG-q in $\mathbb{G}$ is to either use Pollard's Kangaroo with running time $q^{1/2}$, or compute discrete logs in $\mathbb{F}_{p^2}$ in time $L_{p^2}[1/3, 1.923]$. Therefore, $T^{\mathrm{dlog}}_{\mathbb{G},q}$ is the minimum of $L_{p^2}[1/3, 1.923]$ and $q^{1/2}$, and it follows from equations (4.4) and (4.5) that each of these is much greater than $T(\mathcal{U}^*) = 2^{80}$. Hence $k \cdot T^{\mathrm{dlog}}_{\mathbb{G},q} \gg 2^{80}$, and the simplifying assumption can be made.

In order for Corollary 4.9 to have any content, we need $T(\mathcal{A})/\mathrm{Adv}^{\mathrm{DLIN}}_{\mathbb{G},\mathcal{A}}(\lambda)$ to be smaller than the work factor of the fastest known attack on the DLIN problem in $\mathbb{G}$. We consider two attacks, computing discrete logarithms in $\mathbb{G}$ using NFS in $\mathbb{F}_{p^2}$, and computing discrete logarithms in $\mathbb{G}$ using Pollard's Rho. This yields

$$\frac{t \cdot k \cdot T^{\mathrm{dlog}}_{\mathbb{G},q}}{1/(k+5)} \leq L_{p^2}[1/3, 1.923] \tag{4.7}$$

and

$$\frac{t \cdot k \cdot T^{\mathrm{dlog}}_{\mathbb{G},q}}{1/(k+5)} \leq n^{1/2}. \tag{4.8}$$

As an example, let $k = 2^{20}, t = 2^{30}$, and $T^{\mathrm{dlog}}_{\mathbb{G},q} = q^{1/2}$. Note that we need to choose a value for p such that $q^{1/2} \leq L_{p^2}[1/3, 1.923]$ for (4.7) to make sense (otherwise, $T^{\mathrm{dlog}}_{\mathbb{G},q} = L_{p^2}[1/3, 1.923]$ and the inequality is violated). For the 80-bit security level, we have $L_{p^2}[1/3, 1.923] \geq 2^{190}$ which requires a 3270-bit p.

Thus, for 80-bit security, we need a 240-bit q, 3270-bit p, and 380-bit n for $k = 2^{20}$ and $t = 2^{30}$. The complete list of parameters for different choices of t and k is given in Table 4.1.

4.6.2 Efficiency

We compute three efficiency metrics considered by Garg and Gupta in [17]. We do not consider minor optimizations for any of these metrics, in the interest of comparing values obtained here to those in [17].

Public key size. The public key is $\mathsf{vk} = (\mathsf{vk}_{SP}, \mathsf{crs}_1, \mathsf{crs}_2, q)$. Here, crs_1 and crs_2 are of the form $(g, g^a, g^b, g^{ax}, g^{by}, g^z)$, where $z = x + y$ or $x + y - 1$ for crs_1, and $z = x + y + 1$ or $x + y$ for crs_2. Since crs_2 is the shift of crs_1, only 6 elements of group $\mathbb{G}$ are required to represent the two crs's. vk_{SP} consists of $2k + 25$ group elements, if the message to be signed is k group elements. Since in the blind signature scheme, the structure preserving signature is used to sign a blinded message which is actually a commitment comprising of 6 group elements, vk_{SP} contains 37 group elements. The total size is 43 group elements. (Note that one can reduce this size by minor optimizations, but we won't consider them here.) Each group element is a point on the elliptic curve $E/\mathbb{F}_p$, and requires $\log p + 1$ bits. Therefore, the total size

is $43(\log p + 1)$ bits. In contrast [17] computes the public key size to be $43\log n$ bits.

Signature Size. The final signature is a Groth-Sahai NIZK proof. From Table 4 in [19], to produce a GS-NIZK proof of a statement containing m variables, each either in $\mathbb{Z}_n$ or $\mathbb{G}$, ℓ pairing product equations/multiscalar multiplication equations, then under the DLIN assumption, the GS-NIZK proof requires $3m + 9\ell$ elements of $\mathbb{G}$. The blind signature is comprised of $m_{blind}, r, \sigma_{SP}$ and the equations $\mathsf{Com}_{\mathbb{G}}(m; r) = m_{blind}$ and $\mathsf{SPVer}(\mathsf{vk}_{SP}, m_{blind}, \sigma_{SP}) = 1$. From the instantiation seen in §4.5, $m_{blind} = (g, g^a, g^b, g^{ax}, g^{by}, m \cdot g^{x+y})$, $r = (x, y) \in \mathbb{Z}_n^2$, and σ_{SP} is the signature produced by SIG1 instantiated under the DLIN assumption containing 17 elements of $\mathbb{G}$. Thus, there are a total of 25 variables, which require 75 elements of $\mathbb{G}$ in the blind signature. To verify $\mathsf{Com}_{\mathbb{G}}(m; r) = m_{blind}$, we need 3 multiscalar multiplication equations, and to verify $\mathsf{SPVer}(\mathsf{vk}_{SP}, m_{blind}, \sigma_{SP}) = 1$, where the signature scheme is SIG1 under the DLIN assumption, we need 9 pairing product equations (see §4.5). Thus, the above two equations require $12 \times 9 = 108$ elements of $\mathbb{G}$. The signature comprises a total of 183 group elements, with size $183 \log p$ bits. In [17] this is computed as $183 \log n$ bits.

Communication complexity. The overall communication complexity is $18\log q + 41$ elements of $\mathbb{G}$. This requires $(18 \log q + 41) \log p$ bits. In [17] this is computed as $(18 \log q + 41) \log n$ bits.

Chapter 5

Concluding remarks

In this work, we analysed non-constructive security reductions for two cryptographic schemes, an NMAC related scheme called 2-Lane NMAC by Yasuda [33], and a blind signature scheme by Garg and Gupta [17]. To compute security parameters of the two schemes, we considered the fastest known non-constructive attacks on the underlying primitives, instead of considering only constructive attacks as was done in the original works. This resulted in larger security parameters and lower efficiencies.

For 2-Lane NMAC, we attempted to obtain a constructive proof of the security result in [33] and highlighted the difficulty in doing so. We found that the actual security offered by Yasuda's non-constructive theorem was much closer to the security offered by the constructive version presented here (which is really the same as the security offered by GNMAC), as opposed to what is claimed in the paper.

We also investigated the use of coin-fixing in the security proof of a PBKDF scheme considered by Zhou et al. [34]. We found the use of coin-fixing to be incorrect and provided a revised proof. Further, we observed that the result obtained without the use of coin-fixing (Theorem 2.8) was stronger than the result obtained using coin-fixing (Theorem 2.10). For the PBKDF scheme, since it is possible to obtain an upper bound without the use of coin-fixing that is almost tight with a matching attack, the use of coin-fixing is not necessary in the proof, and results in a weaker security result than what can be obtained without its use.

The following are possible directions for future work.

- It remains to see if the original security theorem from [33] for 2-Lane NMAC

can be proved in the constructive setting. If this is not possible, then one would have to rely on Theorem 3.6 which is non-constructive to evaluate the proof-based security of 2-Lane NMAC.

- Analyze the impact of non-constructivity on parameters used for HMAC as a randomness extractor in TLS [14].

- Since non-constructive attacks for the underlying primitives (PRF property and DLOG-q in a symmetric bilinear group) in the schemes considered in this thesis are not well studied, there could be faster attacks. For example, the precomputation considered in [10] for NFS in a prime field $\mathbb{F}_p$ uses a table of size $L_p[1/3, .951]$ that takes $L_p[1/3, (64/9)^{1/3}]$ to compute, after which individual discrete logarithms can be computed in time $L_p[1/3, 3^{1/3}]$. There remains the possibility that with larger precomputation, the online running time can be further reduced.

- In Chapter 4 we made the heuristic assumption that the speedup for NFS with precomputation for discrete logarithms in $\mathbb{F}_p$ where p is prime (see [10]), also applies to the finite field $\mathbb{F}_{p^2}$. It is worthwhile to investigate this assumption.

- Another direction for future work is to look for efficient, round-optimal constructions for blind signature schemes in the standard model with reasonably tight, constructive security proofs.
  We note that round-optimal blind signature schemes were presented in [16, 15] together with reductionist security proofs that use neither complexity leveraging nor non-constructivity. The schemes are significantly more efficient than the Garg-Gupta blind signature scheme. The authors of [16, 15] claim that their schemes are in the standard model. However, their schemes utilize the structure-preserving signature scheme on equivalence classes from [20] whose only known security proof is in the generic group model. Thus, the security proof for the schemes in [16, 15] cannot be considered to be in the standard model.

References

[1] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, M. Ohkubo, Constant-size structure preserving signatures: generic constructions and simple assumptions, *Advances in Cryptology — Asiacrypt 2012*, LNCS 7658, Springer, 2012, pp. 4-24.

[2] G. Adj, I. Martínez, N. Cruz-Cortés, A. Menezes, T. Oliveira, L. Rivera-Zamarripa, F. Rodríguez-Henríquez, Computing discrete logarithms in cryptographically-interesting characteristic-three finite fields, available at https://eprint.iacr.org/2016/914

[3] R. Barbulescu, P. Gaudry, A. Guillevic, F. Morain, Improving NFS for the discrete logarithm problem in non-prime finite fields, *Advances in Cryptology — Eurocrypt 2015*, LNCS 9056, Springer, 2015, pp. 129-155.

[4] R. Barbulescu, P. Gaudry, A. Joux, E. Thomé, A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic, *Advances in Cryptology — Eurocrypt 2014*, LNCS 8441, Springer, 2014, pp. 1-16.

[5] M. Bellare, New proofs for NMAC and HMAC: Security without collision resistance, *Journal of Cryptology*, **28** (2015), pp. 844-878.

[6] M. Bellare, P. Rogaway, Code-based game-playing proofs and the security of triple encryption, available at https://ia.cr/2004/331

[7] D.J. Bernstein, T. Lange, Non-uniform cracks in the concrete: the power of free precomputation, *Advances in Cryptology — Asiacrypt 2013*, LNCS 8270, Springer, 2013, pp. 321-340.

[8] D. Boneh, X. Boyen, H. Shacham, Short group signatures, *Advances in Cryptology — Crypto 2004*, LNCS 3152, Springer, 2004, pp. 41-55.

[9] R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited, *Journal of the ACM*, **51** (2004), pp. 557-594.

[10] A. Commeine, I. Semaev, An algorithm to solve the discrete logarithm problem with the Number Field Sieve. *Public Key Cryptography — PKC 2006*, LNCS 3958, Springer, 2006, pp. 174-190.

[11] H. Corrigan-Gibbs, D. Kogan, The discrete-logarithm problem with preprocessing, *Advances in Cryptology — Eurocrypt 2018*, LNCS 10821, Springer, 2018, pp. 415-447.

[12] eBACs: ECRYPT Benchmarking of Cryptographic Systems, https://bench.cr.yp.to/results-hash.html.

[13] I. Damgård, A design principle for hash functions, *Advances in Cryptology — CRYPTO '89* LNCS 435, Springer-Verlag, 1989, pp. 416-427.

[14] P. Fouque, D. Pointcheval, S. Zimmer, HMAC is a randomness extractor and applications to TLS, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, ACM Press, 2008, pp. 21-32.

[15] G. Fuchsbauer, C. Hanser, C. Kamath, D. Slamanig, Practical round-optimal blind signatures in the standard model from weaker assumptions, *Security and Cryptography for Networks (SCN) 2016*, LNCS 9841, Springer, 2016, pp. 391-408.

[16] G. Fuchsbauer, C. Hanser, D. Slamanig, Practical round-optimal blind signatures in the standard model, *Advances in Cryptology — Crypto 2015*, LNCS 9216, Springer, 2015, pp. 233-253.

[17] S. Garg, D. Gupta, Efficient round optimal blind signatures, *Advances in Cryptology — Eurocrypt 2014*, LNCS 8441, Springer, 2014, pp. 477-495.

[18] R. Granger, T. Kleinjung, J. Zumbrägel, Breaking '128-bit secure' supersingular binary curves, *Advances in Cryptology — Crypto 2014*, LNCS 8617, Springer, 2014, pp. 126-145.

[19] J. Groth, A. Sahai, Efficient non-interactive proof systems for bilinear groups, *Advances in Cryptology — Eurocrypt 2008*, LNCS 4965, Springer, 2008, pp. 415-432.

[20] C. Hanser, D. Slamanig, Structure-preserving signatures on equivalence classes and their application to anonymous credentials, *Advances in Cryptology — Asiacrypt 2014*, LNCS 8873, Springer, 2014, pp. 491-511.

[21] N. Koblitz, A. Menezes, Pairing-based cryptography at high security levels, *Cryptography and Coding 2005*, LNCS 3796, Springer-Verlag, 2005, pp. 13-36.

[22] N. Koblitz, A. Menezes, Another look at generic groups, *Advances in Mathematics of Communications*, **1** (2007), pp. 13-28.

[23] N. Koblitz, A. Menezes, Another look at "Provable Security", *Journal of Cryptology*, **20** (2007), pp. 3-37.

[24] N. Koblitz, A. Menezes, Another look at HMAC, *Journal of Mathematical Cryptology*, **9** (2013), pp. 225-251.

[25] H. Krawczyk, M. Bellare, R. Canetti, HMAC: keyed-hashing for message authentication, *RFC 2104*, available at https://tools.ietf.org/html/rfc2104

[26] R. Merkle, A certified digital signature, *Advances in Cryptology — CRYPTO '89*, LNCS 435, Springer-Verlag, 1989, pp. 218-238.

[27] J. Pollard, Monte Carlo methods for index computation mod p, *Mathematics of Computation*, **32** (1978) pp. 918-924

[28] B. Preneel, P. van Oorschot, On the security of iterated message authentication codes, *IEEE Transactions on Information Theory*, **45** (1999), pp. 188-199.

[29] P. Rogaway, Formalizing human ignorance: collision-resistant hashing without the keys, *Progress in Cryptology — Vietcrypt 2006*, LNCS 4341, Springer, 2006, pp. 211-228.

[30] V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology — Eurocrypt 1997*, LNCS 1233, Springer, 1997, pp. 256-266.

[31] E. Teske, Computing discrete logarithms with the parallelized kangaroo method, *Discrete Applied Mathematics*, **130** (2003) pp. 61-82.

[32] F. Yao, Y. Yin, Design and analysis of password-based key derivation functions, *Topics in Cryptology – CT-RSA 2005*, LNCS 3376, Springer, 2005, pp. 245-261.

[33] K. Yasuda, Multilane HMAC – security beyond the birthday limit, *Progress in Cryptology — Indocrypt 2007*, LNCS 4859, Springer-Verlag, 2007, pp. 18-32.

[34] J. Zhou, J. Chen, K. Pan, C. Zhao, X. Li, On the security of key derivation functions in Office, *2012 International Conference on Anti-Counterfeiting, Security and Identification (ASID)*, IEEE, 2012, pp. 1-5.