On the Security of Leakage Resilient Public Key Cryptography

by

Dale Brydon

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, 2012

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

Side channel attacks, where an attacker learns some physical information about the state of a device, are one of the ways in which cryptographic schemes are broken in practice. "Provably secure" schemes are subject to these attacks since the traditional models of security do not account for them. The theoretical community has recently proposed leakage resilient cryptography in an effort to account for side channel attacks in the security model. This thesis provides an in-depth look into what security guarantees public key leakage resilient schemes provide in practice.

Acknowledgements

There are a number of people who have been integral to the creation of this thesis. First, I would like to thank my friends and family for their support and warmth. My fiancee Sarah, in particular, helped to keep me sane during some of the harder periods. Also particularly important are my fellow grad students who have been an invaluable resource – a special thanks goes out to Andrew, Gurleen, Marco, Ben, David, Becky, and Leanne.

I would also like to thank my readers David Jao and Edlyn Teske-Wilson for their helpful feedback. Sherman Chow also provided feedback during the early stages of my research. I am grateful to Catherine Gebotys who gave me permission to use an image from one of her papers. Additionally, I would like to thank Neal Koblitz for his role in creating a series of papers taking a critical look at the value of "provably secure" cryptography in practice, including the paper that forms the basis for this thesis.

Finally, and most importantly, I wish to thank my advisor Alfred Menezes who has been a better resource than I could have ever imagined. Alfred's guidance has always been excellent and his input has made the once impossible task of writing a thesis a reality.

Dedication

To Dan Bernstein, Ian Goldberg, Kevin McCurley, and Moti Yung, who taught me the meaning of leakage.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Mathematicians are often not concerned about the practicality of their work. Nonetheless, they must make sure that the problem they solve is meaningful in some way. Theoretical cryptographers are faced with an additional challenge: the problems they wish to solve frequently arise in practical situations. Unfortunately, it is usually not useful to solve a practical problem with a theoretical solution. This is particularly the case when the problem does not have a nice theoretical formulation (that is to say, when it is not meaningful or interesting without its practical context). The goal of this thesis is to examine such a practical problem and the response from the theoretical community.

Traditionally, research papers in theoretical cryptography have assumed that an attacker has access to only the inputs and outputs of a cryptographic algorithm. For example, in the case of an encryption scheme, an attacker has access only to plaintexts (the input) and ciphertexts (the output). In public key cryptography the attacker also has access to any public information, like the public key itself. Nonetheless, a fundamental assumption is that the attacker never learns any information about the internal state of the algorithm. This is what is known as a black box assumption and so cryptographic algorithms are often referred to as black boxes.

These assumptions, however, tend to be violated in practice. To be realized in practice, a scheme must have a particular physical implementation. The implementation emits information about the physical processes involved in the computation of the cryptographic algorithm. For example, throughout the computation the device consumes power, emits electromagnetic radiation, and also makes noise (possibly inaudible to the human ear). All of these types of emissions can leak information about the internal state of the algorithm which can in turn lead to the scheme being broken. Throughout this thesis, we will refer to attacks which gather and use information about the internal state of a cryptographic algorithm as *side channel attacks*.

Note that other definitions of side channel attacks can be formulated. For example, some authors restrict the notion to attacks where an attacker can only listen in and cannot disrupt the computation in any way. Additionally, attacks where the hardware is accessed in some non-standard way (for example by depackaging a chip) are not deemed side channel attacks by some. While we do not employ such restrictions, we do not consider

attacks where the cryptographic algorithm is totally circumvented. In particular, we do not consider scenarios where an attacker directly learns a user's password or secret key – for example using a keylogger or social engineering – to be side channel attacks.

Starting with the work of Micali and Reyzin in 2004 [57], there has been an effort to include *all* side channel attacks in the theoretical model of security. Schemes secure in the new models are referred to as *leakage resilient*. However, as noted by Koblitz and Menezes [45], the failure of authors in this area to address many aspects of the practical problem may mean that their schemes are, in fact, a step backwards in terms of achieving security against side channel attacks. This thesis is an in-depth discussion of the theoretical work and its successes and failures. In doing this, we hope to steer future work in directions we feel are most likely to aid practitioners. We also attempt to address some of the issues that remain largely ignored in the theoretical works.

In a certain sense this thesis can be seen as a first response to the following call of Wichs:

> Many interesting and important questions remain unanswered. Perhaps most importantly, the proposed model of leakage may still not be the "right one." There is certainly a need for better empirical analysis of whether the model is sufficient to capture all realistic examples of side-channel attacks [71, p. 145].

While for the most part our analysis is not empirical, it provides a partial answer to this "most important" question as well addressing and asking many others. We should note that our analysis focuses exclusively on the work done on public key schemes.

We stress two important details that are fundamental to understanding the purpose of this work. First, we do not take any issue with the idea of theoretical works or works that are not practical in nature. Nonetheless, we feel that it is fair to analyze works that purport to solve a practical problem in a practical context. Second, we do not wish to disparage the quality and ingenuity of the mathematics involved in any of the works we criticize. Our issue is not with the, often very clever, mathematics the authors use in their papers, but simply the lack of context surrounding that mathematics. In fact, it may turn out that some of the techniques developed in these papers play an important role in creating schemes which are indeed more secure against side channel attacks.

We now present an outline of what is to come. The rest of this chapter gives some preliminary background of the mathematics and notation we will use, as well as a detailed look at side channel attacks. Chapter 2 presents an overview of leakage resilient public key cryptography. It contains a high-level explanation of the various new security models, an overview of results in the field, and criticisms of both the theoretical response as a whole and of specific leakage models. Chapters 3 and 4 contain an in-depth look at two different leakage resilient signature schemes. By examining them, we hope to gain a better understanding of the kinds of techniques used in this field as well as provide a more quantitative understanding of the security guarantees and efficiency of the proposed schemes. Chapter 5 contains our conclusions including next steps and open questions.

1.1 Mathematical Background

The purpose of this section is to explain some of the terminology we will use throughout this thesis. Additionally, we will present the Schnorr signature scheme as an example to which later signature schemes can be compared, as well as to help make the side channel attacks discussed below more concrete.

1.1.1 Notation

Whenever it is used PK refers to the public key of an arbitrary scheme and SK the corresponding secret key. We use log to mean the base 2 logarithm. When x is a number, the notation $|x|$ refers to the (approximate) bit length of x. All schemes are initialized in terms of a security parameter k. When we refer to a scheme (or algorithm) as being *probabilistic polynomial time* (PPT), we mean that it runs in a number of steps that is polynomial in k. As such, all of its associated parameters (e.g. $|PK|$, $|SK|$, the plaintext and ciphertext sizes, etc.) must also be polynomial in k. In practice, when a scheme is implemented at a security level k, it means that we expect an attacker to need to perform at least 2^k computations to break the scheme. In theory, when a scheme is secure with security parameter k, it means that no attacker can break the scheme unless there is a (probabilistic) algorithm that can solve the underlying hard problem in time polynomial in k. A (positive) function f is *negligible* if for every (positive) polynomial p there exists N so that for every $k > N$, $f(k) < 1/p(k)$. When we refer to a problem as being *hard*, we mean that no PPT algorithm to solve the problem is known. The notation $x \in_R S$ means that x is picked uniformly at random from the set S. Vectors are denoted by bold-faced roman letters. When it cannot be avoided we will sometimes have to number vectors; to distinguish between a vector and a component of a vector we will make the subscript bold in these cases (i.e. $\mathbf{x_i}$ is a vector and $\mathbf{x}_i$ is the i^{th} component of the vector $\mathbf{x}$).

1.1.2 Cryptographic Background

A *hash function*, H, takes a string of arbitrary length as input and outputs a fixed length string. One desirable property for hash functions used in cryptographic contexts is collision resistance. We say H is *collision resistant* if it is hard to find inputs, x and y, so that $H(x) = H(y)$ and $x \neq y$. A *random oracle* is a hash function whose outputs are random but consistent (i.e. two different calls to the oracle with the same input will produce the same output).

Most schemes discussed in this thesis use a group G of prime order q. To simplify analysis, we will imagine such schemes as being implemented using elliptic curves (as they are in practice). Over some elliptic curve groups the following problem is believed to be hard. Given some generator $P \in G$ and some element $Q \in_R G$, find an exponent $x \in \mathbb{Z}_q$ so that $xP = Q$. This is known as the discrete log problem and henceforth we will refer to it as DLOG. The best known algorithm for DLOG in this case finds x in about $\sqrt{q}$ steps. Thus when our security parameter is k we pick q so that $|q| \approx 2k$.

In many elliptic curve groups, the following problem is believed to be hard. Given P and three points $xP,\ yP \in_R G$ and $zP \in G$ decide if $xyP = zP$, where $x, y, z \in \mathbb{Z}_q$ (note $x,\ y$, and z are unknown). This is known as the decisional Diffie-Hellman problem (DDH). If instead of being given zP we are asked to compute xyP, this is known as the computational Diffie-Hellman problem (CDH). Note that a PPT algorithm which solves DLOG can be used to solve CDH in polynomial time. Similarly a PPT algorithm which solves CDH can be used to solve DDH in polynomial time. Thus, DLOG is at least as hard as CDH which is in turn at least as hard as DDH. The word "assumption" after a believed hard problem, means we assume that problem is hard. Therefore the DDH assumption implies the CDH assumption which implies the DLOG assumption.

Some schemes also make use of a bilinear pairing.

Definition 1.1.1. Given three groups G_1, G_2, and G_T all of prime order q, a function $e : G_1 \times G_2 \to G_T$ is a *bilinear pairing* if the following three properties hold for all generators $g_1 \in G_1$ and $g_2 \in G_2$:

Non-degeneracy: $e(g_1, g_2) \neq ID_T$.
Bilinearity: For all $a, b \in \mathbb{Z}_q^*$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
Computability: e can be efficiently computed.

We often drop the word "bilinear" when referring to pairings in a cryptographic context. Pairings where $G_1 = G_2$ are called *symmetric* and those where $G_1 \neq G_2$ are called *asymmetric*. (Strictly speaking we should replace equality in the last statement with the availability of efficiently computable isomorphisms between the two groups.) In practice, such pairings can be constructed from special classes of elliptic curves. For an introduction to pairings in a cryptographic context see [55].

The operation of scalar point multiplication is one of the essential ingredients in any elliptic curve cryptosystem. We present two different algorithms that can be used for computing these multiplications. The naïve double-and-add algorithm will later be used to illustrate the susceptibility of this operation to attack. The Montgomery powering ladder (also just the Montgomery ladder), first proposed to speed up factoring attacks on RSA, provides a nice example of how the mathematical structure of groups can be used to compute the same value in different ways. In both algorithms we refer to the group identity as the point ∞. The description of the algorithm in Figure 1.1 is well-known, but our definition is from [64]. The algorithm in Figure 1.2 is due to Montgomery [58], but more clearly presented for our purposes in [40].

A signature scheme consists of three phases: key generation, signing, and verification. In the key generation stage the algorithm outputs a public-key/secret-key pair (PK, SK). The signing phase takes as input a public key PK, a secret key SK, and a message m and outputs a signature σ for m. The verification phase takes as input a public key PK, a message m, and signature σ and outputs a value indicating whether or not σ is a valid signature for m. Usually we require the property that if a message is signed with SK, then the verification of that signature with the public key PK always succeeds. The standard notion of security for a signature scheme is the following.

Input: Point P and m-bit multiplier $r = \sum_{i=0}^{m-1} r_i 2^i$ where $r_i \in \{0, 1\}$
Output: $Q = rP$

1: $Q := P$
2: **if** $r_0 = 1$ **then**
3: $\quad R := P$
4: **else**
5: $\quad R := \infty$
6: **end if**
7: **for** $i = 1$ to $m - 1$ **do**
8: $\quad Q := 2Q$
9: $\quad$ **if** $r_i = 1$ **then**
10: $\quad\quad R := R + Q$
11: $\quad$ **end if**
12: **end for**

Figure 1.1: The double-and-add point multiplication algorithm

Input: Point P and m-bit multiplier $r = \sum_{i=0}^{m-1} r_i 2^i$ where $r_i \in \{0, 1\}$
Output: $R_0 = rP$

1: $R_0 := P$
2: $R_1 := 2P$
3: **for** $i = m - 1$ downto 0 **do**
4: $\quad$ **if** $r_i = 0$ **then**
5: $\quad\quad R_1 := R_0 + R_1$
6: $\quad\quad R_0 := 2R_0$
7: $\quad$ **else**
8: $\quad\quad R_0 := R_0 + R_1$
9: $\quad\quad R_1 := 2R_1$
10: $\quad$ **end if**
11: **end for**

Figure 1.2: The Montgomery powering ladder

Definition 1.1.2. A signature scheme is *existentially unforgeable under adaptive chosen message attacks* if no PPT forger succeeds at the following game with non-negligible probability.

- The challenger generates a key pair (PK, SK) and sends PK to the forger.
- The forger requests and receives signatures on polynomially many messages of her choosing.
- The forger chooses a message m and outputs a pair (m, σ).

The forger succeeds if σ is a valid signature for m and she has not previously requested a signature for m.

Figure 1.3 presents the (generalized) Schnorr signature scheme. It makes use of a group G of prime order q where DLOG is hard, a random oracle $H : \{0,1\}^* \to \mathbb{Z}_q$, and a generator $g \in G$ as system parameters.

Key Generation: To generate her keys, Alice selects $x \in_R \mathbb{Z}_q^*$. Alice's public key is $PK = g^x$ and her secret key is $SK = x$.
Signing: To sign a message m, Alice first selects $r \in_R \mathbb{Z}_q^*$. Next, she computes $A = g^r$ and calculates the hash $c := H(m||A)$. Finally, she outputs the signature $(A, cx + r \bmod q)$.
Verification: To verify Alice's signature (A, α) on a message m, Bob first computes $c := H(m||A)$. Then he checks that $A \cdot (g^x)^c = g^\alpha$. If equality holds, he accepts the signature.

Figure 1.3: The Schnorr signature scheme.

Theorem 1.1.3 ([66])**.** *Assuming the hardness of DLOG in G, the Schnorr signature scheme is existentially unforgeable under chosen message attacks.*

Schnorr first proposed this scheme in [66] where the group used was a subgroup of $\mathbb{Z}_p^*$ for $p \gg q$. The generalization we present is well-known, although we slightly change the structure of the signature for ease of comparison with the scheme we present in Chapter 3.

Public key encryption schemes tend to be used as key encapsulation mechanisms in practice. That is, suppose Alice wishes to encrypt the message m using a public key scheme for Bob. Their agreed upon public key protocol will include a symmetric key cipher. So, to perform the encryption Alice picks a random key x for the symmetric key cipher and encrypts m using x. Then using the public key scheme, she encrypts x and sends both encryptions to Bob. To decrypt, Bob decrypts x using the public key scheme and then uses that to decrypt m using the symmetric key scheme. This mode of operation is referred to as hybrid encryption. Hybrid encryption is used since public key schemes tend to be much slower than symmetric key schemes, and since the message m may in fact be very long whereby encrypting it may require calling the encryption algorithm many times. Because of this observation, it may be fair to assume that public key schemes encrypt only random messages in practice – this will be important in modelling side channel attacks later in the thesis.

1.1.3 A Tiny Bit of Information Theory

The concept of min-entropy gives us an idea of how close a random variable's distribution is to uniform. The concept is well-known; the definition we give is from Katz [42].

Definition 1.1.4. The *min-entropy* of a random variable $X \in \{0,1\}^n$ is defined as

$$H_\infty(X) := \min_{x \in \{0,1\}^n} \{-\log_2 \Pr[X = x]\}.$$

The *conditional min-entropy* of X given an event V is

$$H_\infty(X \mid V) := \min_{x \in \{0,1\}^n} \{-\log_2 \Pr[X = x \mid V]\}.$$

We often refer to min-entropy as a number of bits, since if a variable X can take on 2^η different equally likely values, it has min-entropy η. In this case we might also say that X has η bits of entropy. A useful fact [21] is that on average the conditional min-entropy of X conditioned on the output of some function with δ bit outputs is at most δ less than the min-entropy of X. More precisely, given $X \in S$ and $f : S \to S'$ with $|S| = 2^\delta$, we have $\mathrm{E}[H_\infty(X \mid f(X))] \geq H_\infty(X) - \delta$. So even after she observes δ bits of leakage, we can expect an attacker to have reduced her search space by a factor of at most 2^δ.

The number of bits that are 1 in a binary string is called the *Hamming weight* of the string. The number of bit positions in which two binary strings differ is referred to as the *Hamming distance* of the two strings.

1.2 Side Channel Attacks

1.2.1 History

Side channel attacks appear to have been discovered in the West during World War II. While unintended emissions had previously been used to glean classified information, they were not used to break encrypted channels. According to a paper declassified by the NSA [30], during the war a researcher at Bell Telephone discovered that their cipher device was leaking the plaintext via electromagnetic emanations. Since the device was deployed by the Army and Navy at the time, Bell contacted them immediately. Eventually Bell convinced the Forces of the threat and were commissioned to modify the device to become more secure. They presented a new design but unfortunately,

> Signal Corps took one look at it and turned thumbs down. The trouble was, to contain the offending signals, Bell had to virtually encapsulate the machine. Instead of a modification kit that could be sent to the field, the machines would have to be sent back and rehabilitated. The encapsulation caused problems of heat dissipation, made maintenance extremely difficult, and hampered operations by limiting access to the various controls [30, p. 2].

Bell engineers managed to reduce emissions to a reasonable level, but because their solution failed to understand the entirety of the practical context, it was essentially useless. It also seems that knowledge of these attacks was lost after the war. It wasn't until 1951 that the CIA rediscovered the attacks – on the same cipher machine no less. At that point, the NSA began the process of testing all their cipher equipment and found, to their dismay, that all of their machines were vulnerable to side channel attacks [30].

In Britain in the mid-to-late 1950s MI5 (British counter-intelligence) and GCHQ developed a couple of side channel techniques to break various schemes. In particular, they broke the Hagelin cipher machine based on the sounds made by the rotors being set and the French high-grade ambassadorial cipher by noticing that there was a faint second signal on the line which turned out to be the plaintext [72].

It is unclear what Soviet intelligence organizations knew about this subject, although they almost certainly knew of some of these types of attacks at about the same time as they were discovered in the West. It is certainly possible that their knowledge of side channel attacks was much more advanced than that of their Western counterparts.

It is interesting to note that the idea of such attacks was well publicized in the 1980s. First, there were the presentations of Wim van Eck on the possibility of recovering the image from a computer monitor (or other similar device) based on its electromagnetic emissions. This phenomenon, known as van Eck phreaking, got widespread attention from the general public. Two years later, Peter Wright, a former MI5 intelligence officer, published his autobiography [72]. Likely as a result of the legal battle by the British government to prevent its publication, the book became a bestseller. It contains, in no uncertain terms, a discussion of how side channel attacks can be used to break cryptosystems. Further, parts of the NSA's TEMPEST program to shield against compromising emissions were discussed in unclassified settings in the 1960s. Despite the general knowledge of these techniques, the first paper on side channel attacks did not appear in the cryptographic literature until 1996. In the meantime, theoretical cryptographers loudly proclaimed their work to be invincible because it came with the all important *proof* of security.[1]

Kocher's 1996 publication on timing attacks [46] opened up a new field of cryptographic research. From that point on, side channel attacks have become a hot topic of research and an important consideration for practitioners. A number of different types of attacks have been discovered as well as a host of countermeasures. In the sections that follow we will discuss a few different side channel attacks as well as a variety of countermeasures. Before doing so, it will be helpful to have a better understanding of the sorts of scenarios that give rise to side channel attacks.

One of the points that authors of papers on side channel attacks frequently make is that the equipment needed for such attacks is usually inexpensive. In many cases the total cost of the equipment used is under $500. However, these attacks come with a substantial hidden cost. Two important factors must be considered. First of all, attacks often require physical proximity to the target device. Obtaining such access to the device may be risky

[1]Koblitz and Menezes discuss these claims in [45], part of a series of papers examining the problems of a theoretical community not paying sufficient attention to the context of practice.

(which we can consider as an expense) or at the very least quite difficult. In particular, it might lead to the attackers being detected. Another issue, is that attacks often need to be custom-designed for the target platform and thus require technical expertise. The cost of the labour needed to do the work may be many times the cost of the equipment. These costs must be weighed against the relative importance of the information that is trying to be recovered. Additionally, there may be other attacks possible to obtain access to the data.

Security is only as strong as its weakest link, and so an attacker will only use a side channel attack insofar as it is the best way to recover the secrets. In many practical scenarios things like malware, social engineering, or exploiting weak passwords will all be more likely candidates for attacks. Even if the cryptography itself is exploited, other areas of the system may be targeted. For example, an attack might exploit a weakness in the way random numbers are generated. A recent paper [49] showed that about 4% of the moduli used in the RSA public keys examined shared at least one factor with another public key in the set studied. All such keys are compromised since knowing the secret key (and thus factorization) of a modulus with p as a factor allows one to compute the factorization of any other modulus with p as a factor.

Thus we feel that, so far as it is helpful, we should consider the attacker to be an intelligence organization like the NSA, GCHQ, or CSEC. These organizations have access to all the resources (in particular the personnel) needed to perform the attacks. Thus the single entity Eve will be replaced with the much more nefarious group of attackers EVE.[2] The target for such attacks must also be in possession of a relatively valuable secret, and so potential candidates include governments, companies with good security policies, and individuals believed to be a threat to intelligence organizations, such as suspected terrorists or spies.

In general, we consider the device performing the cryptographic operation to be some form of dedicated device (i.e. only used for cryptography) that is computationally restricted. The prototypical example of this would be the chip used in a smart card. In particular, we *usually* do not consider the attacked device to be a general purpose computer, like a laptop, since they are both vulnerable to malware type attacks and generally much noisier (in a communication sense) than dedicated devices. Further, certain attacks may work better depending on whether the cryptographic operation is implemented in hardware or software. When we say a scheme is implemented in software, we, in general, are referring to the target device as being some form of programmable microcontroller.

The fundamental premise of any side channel attack is that differing input (message and key) may lead to (subtly) different physical output. Therefore, if some relation can be observed between certain key and/or message bits and the observed physical emission, it may be possible to determine portions of the secret information. We usually assume the attacker has the ability to induce the target device to perform its operation on some number of messages of the attacker's choosing.

[2]Despite its capitalization, EVE is not an acronym. Thus it serves as an uncrackable code, ready to stymie unwary cryptanalysts looking for meaning where none can be found.

1.2.2 Simple Power Analysis

Of the many attacks in the literature, simple power analysis (SPA) – introduced by Kocher et al. in 1998 [47] – is among the easiest to understand. The primary assumption made in such attacks is that an attacker can only perform a small number of operations on the target device. In the simplest case an attacker has access to the power consumed by a device while performing its operation on only one pair of key and message (with the key unknown and the message known). We refer to the power data collected by the attacker in this instance as a power trace or sometimes just a trace.

As an example of how even this little information can be of use, consider an attack on Schnorr. Such an attack might consist of observing the power consumed by a smart card while it signs a message.[3] Recall that during signing the exponentiation g^r is computed. In our case, this becomes the elliptic curve point multiplication rP which we will further suppose is implemented using the naïve double and add algorithm (Figure 1.1). Notice that in the algorithm for computing this value, if the bit i^{th} bit r_i of r is a 0, the card performs only a doubling operation; if it is a 1, then the card performs both a doubling and an addition. Since doubling and addition operations tend to have easily identifiable power signatures, the attacker can simply read off the bits of r by looking at the power trace. Figure 1.4 (from [32]) demonstrates exactly this phenomenon; here doubling steps (D) are clearly distinguishable from addition or sum steps (S). Notice that once an attacker learns any single per-message secret r, she can easily compute the secret key x using the known value $\alpha := cx + r \pmod{q}$, since c is also known.

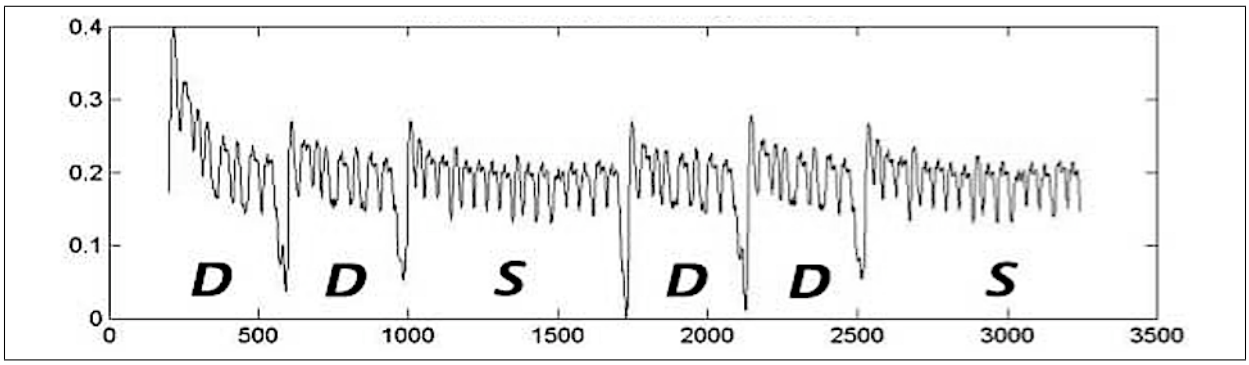

Figure 1.4: Partial power trace for an elliptic curve point multiplication.

Let's recap what the attackers would need to know to perform this attack. If we assume the device is idle while it is not signing, then it is trivial to know when the power trace for the signature starts and ends. Thus, the attackers would need only a rudimentary idea of what a double or addition operation might look like to determine the key. This is because the two operations will look different enough to avoid any confusion between the two unless countermeasures are employed. It is also likely that attackers with a decent idea of what the power signature of a double or an addition looks would not even need to know what method was used to implement the multiplication, since an examination of the power trace would reveal this.

[3]The attack we describe is totally independent of the message being signed.

Given the threat of such attacks, let us now imagine a situation in which a more suitable method of point multiplication is used – the Montgomery powering ladder presented in Figure 1.2. In this case, since both a doubling and an addition are performed regardless of the value of r_i, the simple power analysis attack described above will fail. However, another type of simple power analysis can still be used here. It relies on the ability of the attackers to obtain a device identical to the attacked device. Then they use their device to compute many different signatures. For each signature they can compute the per-message secret r, since they know the secret key.

For many different values of r the attackers have the power trace associated with the point multiplication rP. Then for each bit i of the per-message secrets, they group the traces into two categories: those where the bit r_i is 0 and those where it is 1. They then average all such traces in each case, obtaining "templates". In the end hopefully the average trace is relatively constant except for a spike at the time when the key bit is used in the point multiplication. Then, taking a single power trace of the target device, they compare the trace to the average traces for each of the bits in the per-message secret and so can make a guess for each bit of r. Here even though both a doubling and an addition are performed regardless of the value of r_i, the scheme may still be vulnerable unless the underlying field operations themselves are protected. See [54] for more information on template-style attacks on elliptic curves and [53] for more details on these attacks in general.

For completeness we describe a third type of simple power analysis attack, although it is much more applicable to symmetric key schemes. Let the input to a scheme be the message $m = m_1, \ldots, m_d$ and the secret key $x = x_1, \ldots, x_d$. Here the inputs are broken into d equally sized blocks (for example each m_i might correspond to a bit or a byte of m). It may be the case that for two different message inputs, the same intermediate value is computed. That is, suppose at some point the intermediate value $f(\cdot, \cdot)$ is computed. Then there may be pairs $(m_i, x_j) \neq (m'_i, x_j)$ such that $f(m_i, x_j) = f(m'_i, x_j)$. This is called a collision. Given that power consumption will depend on this intermediate value in some way, the attackers may be able to detect such collisions. Then, by using a template-style attack, when the attackers hypothesize that a collision has occurred, they will be able to restrict the options for the particular bit or byte (since we assume they know the input message) of the key that caused the collision. If they can do this for enough key bits/bytes, they may be able to reduce the keyspace to a small enough size to perform a brute-force search.

One thing to note is that we referred to a single power trace as being a small amount of information. While this is true in a certain sense, a single power trace can be megabits or even gigabits of information. Our description of SPA attacks is based on [53], which focuses on symmetric key schemes, but is very detailed, and [64], which deals with elliptic curve point multiplication. These references are also our sources for the differential power analysis attacks we will describe next.

1.2.3 Differential Power Analysis

Differential power analysis (DPA), also introduced in [47], depends on the attacker being able to obtain a large number of power traces from the attacked device. One scenario in which this might occur is that a group of attackers gain access to a smart card and its PIN, but wish to learn its secret key so that they can make additional copies.

Attacks in this case function similarly to the template attacks described in Section 1.2.2. However, instead of building templates, the attackers make guesses about how key and message pairs will affect power consumption. More concretely, suppose the attacked device contains a 16-bit processor and the group used has order $|q| \approx 256$. Each of the doublings and additions that occur in an exponentiation will then have to be split into chunks. That is to say, for example, when computing $2P$, each coordinate of the point P must first be broken into 16 blocks of 16 bits, which in this context is a word. The various computations involved in a doubling operation will then actually be done one word at a time. As is turns out, the power consumption caused by these operations may be heavily dependent on the structure of their input words. In particular, operations on words with low Hamming weight may have lower or higher power needs than those on words with high Hamming weight. It may also be the case that one of the inputs has a much larger effect on the power consumption than the other (e.g. the power consumption of $\texttt{add}(a, b)$ might be very strongly correlated with the Hamming weight of a and basically independent of the Hamming weight of b).

Now we can examine how the attackers might make use of such a scenario. Instead of a signature algorithm, let us consider the case where the attacked operation is an ElGamal-based decryption using the Montgomery powering ladder (Figure 1.2). In such a decryption Bob uses his secret key $SK = x$ to compute the point multiplication xP (where P is dependent on the ciphertext). The first stage in such an attack consists of recording the power trace while the device computes xP_i for many different values of P_i chosen by the attacker. Having gathered these traces, an attack might proceed as follows. The attackers select an operation, say a multiplication that occurs in the doubling, that they know has power consumption dependent on the Hamming weight of its first input. Their goal is determine which of R_0 or R_1 is being doubled and so find the key bit for that round. They attack the algorithm bit-by-bit and so their guess for the first bit is used to gather enough information to make a guess for the second bit and so on.

In the first round, the attackers know the Hamming weights of the two possible inputs to the multiplication. For each of their gathered power traces they use their guess for the power consumption characteristics of the multiplication to determine which of R_0 or R_1 is being doubled. Then, whichever one was guessed more often leads to a guess for the first key bit. Using this information, they can determine the Hamming weights of the inputs to the target multiplication in the second round. Using the same traces, they repeat this process to make a guess for the second bit. They use the first two bits and the same power traces to make a guess for the third bit and so on. When finished, they will have a guess for the entire key.

The effectiveness of the attack depends on the number of power traces gathered, the

power consumption characteristics of the target multiplication (or other operation), and the quality of the attackers' guesses about these characteristics. Also, if we assume that the probability p of success for each bit is only a function of the number of traces collected, then the probability of correctly guessing the i^{th} bit of the key is p^i. The overall probability of success is therefore $p^{|x|}$. For a 256 bit key we need $p \approx .9973$ for the attackers to succeed with probability at least 1/2. However, an attacker who knows the 128 least significant bits of x can find the discrete log of xP in about 2^{64} steps. In this case, $p \approx 0.9946$ gives an attacker a 50% chance of success.

Instead of one operation like the multiplication in our example, an attacker might look at some sequence of operations. Additionally, in some cases not just the Hamming weight matters and so an attacker could create a model for all 2^{16} input values. In other situations both inputs to an operation might have similar effects on power consumption. In these cases, the combined Hamming weight of both inputs might be important. The attack could proceed in a similar manner as above, although more traces would be needed for the same probability of success (assuming the other properties of the operation were similar).

Another approach that can be used is to perform a template-style attack, like the one described in Section 1.2.2. This requires the assumption that the attackers can make arbitrary requests to the attacked device as well as to an identical device for which they know the key. The attack proceeds as above, except that each power trace is used to calculate the probability p_i that key bit i equals 0 for all i. Using statistical methods, these probabilities are combined to give the overall probability that key bit i is 0. At the end the attacker simply picks the key that corresponds to the greater probabilities.

1.2.4 Cold Boot Attacks

While earlier we said that the attack target tends to be a device like a smart card and not a PC, this is not true for cold boot attacks [34]. The authors envisioned a scenario in which the entire hard disk of a computer is encrypted. Thus, whenever it is operating, the secret key for such a system must be stored in RAM.

The values in RAM dissipate over time while the RAM is not receiving power. However, reading the values shortly after power is lost means that many of the previous values will remain intact. Thus, one method of attack is to turn off the target system by cutting power (other shutdown techniques may give the system time to overwrite RAM values), insert a bootable USB disk into the computer which contains a stripped down operating system whose only purpose is to dump the contents of memory, and then search through the memory for plausible values of the secret key. The effectiveness of the attack can be increased by rapidly cooling the RAM immediately at the time the machine loses power. This can be done by using, for example, a normal "can of air" type duster. If the cooling method is used, it is also possible to remove the RAM and place it into another computer to perform the attack. If the attack target is a desktop, the attackers could swap out the RAM for their own and then leave to perform the attack later. This would conceivably take less than a minute while still being very hard to detect.

It is important to note that the attack requires the computer to be on or in sleep/ standby mode. However, some systems remain vulnerable even if they are completely shut off since they load the secret key into RAM without requiring a password.

The two remaining issues are how the attackers detect where in RAM the secret key is stored and what they do in the event that some key bits have been corrupted. To answer both questions, let's conceive of a slightly different situation in which the attacked computer is running a web server whose public key is known to the attackers. The attackers wish to learn the corresponding secret key. The public information and secret key will likely be stored together in some standardized format. Then, the attackers can search for the public information in RAM and use that to pinpoint what must be the private key. Subsequently, they can use the known public key to attempt to correct errors in the recovered secret key. However, if the secret key is badly corrupted more advanced methods are required.

One such method is that to speed computations, some implementations of cryptosystems contain extra private data. For example, this is true for RSA-style cryptosystems which frequently store not just the decryption exponent d, but also the prime factors p and q of the RSA modulus as well as $d \pmod{p}$ and $d \pmod{q}$ and sometimes also $q^{-1} \pmod{p}$. Each of these pieces of information can (in combination with the public key) be used to derive any of the others. Thus an attacker can further use these extra pieces of information to gather more information about the secret key.

Another important observation of [34] is that within a region of RAM, values tend towards 0 or 1 uniformly. That is to say, one "block" of RAM would either tend towards all its values becoming 0 or all its values becoming 1. Therefore, it is much easier to detect which of the secret key bits have decayed when performing the attack. It will be the case that either all the 1's or all the 0's in the recovered key will be true bits. Further, if a significant fraction of bits have decayed[4], then it will be extremely likely that the Hamming weight of the "block" will determine which direction the decay is occurring in (towards 0 or towards 1).

Techniques also exist to find and recover symmetric keys from RAM. These work using a similar principle as in the public key case when additional secret values are stored to speed up computations. However, if these extra secret values are not stored, then finding the secret key may be too computationally intensive. Nonetheless, if no decay occurs, it is feasible to test every appropriately-sized chunk of RAM to see if it turns out to be the key [34]. We feel this technique may also be possible when a small amount of decay has occurred, especially if the attacker improves the method by only testing values with plausible Hamming weight (since keys are chosen randomly, they will likely have Hamming weight very close to one half of their length).

1.2.5 Countermeasures

Before we begin our discussion of countermeasures, it is important to better understand what sort of information an attacker is actually exploiting. One observation from Sec-

[4]If not very many bits have decayed, the attackers can quickly recover the key using brute force methods.

tion 1.2.2 is that different operations have different physical outputs. Here, operations refer to the actual low-level computations being performed by the chip. However, something we did not discuss is that in many cases operations are performed in parallel. Thus the physical outputs of the attacked device might simultaneously depend on several different computations. This makes the job of the attacker much more difficult, since either the model is more complicated or the measurements are noisier. Another thing to consider is that in all the attacks above, the attackers can chose the input to the attacked device. While this is a scenario that should be protected against, attacks also exist in the case where the attackers know, but cannot choose, the input and where the input is unknown.

The implementation of an algorithm is tied to the device that will be performing the computation. The most obvious way this occurs is that chips of different word sizes (e.g. 8-bit vs. 64-bit) will need different implementations. These (device-specific) implementations play an important role in determining the utility of the physical output to an attacker. For example, AES is structured to be processed in 8-bit chunks and so a 64-bit implementation might simultaneously be performing operations using 8 different portions of the key. As noted above, this drastically increases the complexity of an attack on the device. Other hardware properties of the device are important as well. For example, the manner in which the device powers its registers might make some operations' power consumption more dependent on their input.

Whether the algorithm is implemented in hardware or software seems to have relatively little impact on the complexity of the attack. For example, in [53] the authors use 224 power traces to recover the key of a software implementation of AES and 6532 traces for a hardware version; in [24], however, the authors break the proprietary encryption scheme KeeLoq using between 10 and 30 traces in hardware and several thousand traces in software. Also, as can be seen from these examples, many factors determine the number of traces needed to recover the key. In [53] there is a discussion of a formula for estimating the number of power traces needed provided one can estimate the correlation between actual power consumption and the guess for these values (e.g. how much the actual power consumption depends on the Hamming weight of the input). Note that these are only a couple of data points and there are instances where a successful attack might require orders of magnitude more power traces (e.g. see Figure 11 of [59] which shows how the number of traces affect chances of success in a particular DPA attack).

Countermeasures come in two primary flavours, which we will refer to as hardware/ physical countermeasures and implementation countermeasures. Hardware/physical countermeasures are those which attempt to prevent attackers from learning the (unaltered) physical outputs. For example, an organization might consider placing their important cryptographic devices inside a Faraday cage to prevent electromagnetic emanations. Implementation countermeasures are themselves divided into two categories: hiding countermeasures where the physical outputs are made similar for all inputs, and masking countermeasures where the values being processed are altered in some way (that does not affect the final output). In our discussion below, general purpose countermeasures are based on [53], those for elliptic curves on [39], and cold boot attacks on [34].

Hardware and Physical Countermeasures

We will only briefly touch on the variety of countermeasures that prevent physical observations from occurring. These range from very low-tech solutions such as keeping equipment in a locked and guarded room, to highly engineered solutions such as encasing the device in a tamper-resistant mesh that overwrites the secret key with 0's when it is physically disturbed. Among the defences against power analysis attacks is the use of an onboard battery. Especially when coupled with tamper-resistant surroundings this makes it very hard for an attacker to learn the power consumed by such a device.

Many of these solutions only work in very specialized applications and may substantially increase the cost of the device. For example, smart cards tend not to use batteries for both cost and space reasons. In fact, smart cards are generally chosen as a vehicle for attack in the literature since they are very hard to secure physically (at a reasonable cost).

Almost all of the techniques for preventing cold boot attacks are of this sort. For example, important methods include always turning off a computer when it is to be left alone. Beyond that, it should not be left alone until a reasonable amount of time has passed (maybe 3 or so minutes) from when it was turned off. Further techniques include storing secret keys in special memory that is not subject to the attack (e.g. it is welded to the motherboard and cannot be read from a RAM dump), making sure keys are only in RAM while they are being used, writing over keys when the computer is turned off, and writing over RAM when the computer is turned on. Another countermeasure is to re-compute extra secret values as needed, as opposed to constantly storing them – this involves significant efficiency tradeoffs.

Hiding Countermeasures

To better discuss hiding countermeasures let us consider another kind of side channel attack. Timing attacks, as the name implies, measure the amount of time a device takes to perform a cryptographic operation. Depending on the type of attack, it may be possible to measure the time required to compute chunks of the algorithm instead of the whole thing. In any case, hiding countermeasures prevent timing attacks by making all operations take the same amount of time. For example one method of defeating a timing attack on a naïve implementation of point multiplication is to insert dummy operations. That is, in each round of the double-and-add algorithm if only a doubling is to occur, we also perform an addition, but store the result in a register whose value we don't care about. Dummy operations are an important countermeasure when the operation being patched cannot make use of some form of mathematical structure to perform the operation in another way. Dummy operations can also be used in a more generic way. One can simply insert them randomly into the algorithm making it harder for an attacker to extract the true timing.

Notice however, that all hiding countermeasures against timing attacks have the following downside: the algorithm always performs in its worst case running-time. In fact many hiding countermeasures against other forms of attack also have this downside. For example, one could also randomly insert dummy operations to help prevent a power analysis

attack, but this would again lead to a slower algorithm.

One issue which we ignored in our discussion of power analysis is that for multiple measurements to be used together they need to be correctly aligned. That is, time t in one power trace might not correspond exactly to time t in another power trace (for example they might be misaligned by a matter of nanoseconds). The difficulty of aligning the power traces can be increased if the implementation executes certain instructions in an unpredictable order. This is another hiding countermeasure that can be used.

One of the most important hiding techniques is one which we already discussed above. By parallelizing parts of the algorithm and performing multiple computations at once, an implementer makes it much harder for an attacker to associate one operation with one physical output. This technique is particularly important if one of the operations contributes strongly to the physical output while being affected very little by the input key or message. The resultant noise will make it very difficult to abstract anything meaningful from the physical output. This technique is especially important since it also has the potential to speed up the computation of the algorithm, unlike most other hiding countermeasures.

When the cryptographic operation makes use of elliptic curves there are a number of extra hiding options available. In particular, by writing the curve in different forms (as opposed to the typical Weierstrass form) it is possible, for instance, to make doubling and addition require the exact same series of operations. In that case, the power traces for the two operations would be indistinguishable.

Masking Countermeasures

The most important type of masking countermeasure is the concept of blinding. Blinding works by randomizing the input to some part of the cryptographic algorithm in a reversible way. Consider, for example, the point multiplication xP in a group of order q. This could be computed by first selecting a random $r \in \mathbb{Z}_q$, computing $Q_1 = (x/2-r)P$, $Q_2 = (x/2+r)P$ and then adding Q_1+Q_2. Since $x/2-r$ and $x/2+r$ are random elements of $\mathbb{Z}_q$, this makes the physical output of the two point multiplications computed independent of the value of x. This is an example of additive blinding. Two other common types of blinding are multiplicative blinding, where the input value is multiplied by a random value, and boolean masking, where the input value is XORed with a random value.

Blinding is easier in the case where the function is linear with respect to the operation used for blinding. For example a multiplicative function f has the property that $f(ab) = f(a)f(b)$ and so is linear with respect to multiplication. In most public key settings, functions are not linear with respect to XOR and so additive or multiplicative blinding are used.

For elliptic curves some blinding methods make use of alternative coordinates as opposed to the Weierstrass form. In some cases these can be combined with the hiding methods for elliptic curves described above. This is a useful synergy since it is not always possible to combine countermeasures.

Blinding countermeasures may present a significant efficiency loss. Take for instance our example above for a method of blinding point multiplication. This requires two point multiplications and so approximately doubles the time it takes to perform the operation. There are ways to make blinding less expensive, but they generally require a tradeoff in security (for example, picking small random values).

A Balancing Act

One of the most alarming properties of countermeasures is that certain techniques might make other attacks easier or open the system to new attacks. This problem has existed since the early days of TEMPEST when it was discovered that attempts to soundproof a cipher room actually made acoustic side channel attacks when the microphone is in the same room as the machine much easier [30].

A more current example is an attack on schemes which use dummy operations. In particular, consider a situation where dummy operations are inserted into the double-and-add point multiplication algorithm. If a fault is induced into the device during the addition operation, this will have no effect on the final computation if it is a dummy operation. However, if it was a real operation, this will cause the output to be wrong (something which will eventually be detected). Thus by determining whether or not the output was valid the attacker can learn the key bit for that round.

These examples show that the selection of countermeasures represents a careful balancing act. Practitioners must make sure that in the process of trying to defend against one attack they did not make their scheme insecure against another. In some contexts this might mean they need to leave a scheme more vulnerable against a certain kind of attack to protect against another which is considered more dangerous in the threat model. In other cases, it may mean that implementing one countermeasure prevents them from implementing another. All the while they must also ensure that their solution is practical in the context in which they wish to deploy it.

1.2.6 What We Don't Know

There are no magic bullets when it comes to defending against side channel attacks – no one-size-fits-all cure. Defences must be carefully crafted both for a particular scheme and a particular hardware platform. It may even be the case that countermeasures need to be custom designed for a particular organization. Note that while there is no one formula for success, as with all areas of cryptography, there are a number of design principles that practitioners can use to guide themselves when implementing a scheme. Techniques like blinding and parallelization have proved themselves to be useful in many different scenarios.

Electromagnetic radiation attacks use many of the same techniques of power analysis and are theoretically stronger attacks. Further, it may be possible to perform them at a significant distance from a target device. At the very least, such attacks are much less intrusive (and so bypass many of the physical safeguards designed to prevent other attacks).

Yet, relatively little has been published about these attacks in the cryptographic literature. Seeing as these were the first attacks discovered by the intelligence community, it seems reasonable to assume that their knowledge about such techniques is relatively advanced.

In fact, this particular area of cryptography may be the one where the intelligence community is furtherest ahead of the public. Certainly they have known about and worked on side channel attacks for much longer than the public. Since practitioners cannot be sure how powerful of an attacker they are defending against, it is important they not take the threat of side channel attacks lightly. Another thing to consider is that for a number of reasons attacks in the literature use relatively low cost equipment. However, intelligence organizations might reasonably have access to much higher quality equipment allowing them to make even more precise measurements.

Another important issue is that almost all of the literature on side channel attacks focuses on key recovery. One reason for this is that defences tend to be responsive (that is, defences are presented for attacks that are already known, not against the possibility of an attack). Since attacks which recover the secret key are seen as stronger than attacks which, for example, allow an adversary to recover the plaintext, these are the attacks which are published. Because of the responsive nature of countermeasures, they also mostly cover key recovery attacks. Nonetheless, other forms of attacks present a very real threat. For example, notions of indistinguishability are much harder to achieve in the context of side channel attacks (see Section 2.2.7). Since such notions are at the core of the security of almost all encryption schemes, the inability to realize them in practice may have significant ramifications.

Chapter 2

Leakage Resilience

2.1 Leakage Resilience to the Rescue

In an attempt to unite theoretical cryptography with the practical truth of side channel attacks, Micali and Reyzin devised a theoretical model of cryptography that would cover all such attacks [57]. In particular, their goal was that if a scheme could be proved to be secure under their model, then it would also be secure against side channel attacks. Since showing a scheme to be secure against all such attacks regardless of the associated physical implementation seems to be an impossible task, they acknowledged that future works might need to make some assumptions about the security of the underlying implementation to obtain more meaningful results. Their paper does not contain an example of any cryptographic scheme secure in this new model.

It is important to note the central idea of this work is similar to the central idea of provable security in general. Namely, that if a scheme can be proved secure, then it is secure against *all* (side channel) attacks, even ones that have not yet been discovered. The major benefit of this notion is summarized well in a paper posted shortly after [57] (but not published until 2009). In reference to their similar framework, the authors of [68] say it "allows getting rid of most of the subjective parameters that were limiting previous specialized and often ad hoc approaches in the evaluation of physically observable devices."

Since then, many papers have been published which contain several types of provably secure leakage resilient schemes (signatures, encryption, IBE, etc.). These schemes come with a number of different models of the attacker's abilities. That is to say, in Micali and Reyzin's work they envisioned an attacker who had access to an extra source of information (related to some sort of physical observation). They made some very general restrictions on this leakage, but nonetheless it was still a very powerful adversarial notion. To actually prove their schemes secure, the authors of these papers significantly restricted the types of leakage that were allowed. The various models of leakage are outlined in the next few sections.

One of the implicit properties of the leakage functions we consider is that they are polytime computable. While it is not immediately obvious to us that physical phenomena

can be modelled in this way, Micali and Reyzin claim this is the case [57]. Additionally, Standaert et al. claim that not even the full power of polytime computable functions is needed and a weaker model suffices [69].

2.1.1 Computational vs. Memory Leakage

One of the axioms of Micali and Reyzin is that "only computation leaks information". Initially, all schemes were proved secure with this restriction. With the introduction of cold boot attacks, however, it was clear that the axiom did not hold in practice. Thus, many later papers have done away with this restriction. Nonetheless, it has been argued [26, 44] that under reasonable assumptions cold boot attacks can be modelled under a slight modification of the original axiom. Hence, some recent papers have gone back to the original model (or something close to it). In any case, a compelling argument for one or the other should be made. In our discussion of various schemes, we will add the tag "memory" to models that include memory leakage.

2.1.2 Bounded-Length, Bounded Entropy, Bounded Retrieval, Computationally Hard-to-Invert

Bounded-Length

The most important way in which papers contain the damage due to leakage is by restricting how much information it gives about the secret key. In the simplest model we restrict the length of the output of the leakage function. That is, for some number b we say that an attacker can learn at most b bits of secret information. Note that since there is (almost) no other restriction on what form the leakage takes, we must have that $b < |SK|$, otherwise the function could give away the entire secret key, SK. In fact we must have that $b \leq |SK| - k$, since otherwise we do not meet the requirements of the security parameter, k. More generally, we often refer to the fraction (in terms of secret key size) of bits that can be leaked. That is, each scheme in this model has an associated function f so that $0 \leq c = f(k)/|SK| < 1$. Here c is the proportion of key bits that can be leaked and is known as the leakage rate. We will refer to schemes using this model of leakage as "bounded-length" schemes.

A concrete, and somewhat realistic, example is to suppose a scheme has leakage rate $1/2 + \varepsilon$ for some $\varepsilon > 0$. Then supposing the secret key is 512 bits long, a function which reveals the 256 least significant bits of SK would be allowed here. More specifically, a timing attack which used 256 bits of timing information would also be allowed.

To better understand this notion we formalize the security of a signature scheme under this model (with memory leakage) with the definition below.

Definition 2.1.1 ([11])**.** A signature scheme is *existentially unforgeable under chosen message attacks in the memory bounded-length leakage model with leakage parameters*

$(L_G,\ L_S,\ L_M)$ if every PPT forger $\mathcal{F}$ succeeds at the following security game with at most non-negligible probability.

- $\mathcal{F}$ selects a leakage function f_G and sends it to the challenger who generates the public and private key (PK, SK) and sends them to $\mathcal{F}$ along with $f_G(SK, r_G)$, where r_G is the randomness used during key generation.
- $\mathcal{F}$ may perform a polynomial number of signature queries by providing a message m_i and a leakage function f_{S_i} to the challenger and receiving $f_{S_i}(SK, r_i)$ and a signature σ_i on m_i, where r_i is the randomness used during signing.
- $\mathcal{F}$ may make a polynomial number of leakage queries by providing a leakage function f_i to the challenger and getting back $f_i(SK)$ (these leakages correspond to memory attacks).
- $\mathcal{F}$ concludes the game by outputting a signature σ for the message m whose signature it has not requested.

$\mathcal{F}$ succeeds at the game if σ is a valid signature for m, $|f_G(SK, r_G)| < L_G|SK|$, $|f_{S_i}(SK, r_i)| < L_S|SK|$ for each i, and the total number of bits output from leakage functions is less than $L_M|SK|$.

Notice that if there is no leakage, this definition becomes the standard notion of signature security. A similar statement is true for the altered definition of security for encryption schemes. Unfortunately, a fundamental restriction in all leakage models for encryption schemes is that no leakage can occur after the challenge ciphertext has been issued (there are many ways to come up with a leakage query that will reveal which plaintext was encrypted). To combat this, Halevi and Lin [35] propose the notion of "entropic security" where queries are permitted after the challenge provided they leave it with enough entropy. We will discuss the difficulties of maintaining indistinguishability in a leaky environment in greater detail in Section 2.2.7.

Bounded Entropy

Instead of restricting the length of the output, one could instead ensure that even after seeing the leakage an attacker does not have enough information to determine the secret key. More formally, we require the (min-)entropy of the secret key, conditioned on the leakage to be relatively high. This is a useful extension of bounded-length leakage in that it both allows us to leak as much useless information as we want (we will see why this is actually useful soon) and it also models the fact that the attacker generally receives noisy information. We will refer to schemes using this model of leakage as "bounded entropy" schemes. See the full version (ePrint link) of [60] for more details.

Very few schemes actually prove their security in the bounded entropy model. In most cases, the scheme is proved secure in the bounded-length model (for ease of analysis) and

then the authors claim or argue that their scheme is in fact secure in this more general model. This is an unsurprising development, since many of the proofs of security make use of the fact that the key still has high min-entropy even after (bounded-length) leakage occurs.

As an example of this type of leakage, suppose a secret key is chosen uniformly at random from $\{0,1\}^{1024}$ and given the public key each possibility for SK is still equally likely. Now suppose that an attacker may learn any information about SK provided SK still has $n/8$ bits of entropy. Here a power analysis attack which reveals at most 896 bits of SK would be allowed (the situation is actually more complicated than this, see Section 2.2 for more details). This is because each of the remaining 128 unknown bits is still equally likely to be one or zero and so there are 2^{128} equally likely possibilities left for SK.

Bounded Retrieval

Another quite different approach is the following. Consider the case where you have an extremely large number of secret keys corresponding to one public key. In a particular instantiation of such a scheme the actual secret key consists of many of these secrets. However, a signature or decryption uses only a tiny portion of the secret key. Further, learning even a large number of the individual secrets leaves an attacker unable to compute any of the others. Here, for any bound b on the number of bits to be leaked, by choosing a secret key large enough we can get a secure scheme. This is known as the bounded retrieval model. Since the previous definitions of this model include memory attacks, we will not add the tag "memory" (despite allowing such attacks) to avoid confusion with definitions of the term in the literature.

For an example of an allowed leakage in the bounded retrieval model, suppose we wish to be secure against a cold boot attack that recovers 2^{30} bits (a gigabit) of information. Then for any scheme secure in this model, by creating a large enough amount of secret key information (in this case maybe several gigabytes), we can guarantee security against such cold boot attacks.

A drawback of the bounded retrieval model is that the standard notion of signature security is no longer applicable. In particular, since we generally assume that the number of bits that can be leaked in this scenario is very large, an attacker can simply request a valid signature for a message as a leakage query. To deal with this, Alwen et al. [3] propose "entropically unforgeable signatures." Basically, an attacker's forgery must be on a message chosen from a set of messages with high enough entropy[1]. This choice of message must be made after all leakage queries have occurred. Halevi and Lin's method of allowing leakage after seeing the challenge ciphertext can be thought of as the analogue of this approach in the context of encryption schemes.

[1]Note that both the particular message and the set of messages are chosen by the attacker

Computationally Hard-to-Invert

The auxiliary input model was originally proposed by Dodis et al. in [20]. While this work focused on symmetric key schemes, in [17] Dodis et al. deal with the model in the context of public key schemes. The idea here is that we should be able to allow all leakage functions whose output leaves the secret key hard to compute (even if it is information theoretically revealed). In particular, if $(PK,\ SK)$ is a public-key/secret-key pair, then a function f is allowed if even given $f(PK, SK)$, no polynomial time algorithm can recover SK with non-negligible probability. As it turns out, this model is often too strong. Thus, in the slightly weaker model, it must be hard to compute SK given PK and $f(PK, SK)$. While this seems like a natural restriction, there may be issues with this. For a more in-depth discussion of these issues see Section 2.2.5. We will refer to this type of leakage as "computationally hard-to-invert".

It seems very difficult to come up with a concrete and practical example of a function allowed in the computationally hard-to-invert model. While the model generalizes the bounded-length and bounded entropy models, and so the examples there would also work here, they do not show the greater power of the model.

2.1.3 One-time vs. Continual Leakage

Another restriction which we implicitly used in the discussion of leakage bounds in Section 2.1.2 was that the leakage only happens once. That is to say, in, for example, the bounded-length model, after b bits have been leaked, no more leakage can occur. (These b bits, however, can be collected over time and adaptively, if required.) However, more recently schemes have looked at the case where only the leakage in a particular time frame is bounded. They do this by adding key updates to their scheme. Thus, they restrict the amount (or type) of leakage that can occur between updates. We denote schemes in this model by adding the tag "continual."

As an example we present the formal definition of security for a signature scheme in the continual memory bounded-length leakage model, i.e., the continual version of Definition 2.1.1.

Definition 2.1.2 ([11])**.** A signature scheme is *existentially unforgeable under chosen message attacks in the continual memory bounded-length leakage model with leakage parameters* $(L_G,\ L_U,\ L_S,\ L_M)$ if every PPT forger $\mathcal{F}$ succeeds at the following security game with at most non-negligible probability.

- $\mathcal{F}$ selects a leakage function f_G and sends it to the challenger who generates the public and private key (PK, SK_0) and sends PK to $\mathcal{F}$ along with $f_G(SK_0, r_G)$, where r_G is the randomness used during key generation.
- $\mathcal{F}$ may perform a polynomial number of signature queries by providing a message m_i and a leakage function f_{S_i} to the challenger and receiving $f_{S_i}(SK_j, r_{S_i})$ and a signature σ_i on m_i, where r_{S_i} is the randomness used during signing.

- $\mathcal{F}$ may make a polynomial number of leakage queries by providing a leakage function f_i to the challenger and getting back $f_i(SK_j)$ (these leakages correspond to memory attacks).
- $\mathcal{F}$ may perform a polynomial number of update queries by sending a leakage function f_U to the challenger who updates SK_j to SK_{j+1} and sends $f_U(SK_j, r_{U_j})$ to $\mathcal{F}$, where r_{U_j} is the randomness used during key updates.
- $\mathcal{F}$ concludes the game by outputting a signature σ for the message m whose signature it has not requested.

$\mathcal{F}$ succeeds at the game if σ is a valid signature for m, $|f_G(SK_0, r_G)| < L_G|SK_0|$, $|f_{S_i}(SK_j, r_{S_i})| < L_S|SK_j|$ for each i, $|f_{U_j}(SK_j, r_{U_j})| < L_{U_j}|SK_j|$ for each j, and the total number of bits output from leakage functions between updates is less than $L_M|SK_j|$. Notice that the leakage during key updates counts towards the master leakage in both the period where SK_j is the secret key and where SK_{j+1} is the secret key.

2.1.4 Putting it All Together

Here we provide a brief overview of results in each of the models. In the bounded-length model an (identity-based) encryption scheme has three leakage parameters each expressed as a fraction of the length of the secret key: leakage during key generation, leakage during decryption, and master leakage (the overall amount of leakage that can occur). We express this as the triple (L_G, L_D, L_M). A signature scheme has similar parameters, except we replace L_D with L_S, the leakage during signing. Continual schemes have an additional parameter L_U, the leakage during key updates. In this case, L_M refers to the amount of leakage that can occur between key updates.

On top of the results we present below, a number of general purpose compilers have been proposed [27, 28, 41]. These convert any standard (cryptographic) algorithm into one that is leakage resilient. All rely on the availability of some form of leak-proof hardware.

Throughout this discussion, k is the security parameter, n is the message length (in bits), and ℓ is a parameter affecting the secret key size. In many cases, the secret key will consist of $O(\ell)$ group elements. Furthermore, we will say that a scheme has a *global* leakage parameter if all of its associated leakage parameters are equal (e.g. in a signature scheme $L_G = L_S = L_M$). Also, unless otherwise noted, encryption schemes achieve chosen-plaintext security. Further, we assume that for constructions which make use of groups of prime order q, we have $|q| \approx 2k$.

Bounded Retrieval Schemes

Alwen et al. [3] construct an identification (ID) scheme, an authenticated key agreement (AKA) scheme, and a signature scheme. The signature scheme is entropically secure in the bounded retrieval model. Their AKA and signature schemes rely on random oracles

for their security proofs. Their ID and signature schemes are secure under the gap Diffie-Hellman assumption[2] and the AKA scheme is secure under DDH. It is assumed that no leakage occurs during key generation.

Alwen et al. [2], in their 2009 paper show how to construct public key and identity-based encryption schemes in this model. To accomplish this they propose an identity-based hash proof system. They present three constructions of this primitive using bilinear groups, lattices, and the quadratic residuosity assumption. In each case no leakage is allowed during key generation. They also show that their IBE scheme can achieve CCA security and construct a generic transformation from CPA to CCA security for standard encryption schemes in the bounded retrieval model.

Memory Bounded-Length Schemes

In 2009, Akavia et al. [1] describe an encryption scheme and an IBE scheme that globally tolerate at most a $1/\log n$ and $1/\log^2 n$ fraction of leakage, respectively, under the learning with errors lattice assumption.

Also in 2009, Katz and Vaikuntanathan [43] propose a few different signatures schemes. One globally tolerates a $1 - |SK|^{\varepsilon}$ fraction of leakage assuming the existence of a universal one-way hash function mapping $|SK|$ bits to $|SK|^{\varepsilon}$ bits. Unfortunately, due to its reliance on an unbounded simulation sound non-interactive zero-knowledge proof system, the scheme is fairly inefficient. They also propose two different one-time schemes which they show can be extended to t-time schemes that have global leakage parameter $1/t$.

Another scheme independently discovered in an early version of [42] by Katz and by Alwen et al. [3] is a modification of the Schnorr signature scheme. Chapter 3 provides a detailed discussion of this scheme. It globally tolerates a $\frac{1}{2} - \frac{1}{2\ell} - \varepsilon$ fraction of leakage for any $\varepsilon > 0$ under the discrete log and random oracle assumptions.

Still in 2009, Naor and Segev [60] propose several different encryption schemes. First of all, they show a general construction based on hash proof systems. They proceed to give specific examples, including showing that the scheme of Boneh et al. [9], BHHO, is secure even after a $1 - \frac{2(2k+\log(1/\varepsilon))}{|SK|}$ fraction of global leakage occurs, where ε is a negligible function of k. This scheme appears in a number of papers on leakage resilience and is the subject of further discussion in Section 2.2.6. They also present a modification of the scheme which tolerates a $(\ell - 2 - \log_q 1/\varepsilon)/\ell$ leakage rate. They propose a CCA-1 secure scheme based on Cramer-Shoup "lite" with a global leakage rate of $1 - \frac{\omega(\log k)-n}{4|SK|}$, where ω is the usual "little omega" from complexity theory. Note that the only way to increase the number of bits that can be leaked in this scheme is to increase the size of the underlying group. A similar scheme based on Cramer-Shoup achieves CCA-2 security with a global leakage rate of $1 - \frac{\omega(\log k)-n}{6|SK|}$. The three above schemes rely on the hardness of DDH.

In their 2010 paper, Chow et al. [15] present a number of identity-based encryption schemes. All of their schemes are close in efficiency to the previously known (non-leakage-resilient) schemes they are based on. They present two schemes that have global leakage

[2]Here we assume that DDH is easy, but CDH is hard.

parameter $1/3-\varepsilon$ based upon the decisional bilinear Diffie-Hellman assumption and another which achieves $1/9 - \varepsilon$ using assumptions based on the generalized subgroup decision assumption.

In 2010, Dodis et al. [19] propose a new primitive: true-simulation extractable non-interactive zero-knowledge arguments. Using this and a leakage resilient key generation algorithm, they devise generic constructions of a signature scheme and a CCA-secure encryption scheme. They give specific instantiations of each under the symmetric external Diffie-Helman assumption[3] (SXDH) and the decisional linear assumption[4] (DLIN). The signature scheme has global leakage rate $1-\varepsilon$ where the signature length is approximately $18/\varepsilon+24$ group elements under SXDH and $57/\varepsilon+70$ group elements under DLIN. Similarly, the encryption scheme has global leakage rate $1 - \varepsilon$, where the ciphertext size is approximately $\frac{2}{\varepsilon(2+1/2)} + 15$ group elements under SXDH and $\frac{3}{\varepsilon(3+1/2)} + 34$ group elements under DLIN. Using their signature and encryption schemes they also construct identification and authenticated key agreement protocols.

In 2011, Halevi and Lin [35] construct entropically secure encryption schemes based on the generic constructions of Naor and Segev [60]. The amount of leakage tolerated depends on the specific instantiation of the tools they use (a strong extractor and a hash proof system).

Notice that the schemes of Dodis et al. require a rather large number of group elements even when the fraction of leakage is very small. Unsurprisingly, the operations required to compute these elements make the schemes impractically inefficient. In fact, of the schemes mentioned, only the IBE schemes of Chow et al. and the modifications of Schnorr and BHHO are reasonably close to being practical. Furthermore, even the fastest known IBE schemes are significantly slower than public key encryption schemes used in practice.

Continual Bounded-Length Schemes

In 2010, Faust et al. [26] showed how to convert a 3-time signature scheme that allows for up to L *bits* of leakage into a continually secure scheme that allows $L/3$ bits of leakage per signature. However, their scheme needs access to truly random bits and its efficiency depends (logarithmically) on the number of messages that can be signed in the lifetime of the system.

In 2010, Kiltz and Pietzrak [44] prove the CCA security of a blinded variant of ElGamal encryption in the generic group and continual bounded-length models. They further require that the scheme is instantiated over bilinear groups of prime order p, where $p - 1$ is not smooth (i.e. it has a large prime factor). The scheme has global leakage parameter $1/2-(3\log q+k/2)/|SK|$ where q is the number of queries an attacker can make. They go on to conjecture that a practical variant of their scheme is secure even for arbitrary groups as long as $p - 1$ is not smooth.

[3]This assumption is that DDH is hard in both of the "input" groups to a bilinear asymmetric pairing.
[4]A slight generalization of DDH in the input group of a symmetric pairing.

Recall that blinding is one of the well-known countermeasures for several types of side-channel attacks. Furthermore, ElGamal encryption in the form of ECIES is a scheme that is used in practice. So, it is interesting to see an attempt to prove the security of a countermeasure in this model. Unfortunately, they were unable to prove the security of their practical scheme. Nonetheless, we feel that the idea of proving the security of known countermeasures presents a good compromise between theory and practice. Also, it would be interesting to compare the efficiency of their practical scheme with one where blinding occurs at the implementation level.

Continual Memory Bounded-Length Schemes

In the discussion for schemes in this model, the notation log as a leakage parameter means that the scheme can tolerate $O(\log k)$ *bits* of leakage during that phase. In every case where this appears, the schemes make use of a lemma we will discuss in detail later. For more information see Lemma 4.6.1.

In 2010, Dodis et al. [18] construct the primitive of a continuous leakage resilient one-way relation. Using this they construct a signature scheme in the random oracle model, that under the d-linear assumption has leakage parameters $(\log, \log, \frac{1}{2d+2} - \varepsilon, \frac{1}{2d+2} - \varepsilon)$, where $\varepsilon > 0$ is a constant. They also construct ID schemes in the standard model, and AKA and "challenge-response" interactive signatures in the random oracle model. The efficiency of all these schemes depends on the number of queries an attacker can make between updates.

In 2010, Brakerski et al. [11] propose the first public key encryption scheme in this model. Its leakage parameters are $(\log, \log, \frac{\ell-6-\gamma}{2\ell}, \frac{\ell-6-\gamma}{2\ell})$ under DLIN and $(\log, \log, \frac{\ell-2-\gamma}{\ell}, \frac{\ell-2-\gamma}{\ell})$ under SXDH, for $\gamma > 0$. They also construct an IBE scheme with similar leakage parameters under DLIN. Additionally, they create a signature scheme that makes use of a leakage resilient encryption scheme (actually, it only uses the key generation and update facilities of such a scheme). The leakage parameters of the resulting scheme are $(L_G, L_U, L_M/2k, L_M)$ where the leakage resilient scheme has parameters (L_G, L_U, L_M, L_M). However, every signature of this scheme also leaks $2ks$ bits to the attacker, where s is a parameter that affects the efficiency of the scheme. The signature scheme relies on a primitive for which constructions are only known in the random oracle model.

In 2011, Lewko et al. [51] construct IBE, hierarchical IBE, and attribute-based encryption schemes that rely on three assumptions on composite order bilinear groups. All have leakage parameters approximately $(0, 0, 1-\varepsilon, 1-\varepsilon)$, where ε is a small positive value that shrinks as ℓ increases. In the HIBE scheme, the amount of leakage allowed depends on height of the key in the hierarchy (higher keys can leak less).

In 2011, Lewko et al. [50] construct the first schemes that can tolerate more than a logarithmic amount of leakage per update in this model. They construct an encryption scheme and a signature scheme under three assumptions based on the generalized subgroup decision assumption. Both schemes have security parameters $(0, L, L, L)$ where $L \approx 1/162$. Note that at the 128-bit security level, their scheme needs a key size of at least 2048 bits to allow more leakage during key updates than previous schemes.

In 2011, Malkin et al. [52] construct a signature scheme secure under the SXDH assumption. The scheme has leakage parameters $(\log, \log, 1 - \frac{2+\gamma}{\ell}, 1 - \frac{2+\gamma}{\ell})$ where $\gamma > 0$ is a constant. We will present this scheme in detail in Chapter 4.

In 2011, Boyle et al. [10] construct signature schemes in both the memory bounded-length leakage and continual memory bounded-length leakage model. Their continual scheme is secure under the d-linear assumption with leakage parameters $(0, 0, 1/d - \varepsilon, 1/d - \varepsilon)$.

The only schemes that have close to practical efficiency are those of Boyle et al. and Malkin et al. Our discussion in Chapter 4 of the scheme of Malkin et al. contains a detailed efficiency analysis.

Memory Computationally Hard-to-Invert Schemes

In 2010, Dodis et al. [17] proposed several encryption schemes in the computationally hard-to-invert model. In particular, they show that BHHO is secure (in their strong notion of security) against leakages that leave any polynomial time attacker with a success probability of at most $2^{-(8k)^{\varepsilon}}$ where the secret key has size $(8k)^{1/\varepsilon}$.

For ease of comparison we have compiled Table 2.1. In the table, "type" refers to the type of scheme and "leakage notes" contain leakage parameters or restrictions.

2.2 A Closer Look

At a first glance, leakage resilient cryptography seems tremendously useful; finally we can account for side channel attacks in a theoretical model of security. However, it is unclear how well the proposed solutions actually model realistic attacks. Thus, a fair question to ask at this point is "Are leakage resilient schemes really secure against side channel attacks?" The answer is unfortunately "no": none of the models above cover *all* classes of practical side channel attacks. In particular, any side channel attack which reveals the entire secret key (or reduces the search space to a small enough size) is not allowed under any of the above models. Further, it is impossible to construct a scheme that could be secure against such an attack. This may seem to be incredibly thin criticism at first, since of course some information must remain secret in cryptography. Nonetheless, these schemes provide us with no assurance that such devastating leakage will not occur. Moreover, they go no further in preventing it than schemes used in practice. This is in stark contrast to other countermeasures against side channel attacks, which often prevent or severely limit a large class of practical attacks.

The problem that arises here is that side channel attacks happen at the implementation level. That is to say if someone were to perform a power analysis attack on the Schnorr signature scheme, they would really be attacking the particular implementation[5] of elliptic

[5]There are examples of scheme-level defences like some instances of blinding, but for the most part countermeasures are low-level.

Authors	Year	Security Model	Type	Practical	Leakage notes
Alwen et al. [3]	2009	BR	Sig, ID, AKA	No	None during keygen
Alwen et al. [2]	2009	BR	Enc, IBE	No	None during keygen
Akavia et al. [1]	2009	MBL	Enc, IBE	No	$1/\log n$
Katz, Vaikunanathan [43]	2009	MBL	Sig	No	$1 - \lvert SK\rvert^{\epsilon}$
Alwen et al., Katz [3, 42]	2009	MBL	Sig	Maybe	$< 1/2$
Naor, Segev [60]	2009	MBL	Enc	Maybe	$1 - o(1)$
Chow et al. [15]	2010	MBL	IBE	Maybe	$< 1/3$
Dodis et al. [19]	2010	MBL	Enc, Sig	No	$1 - \varepsilon$
Halevi, Lin [35]	2011	MBL	Enc	No	Entropic security
Faust et al. [26]	2010	CBL	Sig	No	1/3 of underlying scheme
Kiltz, Pietzrak [44]	2010	CBL	Enc	Yes/No	No proof for practical scheme
Dodis et al. [18]	2010	CMBL	Sig, ID, AKA	No	$1/4 - \varepsilon$
Brakerski et al. [11]	2010	CMBL	Enc, Sig, IBE	No	$1 - o(1)$
Lewko et al. [51]	2011	CMBL	(H)IBE, ABE	No	$1 - o(1)$
Lewko et al. [50]	2011	CMBL	Enc, Sig	No	1/162
Malkin et al. [52]	2011	CMBL	Sig	Maybe	$1 - o(1)$
Boyle et al. [10]	2011	CMBL	Sig	Maybe	$1 - o(1)$
Dodis et al. [17]	2010	CHTI	Enc	No	$2^{-(8k)^{\varepsilon}}$

Table 2.1: Comparison of leakage resilient schemes.

BR is bounded retrieval, MBL is memory bounded leakage, CBL is continual bounded leakage, CMBL is continual memory bounded leakage and CHTI is computationally hard-to-invert. Formulas in the leakage column represent the largest of the parameters with the strongest assumptions. Schemes marked "maybe practical" have instantiations requiring only a few more operations than practical schemes.

curve point multiplication that was used and not the scheme itself. Brakerski et al. [11] referring to the previous lack of an encryption scheme secure in the continual memory bounded-length model, subtitled their paper "Overcoming the hole in the bucket." To steal the bucket metaphor, imagine a cryptographic scheme as a bucket full of water (the secret key) and the implementation of that scheme as a lid. As it stands, the quality of the lid varies and so when we turn the bucket upside down a lot of the water may leak out. Leakage resilient cryptography seeks to solve that problem by building a bigger bucket (at the same cost hopefully). Thus, even if some of the water leaks out, we still have enough left for our purposes. But, if the lid is not secured, at the end of they day, there still won't be any water left no matter how big our bucket is. Further, in some cases, the new "leakage resilent" buckets require custom lids which have yet to be invented (and may be harder to build).

More formally, the approach of (public key) schemes in leakage resilient cryptography, since they are above the level at which the leakage is occurring, is to make sure that even if some leakage occurs, the secret is still safe. There are several drawbacks to this approach. First of all, since it is not the scheme itself that is leaking, we have no idea just by looking at it how much it will leak. Secondly, there are very few known attacks against currently deployed schemes where partial key leakage leads to an algorithm faster than the best known generic algorithm on this smaller key space.[6] That is to say, in some ways these new schemes are only more secure in the sense that they have larger keys (and so when a fixed amount of leakage occurs the remaining key space is larger in the new schemes). This comes with the added downside that in many cases, for practical security levels and leakage bounds, currently deployed schemes with increased key sizes are faster than these new schemes. Finally, and perhaps most crucially, one is still left with the need to create secure implementations of these schemes. In particular, a practitioner still needs to add all of the countermeasures he would have previously (otherwise the schemes are still vulnerable to the side channel attacks which fall outside their security models). Unfortunately, the new schemes are often much more complex than practical schemes and may use operations that we have little experience in developing countermeasures for. Consequently, it may[7] be much harder to implement these schemes securely.

Certainly, an argument could be made that it is useful to be able to prove security given some assurances that the implementation is not too leaky. However for such proofs to be worthwhile, we would need evidence that there is a non-leaky implementation of the scheme. None of the papers we examined provided such evidence.

At this point it is important to note that we are neither criticizing the motivation of leakage resilient cryptography nor the quality or creativity of the work in the field. Instead, we are concerned that the current direction of the field will not contribute anything to achieving the original goal: creating cryptosystems secure against side channel attacks.

[6] Only attacks on factoring are known – see [36] for one such attack.

[7] We are forced to concede that since no one has attempted to implement these schemes, there is some possibility that they are not harder to secure.

2.2.1 Leakage Leakage Leakage Leakage

The heading here is not a typo; in fact, it is a quote from the 2011 Crypto Rump session panel on leakage.[8] It also nicely summarizes our sentiment that getting too caught up in leakage (resilience) can lead one to forget its *raison d'être*. In this sense, we feel the name change away from "physically observable cryptography" is a bad one. While that name has it issues (there are classes of side channel attacks that do not correspond to physical observations), it keeps the focus pretty clearly on security against side channel attacks. When that focus slips away, we get results like that of Dziembowski and Pietzrak who created a (provably) leakage resilient stream cipher [23] which was subsequently shown to be less secure against DPA attacks than an unprotected implementation of an AES-based cipher [67].

Related to this problem is that many papers refer to the fact that "in practice" the standard cryptographic assumptions made about the attack model do not hold in the presence of side channel attacks. Nonetheless, their authors fail to address how their schemes fit into a practical cryptographic setting – neither in terms of its abilities to withstand side channel attacks, nor in terms of its efficiency. With respect to efficiency, many schemes in this model are labelled by their designers as "efficient", whereas a practitioner might regard them as "horribly inefficient." In some cases, this is because in the authors' minds there seems to be no distinction between an "efficient" algorithm and a polynomial time algorithm. In other cases an "efficient scheme" means that previous schemes were even more inefficient.

Given the practical nature of the problem leakage resilience attempts to solve, we feel that authors should expend greater effort to place their schemes in a practical setting. They should do this by analyzing both how well their scheme might fare against realistic side channel attacks and the scheme's efficiency (in practical terms, not just with big O notation). Even mentioning that the scheme is impractical and a short discussion of what could be done to address this would be a step forward for the field.

Beyond this, consider this quote from Brakerski et al. on page 5 of [11]:

> We note that the main drawback of our [signature] schemes is that the leakage-rate that we can tolerate is not optimal. For example, we can only tolerate a leakage-rate of $1/4 - \varepsilon$ between each update procedure. We don't view this as a serious drawback, since it simply means that we should update our secret key more often.

Their main signature scheme is totally inefficient (it requires $O(k^2)$ computations), but its "main drawback" is that its leakage rate is "not optimal." In fact, not once do they comment on the efficiency of any of their schemes, despite the fact that all of them are impractical. This is alarming to us since the motivation for their work is that the assumptions of (provable) public key cryptography do "not hold in practice" (p. 1). On top of

[8] Attributable to any of the three panel members as well as the moderator, since the only word said for those two or so minutes was "leakage." A video of the presentation is available at `http://www.youtube.com/watch?v=i-3tfQ9EQQw`. See also the follow-up paper [6].

this, the second part of their quote dramatically oversimplifies the issue as their scheme only allows for a tiny bit of leakage during updates (and so frequently needing to update means making a strong assumption).

Now, consider the following quote about the bounded leakage and bounded retrieval models from Alwen et al. on page 1 of [3]:

> In many situations, the attacker might get some partial information about secret keys through means which were not anticipated by the designer of the system and, correspondingly, not taken into account when arguing its security. Such attacks, referred to as key-leakage attacks, come in a large variety. For example, this includes side-channel attacks [46, 8, 7, 47, 65, 31], where an adversary observes some "physical output" of a computation (radiation, power, temperature, running time etc.) in addition to the "logical output" of the computation. Alternatively, this also includes the "cold-boot" attack of Halderman et al. [34], where an adversary can learn (imperfect) information about memory contents even after a machine is powered down. Lastly, this can include various malware/virus/hacking attacks where the adversary can download arbitrary information from an attacked computer.
>
> ...
>
> In this work, we assume that the attacker can repeatedly and adaptively learn arbitrary functions of the secret key sk, as long as the total number of bits leaked during the lifetime of the system is bounded by some parameter ℓ. Due to its generality, this model seems to include a very large class of attacks mentioned above, and has recently attracted a lot of attention from the research community.

First of all, the authors seem to ignore the fact that any system not secured at the implementation level is vulnerable to attack. Next, as we argue in Section 2.2.2, the bounded-length model might not even be able to account for many of the side channel attacks they mention. Finally, we find it disingenuous to include malware, viruses, and hacking in the list of attacks as these typically result in complete system compromises, in which case no scheme (leakage resilient or otherwise) is helpful.

That last point is part of a broader trend that we find particularly worrying. Many authors feel the need to exaggerate the utility of the previous works in the field. Koblitz and Menezes make this point as well:

> Although an implementer would search in vain for anything of practical use in [57] or [68], among theoreticians [57] and [68] are considered landmark papers. In [42, 43] we read:
>
> > In the past few years, cryptographers have made tremendous progress toward modeling security in the face of such information leakage [57, 68], and in constructing *leakage-resilient* cryptosystems secure even in case such leakage occurs.

> This use of the words "tremendous progress" to refer to the philosophical essays [57, 68] seems to us to be a bit of hyperbole [45, p. 21].

Dodis et al. say,

> [A] new goal has been set within the theory of cryptography community to build general theories of physical security against large classes of side channel attacks. A large body of work has accumulated by now in which different classes of side channel attacks have been defined and different cryptographic primitives have been designed to provably withstand these attacks [17, p. 1].

We take issue with equating classes of leakage with classes of side channel attacks when there is much work to be done on understanding how closely related these two things are. Additionally, this quote seems to be moving the definition of side channel attacks away from its practical and realistic roots to one of theoretical interest only.

The same new view of side channel attacks is expressed in the abstract of [10] by Boyle et al. who state,

> A signature scheme is *fully leakage resilient* (Katz and Vaikuntanathan, ASIACRYPT '09) if it is existentially unforgeable under an adaptive chosen-message attack even in a setting where an adversary may obtain bounded (yet arbitrary) leakage information on *all intermediate values that are used throughout the lifetime of the system*. This is a strong and meaningful notion of security that captures a wide range of side-channel attacks.

On top of this, one of our biggest points of contention, and one which arises repeatedly in this thesis, is that it is very hard to appreciate what these security guarantees mean. So we are not sure in what way Boyle et al. find the notion "meaningful."

In his thesis[9], Wichs provides another example of this phenomenon:

> Leakage-resilience in [the bounded leakage] model is a property of only the algorithmic description of a cryptosystem and not its implementation. If a cryptosystem is shown to be resilient to bounded leakage, then *any* implementation of it on *any* hardware architecture is resilient [71, p. 12] (emphasis in original).

Once again, this disregards that an insecure implementation will leave the scheme totally vulnerable no matter how "leakage resilient" it is.

Overstating the results of the field makes it harder to understand the important question of which definition of leakage is best and undermines areas of research that are critical to the success of leakage resilience. In particular, we find the design of schemes which can tolerate some leakage and are competitive with practical schemes in terms of efficiency and in terms of security of their implementations to be the fundamental problem of leakage resilient cryptography. It seems pointless to solve a problem that only makes sense in a practical context with a theoretical solution.

[9] Wichs's thesis provides an excellent overview of the field in general as well as touching upon a number of important issues which are largely ignored in other works.

Unfortunately, any scheme faces the following efficiency loss in the presence of leakage. To account for leakage we must make the secret key larger than the minimum dictated by the security parameter. For example, a secret key for the Schnorr signature scheme at the 128-bit security level is 256 bits. If we leak even 1 bit of the key, then we no longer have 128 bits of security. Thus, we have to increase the key size by the number of bits we expect to leak. This in turn makes our entire scheme slower. Nonetheless, since current schemes do not have their parameters inflated to deal with such issues, the new scheme may still be a useful notion.

The above criticisms hold even assuming that the various classes of leakage functions actually do model a large class of realistic side channel attacks. However, the assumptions made on the types of leakage may prevent us from successfully modelling side channel attacks. In the next few sections we will look more closely at these restricting assumptions.

2.2.2 Bounded-Length

The bounded-length leakage setting faces the fundamental problem that many attacks (e.g. power analysis attacks) simply require too much data to be covered by the model. More specifically, in any of these schemes, the size of the secret key provides an absolute upper bound on the number of bits that can be leaked. However, a single power trace contains megabits of information and so violates that bound.

This example, however, sidesteps a subtle issue with the definition. How do we deal with the issue of an attacker learning compressed data? As it turns out, since secret information is random, anything that divulges that secret information cannot be compressed much. In particular if the attacker learns a string x that can be (losslessly) compressed to a b-bit string, then we can say that the attacker has learned b bits of information about the secret.[10] In any case, these schemes are actually secure against any leakage that is efficiently compressible to sizes within their leakage bounds. Unfortunately, the output of many practical side channel attacks (e.g. power analysis, timing, cold boot) is not efficiently compressible.

To mitigate concerns over many attacks not being allowed, Kiltz and Pieztrak argue it is only the output of the side channel attack that matters [44]. In general, this output is some number of key bits. Then, provided this number is small enough, this leakage would be allowed by the model. This is dissatisfying for a number of reasons. First of all, the whole point of having provably secure schemes is to avoid having to make intuitive arguments like this. Next, how can one tell how many key bits a given side channel attack will reveal? Given our earlier analysis, we can see that determining this is impossible just by looking at the scheme. Thus, further experimentation is required, beyond what is presented in any of the papers, just to be able to even remotely understand what their security guarantees are. Finally, if we are only considering the outcome of these side channel attacks, why not simplify the model by having the leakage just be some number of key bits?

[10] On average, the best a compression algorithm can do on random strings is to change the string size by 0 bits (it may increase the string size in some instances and decrease it in others).

This last point actually deserves further discussion. The idea has been the subject of research in the past. Dodis et al. put forward the idea of exposure resilient cryptography [12, 16, 22], where a scheme is secure even after an attacker learns a subset of the key bits. Wichs criticizes the notion in his thesis: "exposure-resilient cryptography may be overly restrictive in the type of leakage, by only allowing an attacker to probe individual bits of the state, but not allowing her to learn any global properties" [71, p. 7]. As an example, he says that Hamming weight is often used as part of DPA attacks. Nonetheless, we propose the following situation. Suppose an attacker wishes to break a public key scheme and is given some function $f(SK)$ of the secret key with $|f(SK)| < |SK|$ as extra information. Is there ever a situation in which she would not want f to reveal some number of bits of SK as opposed to some other function? In particular, is there a realistic scheme where learning some function of the secret key of length b is ever better than learning b bits of the secret key? Basically, our issue with Wichs's example of the Hamming weight, is that this is then used in conjunction with other information. Our question is, when looking at that information as a whole, would we prefer it over bits of the secret key? If not, why bother with the additional complexity of the bounded-length model? Is it possible that stronger results can be proved in this slightly weaker[11] model?

Even supposing that a single measurement associated with a given attack is acceptable under the model, it does not account for the fact that attacks like DPA involve repeating a measurement thousands of times. For example, while most schemes will allow leaking the Hamming weight of a per-message secret once, leaking this value every time a message is signed, say, would eventually violate the leakage bound. This is a fundamental weakness of the model, since we expect a scheme that leaks to leak some amount during each execution and so the overall amount it leaks is unbounded.

Continuity

The issue of not allowing repeated measurements is exactly the problem that is solved by making schemes in the bounded model continual. While these schemes run into the same problems of not necessarily being able to allow a single measurement, they do allow for a potentially unbounded amount of leakage. They do this by adding in a mechanism to update the secret key (while leaving the public key unchanged). These sort of primitives can be used to defeat attacks like DPA, by updating the key before too much information is gathered (as long as a single measurement does not contain too much information).

The natural question to ask then is "When do we update the key?" This is a tricky question and depends on a number of factors.[12] Most notably, "what are the assumptions made about leakage during updates?" and "how fast are the updates?" If key updates are significantly faster than the main operation of the scheme (signing, for a signature scheme), then we can perform the update every time we perform this operation without having a significant impact on efficiency. However, if we assume that not many bits (relative to the master leakage) can leak during updates (as several schemes in this model do), then this

[11] Although, if indeed directly leaking bits of the secret key is always best, it is not at all weaker.

[12] Unsurprisingly, it is largely unaddressed by authors of these schemes.

significantly weakens the security of our scheme. In some cases, depending on the update procedure, this may indeed be a valid assumption. Nonetheless, this analysis does not appear in any of the papers that use this model.

Another issue is that by looking at Definition 2.1.2, we can see that the security model implicitly assumes the existence of secure erasures. That is to say, it assumes that after updating the secret key all knowledge of that secret and any per-message secrets since the last update are erased (via a leak-proof operation). It is currently unknown how to (efficiently) achieve such a goal and as such it is a very strong primitive. Note that the model does indeed require secure erasures since otherwise an attacker could (over time) request any amount of information about an old secret key and so eventually recover it. This would be a total break of the scheme, since by learning any of the secret keys associated with the public key, an attacker could forge signatures. This criticism also applies to encryption schemes in this model, since they have a similar definition of security.

2.2.3 Bounded Entropy

Restricting leakage by bounding remaining entropy rather than length potentially enables us to actually allow attacks like power analysis. Recall that the problem before is that a single power trace may contain megabits of data – much more than is allowed to be leaked in bounded-length schemes. Despite this, the trace itself might not contain very much information about the key. By (lower) bounding the amount of entropy the secret key has (in the view of the attacker) after the leakage is observed, we can account for this possibility. In a certain sense, bounded entropy allows us to leak as much useless information as we want. This is significant, since the length of the information an attacker learns is much less important than its quality.

The entropy model also more accurately portrays the type of information an attacker learns. For example, if an attacker performs a cold boot attack, she does not actually learn secret bits. Instead she learns noisy values of those bits. The more noisy the values, the more entropy the secret key retains. Thus, we can see that the entropy model more closely matches the reality of side channel attacks.

Unfortunately, this model still has a major drawback: there is basically no way to tell whether a given leakage is actually allowed under the model. Thus, it is nearly impossible to understand the security guarantee provided by a scheme in this model. We re-explore this issue in our discussion of the computationally hard-to-invert model below.

2.2.4 Bounded Retrieval

The bounded retrieval model is weak against the following type of attack. Suppose Alice and Bob wish to use an encryption scheme that is secure under this model. When Alice proceeds to decrypt Bob's message, Eve performs a side channel attack and recovers the part of the secret key used for that exchange (and the plaintext as well). Notice that one particular round of encryption is not guaranteed to be any more secure than that of a

standard scheme. Thus, the security of the scheme depends heavily on the security of its implementation. While this is true in other models, in this case we have no guarantee of any kind that any given round is secure.

Nonetheless, the model offers several benefits. First of all, side channel attacks often require physical access to the device in question. Since recovering the secret used in one exchange does not help us decrypt future exchanges, fully breaking this scheme may require long-term physical access. Secondly, it may be very expensive to carry out a particular side channel attack and in this case a successful attack would not completely break the scheme. Finally, if the attacker cannot control which subset of keys will be used in each exchange (i.e. she is passive), then this makes attacks like DPA, which build up information about a key over time, basically impossible.

The bounded retrieval model is also the only leakage model that imposes significant constraints on the device in which it is implemented. To be most effective, the decrypter needs to store a large number of secret keys. Unfortunately, this is not possible for devices like smart cards which have limited storage.

Another drawback is that none of the schemes proposed so far in this model allow for leakage during key generation. While that may be a reasonable assumption, it means that the number of bits that can be leaked is absolute. That is, suppose Alice is running a scheme in this model and is worried that she is getting close to the leakage bound. She cannot simply use the key generation algorithm to generate extra secret information, since a small amount of leakage at that stage might totally compromise the algorithm [45].

2.2.5 Computationally Hard-to-Invert

Recall that the first model Dodis et al. [17] propose is that given $f(PK, SK)$ it is hard to compute SK. As it turns out, this model is often too strong. In fact, in the original version of their paper Dodis et al. claimed "for every *public-key encryption scheme*, we show an auxiliary input function h such that it is hard to compute SK given $h(PK, SK)$, and yet the scheme is completely insecure in the presence of the leakage given by h." Details of this h were to follow in the full version of the paper. However, in the full version this claim is simply removed.

Notice that if $|PK| \approx |SK|$ and if $h = SK \oplus PK$ (applying padding as necessary), then in general it seems very hard to recover SK just given this information. However, in any public key scheme we are also given PK and then it is trivial to recover SK.

A natural way to avoid this type of leakage is to redefine the leakage functions allowed. The other definition Dodis et al. consider is to make f allowed only if SK is hard to recover given $f(SK, PK)$ and PK. The two definitions of the classes of functions are given below.

Let $\mathcal{H}_{ow}(\ell(k))$ be the class of all polytime computable functions $f : \{0,1\}^{|SK|+|PK|} \to \{0,1\}^*$ such that given $f(SK, PK)$, no PPT algorithm can find SK with probability greater than $\ell(k)$, where $\ell(k) \geq 2^{-k}$ is the hardness parameter.

Let $\mathcal{H}_{pk-ow}(\ell(k))$ be the class of all polytime computable functions $f : \{0,1\}^{|SK|+|PK|} \to \{0,1\}^*$ such that given $f(SK, PK)$ and PK, no PPT algorithm can find SK with probability greater than $\ell(k)$, where $\ell(k) \geq 2^{-k}$ is the hardness parameter.

The notion of CPA security is the usual one although an attacker is additionally allowed to learn some function $h \in \mathcal{H}$ (where $\mathcal{H}$ is the allowed family of functions) before the challenge ciphertext is issued. A scheme that is CPA secure with respect to $\mathcal{H}_{ow}$ is called auxiliary input CPA secure ($\ell(k)$-AI-CPA secure). A scheme that is CPA secure with respect to $\mathcal{H}_{pk-ow}$ is called weak auxiliary input CPA secure ($\ell(k)$-wAI-CPA secure). Notice that for CPA security to hold, we must trivially have $\ell(k) \leq \text{negl}(k)$. By making ℓ as large as possible (that is by having ℓ be as big a negligible function of k as possible) we allow the largest class of functions.

Despite the more natural seeming definition of the "weak" notion of security, Dodis et al. note that by making PK contain more information about SK a larger class of leakage functions will be allowed. In a certain sense this makes the scheme "more secure". In fact, the authors claim that if $PK = SK$ the scheme is "wAI-CPA 'secure', since we [have] now ruled out all 'legal' auxiliary functions, making the notion vacuously true." This statement is true since their definition makes every scheme secure with respect to auxiliary inputs from the empty set. Since this does not seem to be a fundamental restriction of the definition of security, we find this claim misleading. A more reasonable definition would make such a scheme totally insecure.

The full version also presents a more "convincing" example. But they only show that a stronger looking version of a scheme "might" be insecure, while the weaker (looking) version, is definitely secure. Since they do not provide a concrete example of such a scheme or auxiliary function, it is unclear whether the necessary conditions for the stronger looking scheme to be insecure can ever be met.

Regardless, they claim wAI-CPA security is still a useful notion when PK is short. They prove that when $|PK| = t$, then $\ell(k)$-wAI-CPA security implies $(2^{-t}\ell(k))$-AI-CPA security. Since the proof amounts to guessing a value of PK, we can see that the example given above of the problems with AI-CPA security does not apply to schemes with small public keys (compared to the size of their private keys). They also argue that weak auxiliary input security allows the reuse of the same public key and secret key pair for multiple cryptographic applications (e.g. encryption and signing) without weakening the security of either scheme.

Unfortunately, even if we decide that both of the above notions are "useful", it is not clear how a scheme that is secure with respect to either of them fares against side channel attacks. In particular, a side channel attack (or combination of side channel attacks) seems to be allowed under the definition as long as it doesn't work (i.e. it doesn't recover the secret key). However, the definition gives us no insight into which attacks will work. Thus when asked whether or not a scheme secure in this model is secure against a particular class of side channel attacks, the authors of the scheme will likely be forced to answer "We don't know." While this is a similar criticism to the one we gave for the bounded entropy model, it is even stronger here, since it is even harder to tell which leakage functions are allowed.

There is another way in which it is very hard to analyze the security guarantee of the computationally hard-to-invert model. That is, the leakage parameter of these schemes is an upper bound on the success probability any algorithm has of guessing the secret key based on observing the leakage. So suppose we have a scheme that is $2^{-f(k)^{\varepsilon}}$ secure. What value of ε should we pick when implementing our scheme? The problem is that our desired upper bound on the success probability for an attacker depends on the number of computations she must perform to attain that probability. Since the security definition gives us no information about how many computations the best attacker must perform, it is very difficult to understand what an upper bound on her success probability should be. As an example, consider an attacker who performs no computations – her success probability should certainly not be more than 2^{-k} if our security parameter is to have any meaning. Even having made that assumption, we still cannot tell which functions of the secret key leave us vulnerable to that bound. For example, a lemma in [17] assures us that this as at least as strong as being able to leak $k - \log(1/2^{-k}) = 0$ bits. So while it is likely that we would be able to leak some key bits (initializing BHHO, one of the schemes shown to be AI-CPA secure in [17], at the 128-bit security level with these restrictions yields a key of almost 20000 bits), there is no guarantee of this fact.

2.2.6 BHHO

One of the schemes used by at least two papers [17, 60] is the BHHO encryption scheme proposed by Boneh et al. [9]. This scheme may be vulnerable to a non-standard side channel attack. The private key in the scheme is an m-bit string $\mathbf{s}$ (i.e. $s_i \in \{0, 1\}$). Then for public parameters $g_1, \ldots, g_m$ (elements of some group G), the public key is

$$y := \prod_{i=1}^{m} g_i^{s_i}.$$

Notice that a naïve implementation of this scheme is almost certainly vulnerable to a (simple) power analysis attack on key generation. If a multiplication is done in a given step, then the associated key bit is one, otherwise, it's zero. Obviously, there are countermeasures that could be implemented against such attacks (although care of course needs to be taken since, for example, inserting dummy operations can lead to other vulnerabilities), but they are less studied than countermeasures for point multiplication.

We should note that [60] propose a modification of the scheme that improves its efficiency. They suggest that instead of picking the secret key as a binary string, that it should consist of ℓ large exponents instead. This suggestion seems to accidentally avoid the above concerns, in that it defeats the attack although that was not the motivation for introducing the modification.

2.2.7 Indistinguishability and Leakage Resilience

The notion of indistinguishability may be hard to achieve in the face of leakage in practice. For example, Standaert et al. claim that power analysis attacks often reveal the Hamming

weight of a message [69]. In cases where the adversary has control of the message inputs, learning the Hamming weight of the message being encrypted clearly ruins any chance of indistinguishability. Even if the messages are selected randomly there is little chance that they will have the same Hamming weight. These attacks essentially deny any hope of achieving a standard notion of security for leakage resilient encryption schemes.

The problem with the security definition may not be easy to fix by using a slightly relaxed definition. For example, attempting to define indistinguishability for messages of the same Hamming weight does not seem to be a particularly good solution. This is because an attacker will likely be able to determine Hamming weight of each word (a string of bits equal in length to the processor's instruction inputs) of the secret key. Further, any more precise definition may become platform dependent in addition to being hard to work with.

So if (ciphertext) indistinguishability cannot be achieved, what notion of security should be used for leakage resilient encryption schemes? One of the difficulties in answering that question was brought up in Section 1.2.6. That is, relatively little research has been done on side channel attacks whose goal is message recovery (or partial recovery) especially in public key settings. Nonetheless, it may be the case that a comparatively weak notion of security suffices for public key encryption schemes, since in practice they encrypt random messages. Thus, it might be reasonable to expect that being able to distinguish between messages is not particularly helpful for an attacker.

A related problem is the following dilemma facing leakage resilient encryption schemes. In standard schemes, where encryption is treated as a black box, it makes sense that an attacker only sees one encryption from the challenger. In the new model it may be advantageous to watch the attacker encrypt the same message several times. Furthermore, even if the output of two functions is indistinguishable, it may be easy for an attacker who observed the computation to decide which one was used. For example, some (bit) encryption schemes act differently to encrypt the message 0 as they do for message 1. Depending on how different these operations are, even a simple power analysis attack could be used to determine which message was encrypted. Beyond that, it may be very hard to make the two different operations have indistinguishable physical output. In at least one case [11], a leakage resilient encryption scheme has been proposed where the two operations appear to be easily distinguishable. This points to a fundamental flaw in the security definition for leakage resilient encryption schemes.

Indistinguishability is one of the most important tools in theoretical cryptography. The security proofs for many schemes use the fact that an attacker breaking scheme A must also break scheme B since scheme B's interaction with the attacker is computationally indistinguishable from scheme A's. However, since side channel attacks might reveal which of the functions is being computed, extreme care needs to be taken when applying these methods especially when decisional assumptions are used.

2.2.8 Cold Boot and Partial Key Recovery Attacks

The (memory) bounded leakage model seems to most closely correspond to cold boot attacks where an attacker learns a significant fraction of the key bits. The question facing

standard schemes in the face of such leakage is whether or not they are vulnerable to partial key attacks. That is, can learning a substantial number of key bits lead to an attack that is much faster than the best known algorithm in the smaller key space? In the case of factoring we know the answer to be yes. RSA is particularly vulnerable [36] since when the public encryption exponent is $2^{16} + 1$, as it often is in practice, many of the secret key bits are revealed.

There are no known partial key attacks on systems based on discrete log or other related assumptions. In particular, elliptic curve systems have not been shown to be weak to such attacks. We are also unaware of any research suggesting that such attacks are either likely or unlikely to exist. Evidence for the existence of these attacks would dramatically increase the importance of developing leakage resilient schemes, especially given the discovery of cold boot attacks. However, evidence that such attacks do not exist would leave leakage resilient schemes offering very little in terms of security gains.

Let us more carefully examine the ramifications of the existence of these attacks. If they do exist, there are still two important ways in which leakage resilient schemes may not be as good as they sound. First, while some schemes allow almost the entire secret key to leak, to permit, for example, 90% of the key bits to leak would require them to be impractically slow. Thus, it may be the case that we are only guaranteed safety if a small fraction of key bits leak. Depending on the attacks discovered on standard schemes, it may be that this fraction is less than the fraction needed for the attack to succeed. Even if this is not the case, it will likely be very difficult to ensure that too many bits do not leak. For example, suppose a partial key attack is discovered against Schnorr that requires about 30% of the key bits. Further suppose that a new leakage resilient scheme could guarantee security even if up to 40% of its secret key bits are leaked. Then it would only be reasonable to implement the new scheme if one expects more than 30, but less than 40 percent of the key bits to leak. It seems very unlikely that a practical scenario would arise where this would be the case. Nonetheless, the fact that the security bound of the leakage resilient scheme is violated does not necessarily mean that there is an attack against it.

The second way in which leakage resilience might not be the answer is related to our earlier discussion of exposure-resilient cryptography. That is, in this scenario we are specifically worried about leakages that reveal some number of key bits and not those that correspond to arbitrary functions of the secret key. Therefore, it seems to make little sense to allow the attacker this extra power if we are mostly concerned about attacks which do not make use of it. This is especially important in a context where efficiency is crucial, since the more powerful attacker may lead to more inefficient schemes.

Even if such attacks do not exist, it is still necessary to account for the security loss if we expect such leakages to occur. That is, to maintain 128 bits of security if we expect around half of our secret key to leak, we would need to implement Schnorr with a 512 bit key. This would entail a substantial loss in efficiency. Leakage resilient schemes deal with this issue to some extent, but since their goal is not specifically to maintain efficiency in the face of such leakage they are not optimal. We feel investigation into fast schemes with relatively large secret key sizes to be an interesting direction of research that could lead to practical results. Note that such schemes would not need to provide additional security

guarantees over standard schemes and so would not necessarily be provably secure with respect to the leakage (they would hopefully remain as secure as traditional schemes with large secret keys).

Studies of these sorts of attacks will also have important implications for the computationally hard-to-invert model. One of the issues with that model is determining which leakage functions are allowed. This appears to be a very difficult task. For example, no one can *prove* that discrete logarithm is a hard problem (doing so would mean solving the most important open question in computer science). Thus, it is assumed to be hard in order to prove the security of various schemes. However, the problem of finding a discrete log where some fraction of the bits are known requires a second assumption. The cryptographic community has gathered evidence over time that discrete log is indeed a hard problem and so it is considered a safe assumption. If such evidence were also gathered for discrete log with some (fixed) fraction of known bits, then it might also become a safe assumption. This would in some sense make the computationally hard-to-invert model more meaningful.

2.2.9 Summary

We conclude this chapter by summarizing our outlook on leakage resilience. First, we need to address the question of which leakage model is best. As noted above, all of the models have their downsides. The computationally hard-to-invert model is in some ways the setting most analogous to practical concerns. We only need to worry about attackers who are computationally bounded. However, it is extremely difficult to analyze the security guarantees of this model, since understanding which leakages are allowed is very hard. On the other hand, the bounded leakage model, while being much easier to analyze, may not be strong enough to cover many important examples of side channel attacks. The bounded retrieval model provides a clever approach but does not guarantee the security of a single exchange. Further, there is no compelling evidence to suggest that these models are more secure against realistic attacks than conceptually easier models which only allow key bits to be leaked. Additionally, the advantages imbued by continual variants may not be any greater than schemes in the standard models which allow for key updates.

Another observation is that as leakage resilience has "progressed" schemes have become "better" by allowing for either more leakage or a larger class of leakage functions. However, we feel that this a bad approach to take. In particular, we would rather see schemes which soundly defeat smaller classes of attacks, than those which provide slim guarantees against a large class of attacks. For example, the creation of an efficient scheme which is provably secure against power analysis attacks would be a major accomplishment. Further, implementers of such a scheme would now need to install countermeasures for fewer attacks and not worry about certain countermeasures aiding power analysis attacks. Even a scheme that could guarantee security if an attacker did not collect too many power traces would be an interesting result.

Another reason the mantra of "more leakage" is a poor approach is that by focusing only on allowing as much leakage as possible the proposed schemes are almost all impractically inefficient. This is one of a number of ways in which work on leakage resilience has failed

to address many of the practical concerns surrounding side channel attacks. Impractical solutions to practical problems are useless except to the extent that the ideas present in such solutions can be used in other contexts or the solutions can be adapted to become realistic. Unfortunately, the theoretical model of leakage resilience is not meaningful with the practical motivation removed and the current literature provides no analysis of the details required to implement leakage resilient schemes.

At best, leakage resilience represents only one piece of the puzzle in terms of combatting side channel attacks. Even then, current approaches may be leading away from the desired goal. Significant analysis is needed to better determine what information is revealed by realistic attacks on both standard and new schemes. Especially important is empirical analysis to determine what sort of information is available to attackers in practice. Any successful theoretical approach towards combating side channel attacks needs to be informed by the entire practical context in which the schemes will be deployed.

Chapter 3

A Leakage Resilient Signature Scheme

3.1 Schnorr-ℓ

The first leakage resilient scheme we will discuss is a natural extension of the Schnorr signature scheme, independently discovered by Alwen et al. [3] and Katz [42]. The idea is to move from one base g and exponent x to ℓ of each. The scheme is provably secure in the memory bounded leakage model. One of the key ways it achieves this security is by having many different secret keys correspond to a single public key. Thus, even when an attacker learns some bits of the secret key she will still not be able to determine (information theoretically) which secret key the signer has. This is important, since in general with schemes in this model learning two different secret keys is hard. Our approach mirrors that of [42], although our proofs fill in some missing steps for greater clarity.

The scheme makes use of system parameters H, a random oracle, q, a large prime, and G, a group of order q where the discrete log problem is hard.

Key Generation: To generate her keys, Alice first selects $g_1, g_2, \ldots, g_\ell \in_R G^*$ and $x_1, x_2, \ldots, x_\ell \in_R \mathbb{Z}_q^*$. She then sets $h := \prod_{i=1}^{\ell} g_i^{x_i}$. Alice's public key is $PK = (g_1, g_2, \ldots, g_\ell, h)$ and her secret key is $SK = (x_1, x_2, \ldots, x_\ell)$.
Signing: To sign a message m, Alice first selects $r_1, r_2, \ldots, r_\ell \in_R \mathbb{Z}_q^*$. Next, she computes $A := \prod_{i=1}^{\ell} g_i^{r_i}$ and calculates the hash $c := H(m||A)$. Finally, she outputs the signature $(A, \alpha_1, \alpha_2, \ldots, \alpha_\ell)$, where $\alpha_i = cx_i + r_i \pmod q$.
Verification: To verify Alice's signature $(A, \alpha_1, \alpha_2, \ldots, \alpha_\ell)$ on a message m, Bob first computes $c := H(m||A)$. Then he checks that $A \cdot h^c = \prod_{i=1}^{\ell} g_i^{\alpha_i}$. If equality holds, he accepts the signature.

Figure 3.1: The Schnorr-ℓ signature scheme.

The proof of security will show how a forger can be used to solve the ℓ-representation problem.

Definition 3.1.1. The *ℓ-representation problem* is defined as follows: given group elements $g_1, g_2, \ldots, g_\ell$ find $\mathbf{x} = (x_1, x_2, \ldots, x_\ell)$ and $\mathbf{x}' = (x'_1, x'_2, \ldots, x'_\ell)$ such that $\prod_{i=1}^{\ell} g_i^{x_i} = \prod_{i=1}^{\ell} g_i^{x'_i}$ and $\mathbf{x} \neq \mathbf{x}'$.

It is easy to see that the ℓ-representation problem is at least as hard as discrete log, since solving $g_1^{x_1} g_2^{x_2} = g_1^{x'_1} g_2^{x'_2}$ allows one to recover $\log_{g_1} g_2$.

Recall Definition 1.1.4, the concept of min-entropy, which gives us a measure of how close a random variable's distribution is to uniform. In particular, when there are many different secret keys that correspond to one public key, just given the public key the attacker views all corresponding secret keys as being equally likely and so the min-entropy of the secret key is high. If the min-entropy of the secret key is high even after leakage occurs, then there remain many possible secret keys. Thus, even if an attacker recovers a secret key, it is unlikely that it will equal the original secret key.

Next we will prove a lemma that will be a key ingredient in the security proof of the scheme. In words, it tells us that learning an arbitrary function f of a random variable X is very unlikely to decrease the min-entropy of X by significantly more than the size of the image of f.

Lemma 3.1.2 ([42])**.** *Let X be a random variable with min-entropy η and f be an arbitrary function with image $\{0,1\}^\lambda$. For any $\Delta \in [0, \eta]$, if*

$$Y = \{y \in \{0,1\}^\lambda \mid H_\infty(X \mid y = f(X)) \leq \eta - \Delta\},$$

then $\Pr[f(X) \in Y] \leq 2^{\lambda - \Delta}$.

Proof. Fix $y \in \{0,1\}^\lambda$. Let $x \in \{0,1\}^n$ be such that $f(x) = y$ and x is a minimizer of $H_\infty(X \mid y = f(X))$. Note that

$$\Pr[X = x \mid y = f(X)] = \frac{\Pr[X = x]}{\Pr[y = f(X)]},$$

since $\Pr[X = x \text{ and } y = f(x)] = \Pr[X = x]$. Thus, by the definition of x and Y, we have $y \in Y$ only if

$$\begin{aligned}
-\log_2 \frac{\Pr[X = x]}{\Pr[y = f(X)]} &\leq H_\infty(X) - \Delta \\
-\log_2 \frac{\Pr[X = x]}{\Pr[y = f(X)]} &\leq -\log_2 \Pr[X = x] - \Delta \qquad (1) \\
\frac{\Pr[X = x]}{\Pr[y = f(X)]} &\geq \Pr[X = x] \cdot 2^{\Delta} \\
\frac{1}{\Pr[y = f(X)]} &\geq 2^{\Delta} \\
\Pr[y = f(X)] &\leq 2^{-\Delta},
\end{aligned}$$

where (1) follows since $-\log_2 \Pr[X = x] \geq H_\infty(X)$. Since the range of f is $\{0,1\}^\lambda$, we have $|Y| \leq 2^\lambda$ and so, since y was arbitrary, $\Pr[f(X) \in Y] = |Y| \cdot \Pr[f(X) = y] \leq 2^{\lambda - \Delta}$ as desired. □

The idea of the security proof is to induce the forger to create two different signatures for the same message. We will show that doing this allows us to find exponents $x'_1, \ldots, x'_\ell$ so that $\prod_{i=1}^{\ell} g_i^{x_i} = \prod_{i=1}^{\ell} g_i^{x'_i}$. The proof concludes by arguing that with high probability $\mathbf{x} \neq \mathbf{x}'$ and thus using a forger we can solve the ℓ-representation problem.

Theorem 3.1.3 ([42])**.** *If the discrete log problem is hard in G, then the above scheme is leakage resilient in the memory bounded leakage model with leakage parameters (L, L, L) where $L = \dfrac{\ell - 1 - 2\ell\varepsilon}{2\ell}$.*

Proof. We will construct an algorithm $\mathcal{A}$ that, given a forger $\mathcal{F}$ for the scheme, solves the discrete logarithm problem in G. In particular, such an $\mathcal{A}$ will be polytime and succeed with non-negligible probability as long as both those properties hold for $\mathcal{F}$.

Let q_H be the number of hash queries made by the forger and suppose it succeeds with probability δ. We will assume that if $\mathcal{F}$ outputs the signature $(A, \alpha_1, \ldots, \alpha_\ell)$ for the message m, then at some point it queried the random oracle for $H(m||A)$. We can do this without loss of generality since a successful forgery requires knowledge of $H(m||A)$ except with negligible probability.

To solve the ℓ-representation problem with respect to $g_1, \ldots, g_\ell$, $\mathcal{A}$ first picks a random private key $x_1, \ldots, x_\ell$ and sends the corresponding public key to $\mathcal{F}$. Then $\mathcal{A}$ plays the security game of Definition 1.1.2 with $\mathcal{A}$ simulating the random oracle for $\mathcal{F}$.

Eventually, $\mathcal{F}$ outputs a signature $(A, \alpha_1, \ldots, \alpha_\ell)$ for m. If this is valid, $\mathcal{A}$ rewinds $\mathcal{F}$ to the point where it first queried the random oracle for $H(m||A) = c$. At this point, $\mathcal{A}$ picks a new random value c' for this query and continues $\mathcal{F}$ answering signature queries using new random values, but being consistent with the previous run when answering random oracle queries. We will say $\mathcal{A}$ succeeds if $\mathcal{F}$ outputs a second valid signature $(A, \alpha'_1, \ldots, \alpha'_\ell)$ for m.

If $c \neq c'$, then $h^c h^{-c'} = \prod_i g_i^{\alpha_i - \alpha'_i}$ since both signatures are valid. But $h^{c-c'} = \prod_i g_i^{(c-c')x_i}$, so $\prod_i g_i^{(\alpha_i - \alpha')/(c-c')} = \prod g_i^{x_i}$. Thus, taking $x'_i = (\alpha_i - \alpha'_i)/(c - c')$ yields a set of exponents solving the ℓ-representation problem provided $\mathbf{x} \neq \mathbf{x}'$.

In the following two claims we will argue that the probability of $\mathcal{A}$ succeeding is a non-negligible function of the probability of $\mathcal{F}$ creating a valid forgery and that the probability that $\mathbf{x} \neq \mathbf{x}'$ is also non-negligible.

Claim 3.1.4. *The probability ρ that $\mathcal{A}$ succeeds is at least δ^2/q_H.*

Proof. Let h_i denote the i^{th} hash query made by $\mathcal{F}$. If $\mathcal{F}$ terminates with a valid forgery $(A, \alpha_1, \ldots, \alpha_\ell)$ for m, we say that a hash query h_i is associated with the forgery if $h_i = m||A$. Let a_i be the probability that $\mathcal{F}$ reaches a state where it makes the query h_i. Further, let b_i be the probability that after making the query h_i, $\mathcal{F}$ eventually outputs a valid forgery associated with h_i.

Since every valid forgery is associated with some unique h_i, we have that $\sum_i a_i \cdot b_i = \delta$ the overall probability of success of $\mathcal{F}$. Further, it's clear that $\sum_i a_i$ is at most q_H.

Now, by construction we see that $\mathcal{A}$ succeeds with probability $\sum_i a_i \cdot (b_i)^2$. Recall Jensen's inequality [38] for a convex function φ, x_i in its domain, and positive constants c_i:

$$\frac{\sum c_i \varphi(x_i)}{\sum c_i} \geq \varphi\left(\frac{\sum c_i x_i}{\sum c_i}\right).$$

So we have

$$\begin{aligned}
\sum_i a_i \cdot (b_i)^2 &= \sum_i a_i \cdot \frac{\sum_i a_i \cdot (b_i)^2}{\sum_i a_i} \\
&\geq \sum_i a_i \cdot \left(\frac{\sum_i a_i \cdot b_i}{\sum_i a_i}\right)^2 \\
&= \frac{\delta^2}{\sum_i a_i} \\
&\geq \frac{\delta^2}{q_H},
\end{aligned}$$

as claimed. □

Claim 3.1.5. *The probability that $\mathcal{A}$ solves the representation problem is at least $1/2 \cdot (\rho - 1/q - q_H/q^{2\varepsilon\ell})$.*

Proof. If $\mathcal{A}$ succeeds, two events can happen which prevent $\mathcal{A}$ from solving the representation problem. First, it could be the case that $c = c'$, but this happens with probability $1/q$. Second, $\mathbf{x}$ could equal $\mathbf{x}'$. We will show that this is an unlikely occurrence. In particular, we will show that with high probability the min-entropy of $\mathbf{x}$ is at least 1 and so the probability that $\mathbf{x} \neq \mathbf{x}'$ is at least $1/2$.

Let $\lambda = (1/2 - 1/(2\ell) - \varepsilon) \cdot \ell \cdot \log q$. This is an upper bound on the number of bits leaked during each run of $\mathcal{F}$. The public key constrains $\mathbf{x}$ to lie in an $(\ell - 1)$-dimensional vector space over $\mathbb{Z}_q$ (i.e. there are $q^{\ell-1}$ possible choices for $\mathbf{x}$). Also, it is known [63] that signature queries do not further constrain $\mathbf{x}$. Thus, the min-entropy of $\mathbf{x}$ in the view of $\mathcal{F}$ based on the public key and the signature queries is $(\ell - 1) \cdot \log q$ bits. As well, $\mathcal{F}$ learns at most $2 \cdot \lambda$ bits of information about $\mathbf{x}$ from the leakage queries and an additional $\log q_H$ bits from its state associated with the first forgery. Applying lemma 3.1.2, the conditional min-entropy of $\mathbf{x}$ is at least 1 except with probability at most

$$2^{2\lambda + \log q_H - (\ell-1)\cdot\log q} = 2^{(\ell - 1 - 2\varepsilon\ell)\cdot\log q + \log q_H - (\ell-1)\cdot\log q} = q_H \cdot 2^{(-2\varepsilon\ell)\log q} = q_H \cdot q^{-2\varepsilon\ell}.$$

Thus, the probability that $\mathcal{A}$ succeeds is as claimed. □

Combining the two claims we see that $\mathcal{A}$ solves the ℓ-representation problem with probability at least

$$\frac{1}{2} \cdot \left(\frac{\delta^2}{q_H} - \frac{1}{q} - \frac{q_H}{q^{2\varepsilon\ell}}\right).$$

Since q_H is polynomial, ε is constant, and $1/q$ is negligible, this probability is non-negligible whenever δ is. Thus, no PPT forger can exist by the assumed hardness of the discrete logarithm problem. □

3.2 Comparison with Standard Schnorr

The obvious question to ask at this point is "What have we gained?". In particular, what security guarantees do we now have compared to standard Schnorr? To answer this, we will make the schemes more concrete. First of all, suppose we wish to instantiate both schemes using elliptic curves at the 128-bit security level. We will thus replace group exponentiations with point multiplications (so $g_i^{x_i}$ becomes $x_i P_i$). To know exactly how much we can leak, we need to know what values of ℓ and ε to pick.

Clearly, the larger ℓ is the more inefficient our scheme becomes, but the more leakage we can tolerate. With ε, the analysis is more difficult. Certainly smaller values of ε lead to better leakage results. However, here the impact is on the proof itself. More specifically the value of ε will determine how good a bound the proof will give us for the success probability of an attacker given some assumptions about the hardness of DLOG. Suppose we wish no attacker to forge signatures with probability greater than $\delta = 2^{-31}$. We believe that any algorithm trying to recover discrete logs in G that operates in t steps and succeeds with probability p satisfies $p/t \leq 2^{-128}$. Taking q_H to be the number of steps a forger performs, for the proof to be meaningful we need that

$$\frac{\frac{1}{2} \cdot \left(\frac{\delta^2}{q_H} - \frac{1}{q} - \frac{q_H}{q^{2\varepsilon\ell}} \right)}{q_H} \geq 2^{-128}.$$

If we take q_H to be 2^{32}, substituting in the desired value of δ yields $q^{2\varepsilon\ell} \geq 2^{127}$, which gives $\varepsilon\ell \geq 127/512 \approx 1/4$. Notice that since q_H is the bound on the running time and we only restrict it to being 2^{32}, this is a far too strong a restriction of our adversary to be meaningful in practice. Unfortunately, for $\delta = 2^{-31}$, the inequality above gives $q_H^2 \leq 2^{65}$ and so we cannot make q_H any bigger than 2^{32}. If instead we set $\delta = 2^{-10}$, then we could have q_H as large as 2^{53} which is somewhat of a meaningful restriction. In this case though, δ is too small for the result to be meaningful in practice. We should note that similar statements are true for the security guarantees provided by standard Schnorr, although the results are somewhat less severe in that case.

Table 3.1 gives the number of bits we can leak, for various values of ℓ and ε. Here for a given value of ℓ, we use the above calculation to determine an appropriate value of ε. Throughout the table we have that $k = 128$, $\delta = 2^{-31}$, and $q_H = 2^{32}$.

Notice that since Schnorr-ℓ contains ℓ times as many point multiplications as Schnorr during signing, we can expect it to take significantly longer to sign messages. However, the cost of these multiplications can be reduced using Shamir's trick. To calculate the value $\sum r_i P_i$, one first precomputes the 2^ℓ possible values of $\sum b_i P_i$ where $b_i \in \{0, 1\}$. Then one considers the matrix whose rows are the r_i's written in binary, starting with the least significant bit. Starting with the lefthand column one sets $A := \sum r_{i1} P_i$, where r_{ij} is the j^{th} bit of r_i. Proceeding rightwards we update $A := 2A + \sum r_{i2} P_i$ and so on. On average, this requires the same number of doublings and $2 - 1/2^{\ell-1}$ times as many additions as a single (naïve) point multiplication. For very small ℓ the precomputation stage is also very fast and reasonable to store (the table requires $2k \cdot 2^\ell$ bits to store). However, even for ℓ

Table 3.1: Allowed leakage in Schnorr-ℓ

ℓ	ε	Leakable bits	Secret key bits
3	1/12	192	768
4	1/16	320	1024
5	1/20	510	1280
6	1/24	576	1536
10	1/40	1088	2560
20	1/80	2568	5120
40	1/160	4928	10240
50	1/200	6208	12800
75	1/300	9408	19200
100	1/400	12608	25600

as small as 20, it requires 32 megabytes of storage at the 128-bit security level. Also, this operation still needs to be protected against side channel attacks and the added complexity of Shamir's trick may make this more difficult.

Despite this improvement, since even microsecond speedups can be important in practice, this slow down may represent a very large performance hit. Thus, it might be reasonable to demand that the security gain also be very large. To examine the security gain, suppose we implement Schnorr-4 to try to mitigate the efficiency loss. So what assurance does being able to leak 320 bits give us? Certainly, we are still not known to be safe against power analysis attacks since those attacks use megabits of information. In fact, in many ways, cold boot attacks are the thing best modelled by this type of security. These clearly result in a one time leakage of information which might be reasonably within the bounds of the model.

Note, as discussed in [45], if many signatures are computed in quick succession, there may be a high amount of key information in RAM. Suppose 50 signatures are computed back-to-back. Each signature requires 1024 bits of randomness and likely also requires us to copy the key into RAM. Further, there is no guarantee that the system will overwrite the RAM used for the operation when it finishes. Thus we might reasonably expect there to be about 100 kilobits (102400 bits) of secret information in RAM. Our ability to leak only 320 of those bits seems suddenly not so useful.[1] Even if the values in RAM are deleted after each signature, meaning only about 2048 secret bits are present at once, is there any reason to believe that an attacker would recover fewer than 320 bits? Since cold boot attacks can be assumed to reveal a percentage of secret key information, a similar attack on standard Schnorr with a secret key size of 256 bits would reveal less than 80 secret bits. Is that enough to be useful in practice?

One might argue, however, that even though computing signatures in quick succession might leave a lot of information in RAM, it may be that most of it is useless. For example,

[1]This example may be unrealistic. There is, however, no indication of this in the leakage resilient literature. The scarcity of analysis relating security guarantees to the strength of practical attackers makes it very hard to tell what "realistic" is.

an attacker may learn the same bits of one of the x_i's several times or perhaps they may learn several bits of the r_i's from different rounds. Even disregarding the validity of such claims (we will see an attack that recovers the key in the second scenario), the major advantage Schnorr-ℓ has over regular Schnorr in terms of leakage is that it comes with a proof of security.

If we start letting intuition about attacks guide us for Schnorr-ℓ, then we ought to do the same for Schnorr. For example, suppose we were to use Schnorr with a 256-bit key. The best known attack in this scenario is Pollard's rho algorithm which takes approximately $2^{256/2} = 2^{128}$ steps. Now suppose we learn the 128 least significant bits of the key. The best known attack now takes $2^{(256-128)/2} = 2^{64}$ steps. In other words, we have no evidence to suggest that revealing half the secret key in Schnorr does anything but leave us as if we were using a key half the size.

All this being said, if we have reason to believe that we may, over time, leak half the bits of our secret key, to obtain the same security level using normal Schnorr, we would have to double its size. Alternatively, we could use Schnorr-2. We have almost every reason to believe that Schnorr-2 is at least as secure as doubling the key size in Schnorr. This comes with the added benefit that the efficiency of doing two exponentiations in a fixed sized group is much better than doing one exponentiation in a group of squared size (even without using Shamir's trick).

Notice, however, that if no key leakage occurs, then a discrete logarithm attack on Schnorr-2 will run in about the square root of the amount of time it would take on Schnorr with a doubled secret key size (i.e. $x \approx q^2$). For example, if larger secret key sizes for Schnorr are suggested in the future, it will likely be because we believe that discrete log attacks on small key sizes are becoming feasible due to better hardware. In this case, we cannot just use Schnorr-2 with the same group order to increase security. Also, a minor drawback to Schnorr-ℓ is that there is a class of side channel attacks which function slightly faster than on regular Schnorr (where both have the same key size). Such an attack will be the topic of our next section.

3.3 Lattice Attacks

Let us consider a type of leakage that at first glance seems to fit within the model above. In this attack the attacker learns a few bits of the per-message secret (the r_i's) every time a message is signed. Certainly, each such leakage is allowed under the model. However, after many such leakages the attacker will have learned too much information and so this falls outside the model. Nonetheless, there is an attack that depends on this sort of leakage.

Howegrave-Graham and Smart [37] discovered lattice attacks on Schnorr-like signatures. Nguyen and Shparlinski [62] improved their attack and were able to recover the secret when even as few as 3 bits of each per-message secret were learned. We will describe the attack on regular Schnorr, then explain how it extends to the Schnorr-ℓ case.

These attacks centre around the fact that each signature is of the form (A, α) where $\alpha \equiv cx + r \pmod{q}$. By having many such congruences where a few of the least significant

bits of r are known, the attack is able to recover the secret key x. To continue with the description, we will need another piece of terminology.

Definition 3.3.1. Let $\mathbf{b_1}, \ldots, \mathbf{b_m} \in \mathbb{R}^n$ be a set of linear independent vectors. A *lattice* is the set of integer linear combinations of the $\mathbf{b_i}$'s. In particular, any element $\mathbf{v}$ of the lattice has the form $\mathbf{v} = a_1\mathbf{b_1} + \cdots + a_m\mathbf{b_m}$ where $a_i \in \mathbb{Z}$. We refer to the set $B = \{\mathbf{b_1}, \ldots, \mathbf{b_m}\}$ as a *basis* of the lattice.

The attack works by finding a short vector in a specially constructed lattice. In theory, finding such vectors is hard, but in practice there are efficient algorithms that have a reasonable chance of success. One such algorithm that we will use in our attack is the LLL lattice reduction algorithm [48]. The algorithm, given a basis B for a lattice L, outputs a second basis B' where all the vectors are short (i.e. they are not too much bigger than the smallest possible nonzero vectors in the lattice with respect to some norm). While the theoretical guarantees on the shortness of the vectors in B' are not very good, in practice LLL tends to do much better than what is guaranteed.

The attack

Throughout this section the notation $a \pmod q = b$ means the least positive residue of $a \bmod q$ equals b. A similar notion is meant if "=" is replaced with some other comparison sign like "$\leq$". Here we describe the approach of [62]. The authors formulate the problem of finding x as a hidden number problem (HNP). In HNP one is given d random values t_i as well as values u_i so that $(xt_i - u_i) \bmod q \leq q/2^{s+1}$. In a sense, we can view the u_i's as revealing the s most significant bits of xt_i. The problem is then to find the hidden number x.

Suppose we are given the s least significant bits of the per-message secret r in a Schnorr signature. In particular, we know $0 \leq a \leq 2^s - 1$ so that $r - a = 2^s b$ for some non-negative integer b. Then the congruence $\alpha \equiv cx + r \pmod q$ can be re-written as

$$\begin{aligned} \alpha - cx &\equiv a + 2^s b \pmod q \\ (\alpha - a)2^{-s} - 2^{-s}cx &\equiv b \pmod q. \end{aligned}$$

In particular, if we set $t := -2^{-s}c \pmod q$ and $u := (\alpha - a)2^{-s} \pmod q$, then $(t - u) \pmod q = b < q/2^s$. Note that since c, α, and a are known, t and u can be easily computed. Thus, learning the s least significant bits of r reveals the s most significant bits of xt.

Now suppose we are given d signatures and the s least significant bits of the per-message secret for each signature. As seen above, this is equivalent to being given d random $t_i \in \mathbb{Z}_q^*$ and the corresponding s most significant bits u_i of xt_i. By definition we then have $(xt_i - u_i) \bmod q \leq q/2^{s+1}$. Now consider the $(d+1)$-dimensional lattice L

spanned by the rows of the matrix

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 \\ 0 & q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & q & 0 \\ t_1 & \cdots & \cdots & t_d & 1/2^{s+1} \end{pmatrix}.$$

Let $xt_i \pmod q =: y_i$. Notice that the vector $\mathbf{h} = (y_1, \ldots, y_d, x/2^{s+1})$ belongs to L since it can be obtained by multiplying the last row by x and then subtracting multiples of the first d rows. From $\mathbf{h}$ we can clearly recover the hidden number x and so it is known as the hidden vector. Notice $\mathbf{h}$ is very close the vector $\mathbf{u} = (u_1, \ldots, u_d, 0)$. In fact, since $(x_i - u_i) \bmod q < q/2^{s+1}$ and $x < q$ we have $\|\mathbf{h} - \mathbf{u}\|_\infty < q/2^{s+1}$.

Suppose $\mathbf{v}$ is a vector in L even closer to $\mathbf{u}$ than $\mathbf{h}$ is, i.e. $\|\mathbf{v} - \mathbf{u}\|_\infty < \|\mathbf{h} - \mathbf{u}\|_\infty$. Then, by the triangle inequality, $\|\mathbf{h} - \mathbf{v}\|_\infty < q/2^s$. Notice however that $\mathbf{h} - \mathbf{v}$ is a vector in L. Without going into too many details, this vector is very short (we would expect an arbitrary vector to have infinity norm close to q). Thus, it is reasonable to expect it has a special form. Lemma 1 in [62] proves that with high probability (given large enough s and d) the vector $\mathbf{h}-\mathbf{v}$ is of the form $(0, \ldots, 0, \beta/2^{s+1})$ where β is a multiple of q. In particular, we have that if γ is the last coordinate of $\mathbf{v}$ then $2^{s+1}\gamma = x + \beta$ and so $x \equiv 2^{s+1}\gamma \pmod q$. Thus, finding $\mathbf{v}$ reveals x.

All that remains is to find a method to find such a $\mathbf{v}$. As it turns out, this is a well-known problem called the closest vector problem (CVP). Babai's CVP algorithm [4] takes as input a lattice and a vector and outputs the vector in the lattice that is closest (with respect to some norm) to the given vector. While this is certainly one approach that can be taken, experiments in [62] yielded better results using the following approach. Define a new lattice L' spanned by the rows of the following matrix

$$\begin{pmatrix} \Lambda & \mathbf{0} \\ \mathbf{u} & q/2^{s+1} \end{pmatrix}.$$

We then apply the LLL algorithm to reduce the basis into short vectors. Our hope is then that the shortest vector in L' or one of the reduced basis vectors is of the form $(\mathbf{u} - \mathbf{v}, q/2^{s+1})$. Table 3.2 shows values of d and s known to recover x for a 160-bit prime q [37, 62].

The attack can be extended to one on Schnorr-ℓ by finding each x_i individually. For example, one could use the equations of the form $\alpha_1 = cx_1 + r_1$ to recover x_1 and then do the same for x_2 and so on. Thus, such an attack should take exactly ℓ times as long and require ℓ times as many leaked bits, as an attack on regular Schnorr.

We can see that at a minimum the attack needs at least the same number of bits as the length of the key to be leaked. Thus, it clearly falls outside the security model of Schnorr-ℓ. In fact, any scheme using the (memory) bounded leakage model will not be able to account for such an attack. The problem here is that while only a few bits are being leaked each

s	d
80	2
40	4
16	11
8	30
5	70
3	100

Table 3.2: Parameters for successful lattice attacks

signature, over time this will eventually mean the leakage conditions are violated for any scheme in this model. This fundamental problem can be avoided by including a key update mechanism. Such schemes, and the corresponding security model, will be the topic of the next chapter.

Chapter 4

Leakage Resilience and Key Updates

4.1 The Case for Key Updates

We have seen that lattice attacks gather information each round until a threshold is reached and the secret key can be recovered. Therefore, the attacks can be totally defeated by changing the secret key before too much information is leaked. Unfortunately, the only way to change the secret key in standard schemes is to re-key the system and so also create a new public key. Since public key distribution is a very expensive operation, it should be avoided except when absolutely necessary. Thus, an additional property one might desire for a scheme is the ability to update the secret key without changing the public key.

Schemes with key updates have the property that there are many secret keys corresponding to a single public key. Notice that this was the case in Schnorr-ℓ. Possession of any one of these secret keys allows the owner to perform the main function of the scheme (e.g. signing). Thus, an attacker learning any one of these keys (even one never used by the real owner of the secret key) represents a total compromise of the system. However, many of these systems have the property that information learned about one secret key gives (almost) no information about any of the other possible secret keys.

With that in mind, let's investigate a method to add key updates to Schnorr-ℓ. Recall that Schnorr-ℓ satisfies the property that there are $q^{\ell-1}$ secret keys corresponding to a given public key (where q is the group size). Unfortunately, the hardness of the ℓ-representation problem guarantees that the signer, Alice, cannot update her secret key. However, suppose that instead of choosing random group elements obliviously, Alice generates her public key by selecting a generator $g \in_R G$ and ℓ exponents $a_1, \ldots, a_\ell \in_R \mathbb{Z}_q^*$ and then setting $g_i := g^{a_i}$. Then to update her secret key she could first compute $X := \sum_i a_i x_i$. Next she could select $x'_1, \ldots, x'_{\ell-1} \in_R \mathbb{Z}_q^*$ and set

$$x'_\ell := \left(X - \sum_{i=1}^{\ell-1} a_i x'_i \right) / a_\ell.$$

Her new secret key $\mathbf{x}'$ still satisfies the property $\prod_i g_i^{x'_i} = h$ and so is a valid secret key corresponding to the public key, h. The scheme also has the nice property that Shamir's

trick is no longer needed to perform the exponentiation during signing; instead we have that $\prod_i g_i^{r_i}$ can be computed as g^R where $R := \sum a_i r_i$. Thus, the cost of an exponentiation is basically identical to what it is in standard Schnorr.

Unfortunately, this approach is totally insecure in the context of leakage resilience. Recall that continual memory bounded length schemes can leak an unbounded amount of information over time. In particular, over time an attacker can learn any information which does not change between key updates. In this case, we have that the values $a_1, \ldots, a_\ell$ and the discrete logarithm X of h are all fixed. Thus an attacker can eventually learn these values and then compute a secret key using the exact same procedure Alice uses to update her key.

However, this method does prevent the lattice attacks described earlier provided that the key is updated before the threshold of bits required to recover the key is breached. Notice that with high probability even an attacker who learns the values $x_1, \ldots, x_{\ell-1}$ does not learn anything about the discrete logarithm X of h unless she also learns x_ℓ. Further, it would appear that the operations associated with the scheme are unlikely to leak information about X directly under known side channel attacks. Thus, even though this scheme is not leakage resilient, it may be more secure against side channel attacks than standard schemes.

To attain continual leakage resilient security we will examine a totally different scheme proposed by Malkin, Teranishi, Vahlis, and Yung in [52], which will refer to as MTVY. While lattice attacks are not applicable to this scheme, it also aims to defeat DPA attacks, which share many properties with lattice attacks. Most notably, recall that DPA works by collecting many power traces using the same secret value. If too few traces are collected, then the noise is too great and little or no information is revealed. However, if enough traces are collected potentially the whole secret key can be recovered. Thus, by changing the secret key relatively frequently, the goal is to ensure that the required number of traces can no longer be gathered. The approach we take in the next few sections mirrors that of [52], although we make a few minor changes. Most notably, in several places we "sacrifice" formality in an attempt to make concepts easier to understand. Nonetheless, the various theorems and lemmas (and the ideas used to prove them) that make up the security proof are essentially unchanged from [52].

4.2 Intuition for MTVY

To explain the idea behind MTVY, let us first consider what exactly a signature is. A signature on a message m can be viewed as a (non-interactive) proof of knowledge of SK that somehow incorporates m. Furthermore, this proof should give almost no information about the secret value SK. The secret value used in a proof of knowledge is known as a *witness*. MTVY makes use of a proof system which guarantees that no information is revealed about which witness was used, provided that a certain problem is hard. This forms the basis for the signature scheme.

Signature schemes, like many non-interactive proof systems, are often constructed by performing a transformation to an interactive protocol. Traditionally, the security proofs for signature schemes often relied on a random oracle assumption, since they were based on the Fiat-Shamir transform [29]. This transformation turns an identification scheme, where the prover sends a commitment to the verifier and then receives back a (random) challenge before sending the proof, into a signature scheme. It works by using the random oracle to preserve both the commitment and the random challenge. Recently, Waters developed a clever method to avoid the use of random oracles in many schemes that rely on them for security [70]. This technique is known as the Waters hash and is used by MTVY to avoid the random oracle assumption.

With these two pieces in place we can begin to describe the idea behind MTVY. First, parameters for the proof system are chosen and they become the secret and public keys. To sign a message, a common reference string (CRS) is chosen based on the Waters hash of the message. The signature is a proof using the proof system as well as the common reference string. To verify a signature, Bob runs the verification algorithm of the proof system using the common reference string. The secret key is actually a randomly selected element of a given affine space (any element in this space will be a valid secret key). To update, Alice simply selects another random element of the affine space (by adding a specially formed random offset). The security of the scheme is based on the hardness of finding a witness not in the affine space. The proof of security relies on properties of the proof system and a lemma from Brakerski et al. [11] to show that even with leakage a successful forgery leads to the recovery of a witness outside the affine space with high probability.

4.3 Preliminaries

The MTVY scheme makes use of three groups of prime order q, G_1, G_2, and G_T and a pairing $e : G_1 \times G_2 \to G_T$. We will refer to the identity element in G_T as ID_T and the identity elements in G_1 and G_2 as 0. The hardness assumption is the *symmetric external Diffie-Hellman* (SXDH) assumption which states that DDH is hard in both G_1 and G_2. For vectors $\mathbf{u} \in G_1^\ell$ and $\mathbf{v} \in G_2^\ell$ the notation $e(\mathbf{u}, \mathbf{v})$ refers to the product $\prod_{i=1}^{\ell} e(u_i, v_i) \in G_T$. It is easy to see that $e(a\mathbf{u}, b\mathbf{v}) = e(\mathbf{u}, \mathbf{v})^{ab}$ for all $a, b \in \mathbb{Z}_q$.

Let $R \in (\mathbb{Z}_q)^{n \times \ell}$ be a matrix, $\mathbf{u} \in G_1^\ell$ and $\mathbf{v} \in G_2^n$. If R_i is the i^{th} row of R, we have

$$\begin{aligned} e(R\mathbf{u}, \mathbf{v}) &= \prod_i e(R_i \cdot \mathbf{u}, v_i) = \prod_i e\left(\sum_j R_{ij} u_j, v_i\right) \\ &= \prod_i \prod_j e(R_{ij} u_j, v_i) = \prod_i \prod_j e(u_j, R_{ij} v_i) \\ &= \prod_j \prod_i e(u_j, R_{ij} v_i) = \prod_j e(u_j, (R^T)_j \cdot \mathbf{v}) \\ &= e(\mathbf{u}, R^T \mathbf{v}). \end{aligned}$$

Remark 4.3.1. The security proof will require the concept of *statistical distance*, a measure of how similar two different distributions are. We will not provide a more rigorous definition, since the details are not important. We will only make use of two properties of statistical distance. First, if the distance between two distributions is negligible then *any* adversary has negligible advantage in distinguishing between them. Further, the distance satisfies the triangle inequality; that is if the distance is denoted $\text{dist}(\cdot,\cdot)$, then for any random variables X, Y, and Z, we have

$$\text{dist}(X,Z) \leq \text{dist}(X,Y) + \text{dist}(Y,Z).$$

We also require some background from linear algebra.

Definition 4.3.2. An *affine space* A is the set of all vectors formed by summing all the elements of a vector space with a fixed vector that is not in the vector space. In other words, let V be a vector space and $\mathbf{v}$ an arbitrary vector not in V. Then we can write $A = \{\mathbf{v}+\mathbf{w} \mid \mathbf{w} \in V\}$. It is easy to see that any affine space satisfies the following property. If $\mathbf{z_1}, \ldots, \mathbf{z_n} \in A$, then we have

$$\sum_{i=1}^{n} a_i \mathbf{z_i} \in A \text{ whenever } \sum_{i=1}^{n} a_i = 1.$$

For any vectors $\mathbf{x_1}, \ldots, \mathbf{x_n}$ we denote by $\text{Aff}(\mathbf{x_1}, \ldots, \mathbf{x_n})$ the smallest affine space containing $\mathbf{x_1}, \ldots, \mathbf{x_n}$.

4.3.1 Groth-Sahai Proofs

The Groth-Sahai proof system [33] makes use of two algorithms, a prover and a verifier. The proofs show knowledge of a witness $\mathbf{w} \in G_2^\ell$ so that $e(\mathbf{v}, \mathbf{w}) = t$ where $(\mathbf{v}, t) \in G_1^\ell \times G_T$ is the input to the verifier. Let the common reference string be $crs = (\mathbf{g}, \mathbf{h}) \in G_2^{2\times 2}$. The two algorithms can then be described as follows.

Prover To prove a statement using the inputs crs, $(\mathbf{v}, t)$ and $\mathbf{w}$, Alice selects the matrix $R \in_R \mathbb{Z}_q^{\ell\times 2}$, then computes

$$\mathbf{c} := R\mathbf{g}, \qquad \mathbf{d} := \mathbf{w} + R\mathbf{h}, \qquad \text{and} \qquad \mathbf{f} := R^T\mathbf{v}$$

and outputs $\sigma = (\mathbf{c}, \mathbf{d}, \mathbf{f})$.

Verifier To verify a proof $\sigma = (\mathbf{c}, \mathbf{d}, \mathbf{f})$ under crs, Bob checks if $e(\mathbf{v}, \mathbf{c}) = e(\mathbf{f}, \mathbf{g})$ and $e(\mathbf{v}, \mathbf{d}) = t \cdot e(\mathbf{f}, \mathbf{h})$. If both equalities hold, he accepts and otherwise he rejects.

Notice that

$$e(\mathbf{v}, \mathbf{c}) = e(\mathbf{v}, R\mathbf{g}) = e(R^T\mathbf{v}, \mathbf{g}) = e(\mathbf{f}, \mathbf{g})$$

and

$$e(\mathbf{v}, \mathbf{d}) = e(\mathbf{v}, \mathbf{w} + R\mathbf{h}) = e(\mathbf{v}, \mathbf{w}) \cdot e(R^T\mathbf{v}, \mathbf{h}) = t \cdot e(\mathbf{f}, \mathbf{h})$$

and so the verifier always accepts a correctly generated proof.

Groth and Sahai showed that for random $\mathbf{g}$ and $\mathbf{h}$, the proof system is perfectly witness hiding. That is, no (unbounded) attacker can determine which witness was used to compute the proof with more than negligible probability. On the other hand, when $\mathbf{g}$ and $\mathbf{h}$ form a DDH tuple, i.e. $\mathbf{h} = \alpha\mathbf{g}$ for some $\alpha \in \mathbb{Z}_q$, then whenever $\mathbf{c}$ and $\mathbf{d}$ are part of a valid proof we have that $e(\mathbf{v}, \mathbf{d} - \alpha\mathbf{c}) = t$. Thus $\mathbf{w}^* := \mathbf{d} - \alpha\mathbf{c}$ is a valid witness. This property is known as perfect extractability. Notice that under SXDH an attacker will not be able to tell if the CRS is a DDH tuple and so even in the witness extractable case, the witness remains computationally hidden.

4.3.2 Independent Preimage Resistant Hash Functions

Ideally the proof of security for the scheme would make use of the fact that finding a witness $\mathbf{w}' \neq \mathbf{w}$ so that $e(\mathbf{v}, \mathbf{w}) = t = e(\mathbf{v}, \mathbf{w}')$ is hard. This can be viewed as the idea that the hash function $H_{\mathbf{v}}(\mathbf{y}) = e(\mathbf{v}, \mathbf{y})$ is second preimage resistant. Unfortunately, this would make it impossible to update the secret key. Instead Malkin et al. [52] generalize the notion of second preimage resistance to (ℓ, k)-independent preimage resistance where finding a preimage outside of a particular affine subspace of dimension k is hard. Since we do not need the full details of the definition we will not present them and instead only show that the hash function used inside the proof of security is secure in this sense.

The proof will make use of the fact that we have assumed DDH is hard in G_1. However it will not use this fact directly and instead relies on a related problem.

Definition 4.3.3. Given points $P_1,\ Q_1 \in G_1$, the *double pairing problem* is to find points $P_2,\ Q_2 \in G_2$ so that $e(P_1, P_2) \cdot e(Q_1, Q_2) = ID_T$ and $(P_2, Q_2) \neq 0$.

It is easy to see that any adversary who succeeds in solving this problem with probability ε leads to any adversary who solves DDH in G_1 with probability ε. We know this since given an instance (P, Q, P', Q') of DDH the double pairing problem adversary can compute $R,\ R'$ such that $e(P, R) \cdot e(Q, R') = ID_T$. But then the original instance is a DDH tuple iff $e(P', R) \cdot e(Q', R') = ID_T$.

With this definition in hand, we can move on to showing the desired property of the hash function.

Lemma 4.3.4. *Given* $\mathbf{v} \in G_1^\ell,\ \mathbf{y_1}, \ldots, \mathbf{y}_{\ell-\mathbf{1}} \in G_2^\ell$, *and* $t \in G_T$ *such that* $H_{\mathbf{v}}(\mathbf{y_i}) := e(\mathbf{v}, \mathbf{y_i}) = t$, *it is hard to find* $\mathbf{y}^* \in G_2^\ell$ *so that* $H_{\mathbf{v}}(\mathbf{y}^*) = t$ *and* $\mathbf{y}^* \notin \mathrm{Aff}(\mathbf{y_1}, \ldots, \mathbf{y}_{\ell-\mathbf{1}})$.

Proof. Given a PPT algorithm $\mathcal{A}$ that solves the above problem with probability ε we will construct a PPT algorithm $\mathcal{B}$ which solves the double pairing problem with probability ε. To do this $\mathcal{B}$ first receives an instance (P, Q) of the double pairing problem. It then chooses $\mathbf{a}, \mathbf{b} \in_R \mathbb{Z}_q^\ell$ and $\mathbf{y}_{\ell-\mathbf{1}} \in_R G_2^\ell$ and sets $\mathbf{v} := \mathbf{a}P + \mathbf{b}Q$ and $t := e(\mathbf{v}, \mathbf{y}_{\ell-\mathbf{1}})$. $\mathcal{B}$ then selects $\ell - 2$ vectors $\mathbf{z_j} \in_R \mathbb{Z}_q^\ell$ so that $\mathbf{a} \cdot \mathbf{z_j} = \mathbf{b} \cdot \mathbf{z_j} = 0$ and $Z := \{\mathbf{z_1}, \ldots, \mathbf{z}_{\ell-\mathbf{2}}\}$ is linearly independent. Notice that Z forms a basis for the nullspace of the matrix $\begin{pmatrix} \mathbf{a} & \mathbf{b} \end{pmatrix}$. Next $\mathcal{B}$ selects $W \in_R G_2$

and sets $\mathbf{y_j} := \mathbf{y_{\ell-1}} + \mathbf{z_j}W$, for $1 \leq i \leq \ell - 2$. $\mathcal{B}$ then gives $(\mathbf{v}, t, \mathbf{y_1}, \ldots, \mathbf{y_{\ell-1}})$ to $\mathcal{A}$ as input and receives back $\mathbf{y}^*$. $\mathcal{B}$ sets $\mathbf{z} := \mathbf{y}^* - \mathbf{y_{\ell-1}}$ and $(R, R') = (\mathbf{a} \cdot \mathbf{z}, \mathbf{b} \cdot \mathbf{z})$. Notice that $e(\mathbf{v}, \mathbf{z}) = e(\mathbf{v}, \mathbf{y}^*)/e(\mathbf{v}, \mathbf{y_{\ell-1}}) = ID_T$ if $e(\mathbf{v}, \mathbf{y}^*) = t$. Further if $\mathbf{y}^*$ is affinely independent from $\{\mathbf{y_1}, \ldots, \mathbf{y_{\ell-1}}\}$, then $\mathbf{z}$ is linearly independent from $\{\mathbf{y_1} - \mathbf{y_{\ell-1}}, \ldots, \mathbf{y_{\ell-2}} - \mathbf{y_{\ell-1}}\} = \{\mathbf{z_1}W, \ldots, \mathbf{z_{\ell-2}}W\}$. This in turn implies that $\mathbf{z}'$, the vector consisting of the discrete logs (to the base W) of all the elements in $\mathbf{z}$, is linearly independent from $Z = \{\mathbf{z_1}, \ldots, \mathbf{z_{\ell-2}}\}$. But since Z is basis for the nullspace of $\begin{pmatrix}\mathbf{a} & \mathbf{b}\end{pmatrix}$ we have that $(\mathbf{a} \cdot \mathbf{z}', \mathbf{b} \cdot \mathbf{z}') \neq 0$ and so $(R, R') \neq 0$ (since $\mathbf{z} \neq 0$). Thus (R, R') forms a solution to the double pairing problem and so $\mathcal{B}$ succeeds with the same probability as $\mathcal{A}$. □

4.4 The Scheme

We are almost ready to fully describe the MTVY scheme, but first we need to explain the Waters hash in more detail. The hash is defined in terms of a matrix $H = (H_0, H_1, \ldots, H_n) \in G_2^{2\times(n+1)}$ where the message space is $\{0,1\}^n$. Let m_j be the j^{th} bit of a message m. Then the Waters hash is the function h where

$$h(H, m) = H_0 + \sum_{j=1}^{n} m_j H_j.$$

Throughout the scheme the matrix H will be fixed (but will change in the security proof) and so we will just write $h(H, m)$ as $h(m)$. Notice that $h(m) \in G_2^2$.

Key Generation: To generate her keys, Alice first selects $\mathbf{g} \in_R G_2^2$ and the hash matrix $H = (H_0, \ldots, H_n) \in_R G_2^{2\times(n+1)}$. Next she selects $v \in_R G_1$, $u \in_R G_2$, and $\mathbf{a}, \mathbf{b} \in_R \mathbb{Z}_q^\ell$ such that $\mathbf{a} \cdot \mathbf{b} = 0$. Then she sets $\mathbf{v} := \mathbf{a}v$ and $\mathbf{u} = \mathbf{b}u$. Finally she selects $\mathbf{w} \in_R G_2^\ell$ and computes $t := e(\mathbf{v}, \mathbf{w})$. Her public key is $PK = (\mathbf{g}, H, \mathbf{v}, t, \mathbf{u})$ and her secret key is $SK = \mathbf{w}$.

Key Update: To update her secret key Alice first selects $r \in_R \mathbb{Z}_q^*$. She updates $\mathbf{w} := \mathbf{w} + r\mathbf{u}$ and this new value of $\mathbf{w}$ is her new secret key.

Signing: To sign a message m, Alice first computes the hash $\mathbf{h} := h(m)$. Then using the CRS $(\mathbf{g}, \mathbf{h})$ she computes a Groth-Sahai proof σ on the pair $(\mathbf{v}, t)$ using $\mathbf{w}$ as a witness. Her signature on m is σ.

Verification: To verify Alice's signature σ on a message m, Bob first computes $\mathbf{h} := h(m)$. He then runs the verification component of the proof system using $(\mathbf{g}, \mathbf{h})$ as the CRS. If the verification succeeds he accepts the signature and otherwise he rejects it.

Figure 4.1: The MTVY signature scheme

The MTVY signature scheme is presented in Figure 4.1. Since all the different letters used in the scheme can be confusing, we tried to name things in a way that would be easy

to remember. H is so named because it is the **h**ash matrix and **g** is named because it is always the first component of the CRS and so it always comes before the output **h** of the hash function. The two main components of the public key, which is sometimes called the **v**erification key, are **v** and t which is in G_T. The vector **u** is only used in the key update procedure and so can be thought of as the **u**pdate vector. Finally the secret key **w** acts as a **w**itness in the Groth-Sahai proof system.

Correctness of the MTVY scheme clearly follows from correctness of Groth-Sahai proofs. Also, $e(\mathbf{v}, \mathbf{u}) = e(\mathbf{a}v, \mathbf{b}u) = e(v, u)^{\mathbf{a}\cdot\mathbf{b}} = e(v, u)^0 = ID_t$. Thus, the update portion is also correct since $e(\mathbf{v}, \mathbf{w} + r\mathbf{u}) = e(\mathbf{v}, \mathbf{w}) \cdot e(\mathbf{v}, r\mathbf{u}) = t \cdot e(\mathbf{v}, \mathbf{u})^r = t$ for all $r \in \mathbb{Z}_q^*$.

4.5 Sketch of Proof

In this section we will explore most of the details of the proof of security for MTVY. In particular, we will explain the idea behind the proofs of the various lemmas used in the main security theorem. Where the rigorous details are easy to explain or add additional insight they are presented; otherwise they can be found in [52]. Additionally, we will be explicit about when notions of indistinguishability are used and treat them with extreme caution as we recommended in Section 2.2.7.

We will now state the main security theorem and prove it in terms of a number of claims and lemmas. The rest of this section will provide proofs (or sketches) for these claims except for Lemma 4.6.1 to which we will pay special attention in Section 4.6.

Theorem 4.5.1 ([52])**.** *Under the SXDH assumption the MTVY signature scheme is existentially unforgeable under chosen message attacks in the continual memory bounded-length leakage model with leakage parameters*

$$\left(\frac{c\log k}{\ell\log q},\ \frac{c\log k}{\ell\log q},\ 1-\frac{2+\gamma}{\ell},\ 1-\frac{2+\gamma}{\ell}\right),$$

where $c > 0$ and $\gamma > 2$ are constants.

One thing to note is that Theorem 4.5.1 has slightly worse leakage bounds than the one proved by Malkin et al. in [52]. When we plugged the values into the proof, something that is never done in [52], we found that we needed $\gamma > 2$; in contrast, the authors of [52] require that $\gamma \in \Theta(1/\sqrt{k})$.

Proof. We will prove the theorem by showing that any PPT forger $\mathcal{F}$ that can break the MTVY scheme with non-negligible probability can be used to solve SXDH with non-negligible probability. The first step in showing this is Lemma 4.6.1 which allows us to change the first two leakage parameters both to 0. That is, we continue with the proof as we would for a scheme that allows no leakage during key generation or update.

Suppose $\mathcal{F}$ can forge a signature with probability ε. Let q_u and q_s be upper bounds on the number of update and signature queries made by $\mathcal{F}$, respectively. For $i \in \{1, 2\}$,

let δ_i be the advantage any PPT algorithm has in solving DDH in G_i. We will show that $\mathcal{F}$ can forge a signature in a slightly different signature scheme with only slightly reduced probability. In this scheme the Waters hash is computed in a different way. In particular let $\theta := \lceil q_s/\varepsilon \rceil$ and $\mathcal{X} := \{-n(\theta-1), -n(\theta-1)+1, \ldots, 0\} \times \{0, 1, \ldots, \theta-1\}^n$. Let $\mathbf{x} \in_R G_2^2$, $\mathbf{x}' := (x'_0, \ldots, x'_n) \in_R \mathcal{X}$, $\alpha \in_R \mathbb{Z}_q$, $\mathbf{y} := \alpha\mathbf{g}$, and $\mathbf{y}' = (y'_0, \ldots, y'_n) \in_R \mathbb{Z}_q^{n+1}$. Then we set $H_j := x'_j\mathbf{x} + y'_j\mathbf{y}$ for each $j \in \{0, \ldots, n\}$ and so the hash matrix is $H := (H_0, \ldots, H_n)$. Further, define $K(\mathbf{x}', m)$ and $L(\mathbf{y}', m)$ as

$$K(\mathbf{x}', m) := x'_0 + \sum_{j=1}^{n} x'_j m_j \qquad \text{and} \qquad L(\mathbf{y}', m) = y'_0 + \sum_{j=1}^{n} y'_j m_j \pmod{q}.$$

Note that K is computed over the integers while L is computed over $\mathbb{Z}_q$. With this notation we can rewrite $h(H, m) = K(\mathbf{x}', m)\mathbf{x} + L(\mathbf{y}', m)\mathbf{y}$.

We say $\mathcal{F}$ is only successful if for every message m whose signature it queried we have $K(\mathbf{x}', m) \neq 0$ and $K(\mathbf{x}', m^*) = 0$ holds for the message m^* whose signature it forged. We will call a forgery with this property a "special forgery". Lemma 4.5.2 shows that $\mathcal{F}$ succeeds in producing a special forgery in this new scheme with probability at least $\varepsilon^2/(nq_s) - \delta_2$.

Notice that when a special forgery is produced, the CRS computed in all the signature queries made by $\mathcal{F}$ ensures that the Groth-Sahai proof is witness indistinguishable. This happens since $h(H, m) = K(\mathbf{x}', m)\mathbf{x} + L(\mathbf{y}', m)\mathbf{y}$ and $K(\mathbf{x}', m) \neq 0$ and so by choice of $\mathbf{g}$, $\mathbf{x}$ and $\mathbf{y}$ the CRS is only a DDH tuple when $\mathbf{x} = \alpha'\mathbf{g}$ for some $\alpha' \in \mathbb{Z}_q$ which happens with probability $1/q$. Further, since $K(\mathbf{x}', m^*) = 0$ we have that $h(H, m^*) = L(\mathbf{y}', m^*)\mathbf{y} = L(\mathbf{y}', m^*)\alpha\mathbf{g}$. Thus the CRS used in the forgery made by $\mathcal{F}$ is $(\mathbf{g}, L(\mathbf{y}', m^*)\alpha\mathbf{g})$ and so a witness can be extracted from it, provided α is known. We will use these two properties later in the proof.

Next we change how the witness $\mathbf{w}$ is selected. The vectors $\mathbf{v}$ and $\mathbf{u}$ are selected the same way as before (in particular $\mathbf{v} = \mathbf{a}v$). In addition, vectors $\alpha_{\mathbf{1}}, \ldots, \alpha_{\ell-\mathbf{3}} \in_R \mathbb{Z}_q^\ell$ and an element $z \in_R G_2$ are selected so that $e(\mathbf{v}, \mathbf{z_j}) = ID_T$ for all j, where $\mathbf{z_j} = \alpha_{\mathbf{j}} z$ (this is done by selecting $\alpha_{\mathbf{j}}$'s so that $\alpha_{\mathbf{j}} \cdot a = 0$). Further $\mathbf{s} \in_R G_2^\ell$ is selected and now $t := e(\mathbf{v}, \mathbf{s})$. Let $\mathcal{W} = \mathbf{s} + \text{span}(\mathbf{z_1}, \ldots, \mathbf{z}_{\ell-\mathbf{3}})$. The secret key $\mathbf{w}$ is now selected as a random element of $\mathcal{W}$ in key generation and update. Lemma 4.5.6 shows that $\mathcal{F}$ produces a special forgery in this new scheme with probability at least $\varepsilon^2/(nq_s) - 2\delta_2$. Lemma 4.5.9 shows that the new scheme is statistically indistinguishable from one where the secret key is chosen as a random element of the set $\mathcal{S} := \{\mathbf{w} \in G_2^\ell \mid e(\mathbf{v}, \mathbf{w}) = t\}$. Notice that $|\mathcal{S}| \geq q^{\ell-1}$ and $|\mathcal{W}| \leq q^{\ell-3}$. Thus the probability that a randomly selected element of $\mathcal{S}$ is in $\mathcal{W}$ is at most q^{-2}. Therefore, if a secret key can be recovered from the forgery that $\mathcal{F}$ produces, then with high probability it will not be in the affine space $\mathcal{W}$.

Finally we show how a forger for the new scheme violates the property of the hash function $h_{\mathbf{v}}(\mathbf{w}) = e(\mathbf{v}, \mathbf{w})$ proved in Lemma 4.3.4. Algorithm $\mathcal{A}$ runs $\mathcal{F}$ as a subroutine by choosing all the parameters of the new scheme correctly and sending the public information to $\mathcal{F}$. Since it knows the secret information it can answer signing and leakage queries. At the end $\mathcal{F}$ outputs a forgery (m^*, σ). We know that by construction the CRS used in the

proof σ is a DDH tuple $(\mathbf{g}, L(\mathbf{y}', m^*)\alpha\mathbf{g})$. Further, since $\mathcal{A}$ knows α it can extract a witness $\mathbf{w}^*$ from σ as discussed in Section 4.3.1. From the argument above, however, we can see that with high probability the witness $\mathbf{w}^*$ is not in the affine space $\mathcal{W}$. But then $\mathcal{A}$ has violated the independent preimage resistance of $h_{\mathbf{v}}(\mathbf{w}) = e(\mathbf{v}, \mathbf{w})$.

In Lemma 4.5.9, we see that the probability that the subspace $\mathcal{W}$ is not hidden from the view of $\mathcal{F}$ is at most $q^{1-\gamma/2}/q_u$. Thus the success probability ρ of $\mathcal{A}$ is

$$\begin{aligned}\rho &= \Pr[\mathcal{F} \text{ produces a special forgery in the new scheme}] - q^{1-\gamma/2}/q_u \\ &\geq \varepsilon^2/(nq_s) - 2\delta_2 - q^{1-\gamma/2}/q_u - \text{negl}(k),\end{aligned}$$

where $\text{negl}(k)$ is bounded by some polynomial p evaluated at q^{-1}. The $\text{negl}(k)$ is a tiny factor (approximately 2^{-2k}) and has no bearing on the meaning of the proof in practice. □

Lemma 4.5.2. *Any PPT forger $\mathcal{F}$ for the MTVY signature scheme who succeeds in forging a signature with probability ε succeeds in forging a special signature for the scheme where the Waters hash has been changed as described above with probability at least $\varepsilon^2/(nq_s) - \delta_2$.*

Proof. The proof of this lemma is broken into three claims. When proving these claims it is important to keep in mind that *no leakage occurs during key generation.*

Claim 4.5.3. *Suppose the matrix H for the Waters hash is computed as follows. Set $\theta = \lceil q_s/\varepsilon \rceil$, and $\mathcal{X} = \{-n(\theta-1), -n(\theta-1)+1, \ldots 0\} \times \{0, 1, \ldots, \theta-1\}^n$. Let $\mathbf{x},\ \mathbf{y} \in_R G_2^2$, $\mathbf{x}' := (x'_0, \ldots, x'_n) \in_R \mathcal{X}$, $\mathbf{y}' = (y'_0, \ldots, y'_n) \in_R \mathbb{Z}_q^{n+1}$. Then we set $H_k := x'_k\mathbf{x} + y'_k\mathbf{y}$ for each $k \in \{0, \ldots, n\}$ and so the hash matrix is $H := (H_0, \ldots, H_n)$. Notice that $\mathbf{y}$ is chosen randomly, which is different from how it is chosen in the actual modified scheme used in the proof of security. Using this Waters hash instead of the one defined by MTVY does not change the success probability of $\mathcal{F}$.*

Proof of Claim 4.5.3. Notice that each H_k is still a random element of G_2^2 and thus this definition does not change the distribution of H. Since no leakage occurs during key generation, $\mathcal{F}$ cannot possibly tell that the parameters have been generated differently and so this change cannot alter its success probability. □

Claim 4.5.4. *Recall $K(\mathbf{x}', m) := x'_0 + \sum_{j=1}^{n} x'_j m_j$. A special sequence of messages is one where every message m submitted as a signature query satisfies $K(\mathbf{x}', m) \neq 0$ and the forgery is on a message m^* where $K(\mathbf{x}', m) = 0$. When the Waters hash in Claim 4.5.3 is used, $\mathcal{F}$ produces a special signature with probability at least $\varepsilon^2/(nq_s)$*

Sketch of proof of Claim 4.5.4. Two lemmas from [5] give a lower bound on the probability of any given sequence of messages satisfying the special property and show that the probability that a particular sequence is used is independent from the probability that it is special. In sum we have that

$$\Pr[\text{special}] \geq \frac{1-\varepsilon}{n(q_s/\varepsilon - 1) + 1} \qquad \text{and} \qquad \Pr[\text{special and chosen}] = \Pr[\text{special}] \cdot \Pr[\text{chosen}].$$

Notice also that the probability of a sequence being special is dependent only on $\mathbf{x}$ whose distribution is hidden from $\mathcal{F}$ (once again because no leakage occurs during key generation). Thus the probability of $\mathcal{F}$ producing a special sequence of messages is independent from its probability of producing a valid forgery. So we have

$$\begin{aligned}
\Pr[\text{valid and special}] &= \Pr[\text{valid}] \cdot \Pr[\text{special}] \\
&= \varepsilon \cdot \sum_{\substack{\text{message} \\ \text{sequences}}} \Pr[\text{special and chosen}] \\
&= \varepsilon \cdot \sum_{\substack{\text{message} \\ \text{sequences}}} \Pr[\text{special}] \cdot \Pr[\text{chosen}] \\
&\geq \varepsilon \cdot \sum_{\substack{\text{message} \\ \text{sequences}}} \frac{1-\varepsilon}{n(q_s/\varepsilon - 1) + 1} \Pr[\text{chosen}] \\
&= \varepsilon \cdot \frac{1-\varepsilon}{n(q_s/\varepsilon - 1) + 1} \sum_{\substack{\text{message} \\ \text{sequences}}} \Pr[\text{chosen}] \\
&\geq \varepsilon \cdot \frac{1-\varepsilon}{n(q_s/\varepsilon - 1) + 1} \cdot 1 \\
&\geq \frac{\varepsilon^2(1-\varepsilon)}{n(q_s - \varepsilon)} \\
&\geq \frac{\varepsilon^2}{nq_s}
\end{aligned}$$

as desired. □

Claim 4.5.5. *Suppose that the Waters hash for a scheme otherwise identical to MTVY is generated as in Claim 4.5.3 except that* $\mathbf{y}$ *is generated differently. In particular,* $\alpha \in_R \mathbb{Z}_q$ *is selected and then* $\mathbf{y} := \alpha\mathbf{g}$*. $\mathcal{F}$ succeeds in producing a special forgery in this scheme with probability only negligibly different than in Claim 4.5.4.*

Proof of Claim 4.5.5. Suppose $\mathcal{F}$ makes a special forgery in the setting of Claim 4.5.4 with probability δ and in the setting of this claim with probability δ'. Then clearly we have that $\delta_2 \geq |\delta - \delta'|$ since we can create an algorithm for DDH in G_2 based on whether or not $\mathcal{F}$ succeeds. In other words we have $\delta' \geq \delta - \delta_2 = \delta - \text{negl}(k)$ since DDH is assumed to be hard. Here it is crucial that no leakage occurs during key generation, since even a single bit of leakage could be enough to solve DDH. □

Putting the three claims together we can see that $\mathcal{F}$ produces a special forgery for MTVY with the altered Waters hash with probability at least $\varepsilon^2/(nq_s) - \delta_2$. □

Lemma 4.5.6. *Consider the MTVY signature scheme altered in the following way. First the hash function is changed as in Lemma 4.5.2. Next, vectors* $\mathbf{z_1}, \ldots, \mathbf{z}_{\ell-\mathbf{3}} \in_R \mathbb{G}_2^\ell$ *and* $\mathbf{s} \in_R G_2^\ell$ *are selected (as described in the proof of Theorem 4.5.1 above) so that* $e(\mathbf{v}, \mathbf{z_i}) = ID_T$ *for all* i *and* $t := e(\mathbf{v}, \mathbf{s})$*. Let* $\mathcal{W} = \mathbf{s} + \text{span}(\mathbf{z_1}, \ldots, \mathbf{z}_{\ell-\mathbf{3}})$*. The secret key* $\mathbf{w}$ *is now selected as a random element of* $\mathcal{W}$ *in key generation and update. A forger $\mathcal{F}$ for MTVY that succeeds with probability* ε*, succeeds in this new scheme with probability at least* $\varepsilon^2/(nq_s) - 2\delta_2$*.*

Proof. To begin this proof we first state a well-known result. Consider the problem where an adversary must determine whether a given vector was generated randomly or in the following way. First $\mathbf{z'_{\ell-2}} \in_R G_2^{\ell-1}$ and $\beta_1, \ldots, \beta_{\ell-3} \in_R \mathbb{Z}_q$ are selected. Then $\mathbf{z'_i} := \beta_i \mathbf{z'_{\ell-2}}$ is computed for $1 \leq i \leq \ell - 3$ and the vector output is $(\mathbf{z'_1}, \ldots, \mathbf{z'_{\ell-2}})$. We will use the fact that the advantage of any PPT adversary for this problem is at most δ_2. An intuition for why this is true can be seen in the following way. Given the DDH challenge $C := (P, Q, xP, yQ)$ we can create any number of independent challenges C_i, by setting

$$C_i := (a_i P + b_i Q, c_i P + d_i Q, a_i x P + b_i y Q, a_i x P + b_i y Q),$$

where $a_i, b_i, c_i, d_i \in_R \mathbb{Z}_q$. It can be verified that whenever C is a DDH tuple (i.e. $x = y$) then so is C_i and whenever C is random then so is C_i. Further we can easily extend this technique to make these challenges contain an arbitrary number of elements (i.e. they can be of the form $(P_1, \ldots, P_\ell, x_1 P_1, \ldots, x_\ell P_\ell)$ for any choice of $\ell \geq 2$).

We will now construct an algorithm $\mathcal{A}$ that uses a forger $\mathcal{F}$ for MTVY to solve the above problem. Basically $\mathcal{A}$ will construct inputs for the forger based on a challenge for the above problem in such a way that if the challenge was random then the secret key will be selected as it is in the MTVY scheme where only the hash is modified and otherwise it will be selected as described in the statement of this lemma.

$\mathcal{A}$ takes as input a challenge $(\mathbf{z'_1}, \ldots, \mathbf{z'_{\ell-2}})$ for the above problem. It generates the parameters for MTVY as follows. First it generates $\mathbf{v}$ by selecting $v \in_R G_1$ and $a_1, \ldots, a_{\ell-1} \in_R \mathbb{Z}_q$ and setting $\mathbf{v} := (a_1 v, \ldots, a_{\ell-1} v, v)$. Then for each $1 \leq i \leq \ell - 2$ it sets

$$\mathbf{z_i} := \left(z'_{i,1}, \ldots, z'_{i,\ell-1}, -\sum_{j=1}^{\ell-1} a_j z'_{i,j} \right)$$

where $\mathbf{z'_i} = (z'_{i,1}, \ldots, z'_{i,\ell-1})$. Further it sets $\mathbf{u} := \mathbf{z_{\ell-2}}$. Notice that by definition of the $\mathbf{z_i}$'s we have

$$e(\mathbf{v}, \mathbf{z_1}) = \cdots = e(\mathbf{v}, \mathbf{z_{\ell-2}}) = e(\mathbf{v}, \mathbf{u}) = ID_T.$$

$\mathcal{A}$ now selects $\mathbf{s} \in_R G_2^\ell$ and sets $\mathbf{w}$ to be a random element of $\mathcal{W} := \mathbf{s} + \text{span}(\mathbf{z_1}, \ldots, \mathbf{z_{\ell-3}})$ for both key generation and updates. It proceeds with the remainder of key generation as normal and gives the public key to $\mathcal{F}$ as input. Since $\mathcal{A}$ has access to the secret key, it can faithfully respond to all queries from $\mathcal{F}$.

Clearly when $(\mathbf{z'_1}, \ldots, \mathbf{z'_{\ell-2}})$ is random, this is exactly the modified key generation method in the hypothesis of the lemma. Further when $\mathbf{z'_i} = \beta_i \mathbf{z'_{\ell-2}}$ for some $\beta_i \in \mathbb{Z}_q$, it is clear that $\mathbf{z_i} = \gamma_i \mathbf{z_{\ell-2}} = \gamma_i \mathbf{u}$ for some $\gamma_i \in \mathbb{Z}_q$. Hence, if the challenge is not random, $\mathcal{W} = \mathbf{s} + \text{span}(\mathbf{u})$. But this is exactly the set from which $\mathbf{w}$ is picked in the version of MTVY where only the hash is modified. Thus if $\mathcal{A}$ simply outputs 1 based on whether or not the forger is successful we see that $\mathcal{A}$ has advantage $|\varepsilon^2/(nq_s) - \delta_2 - \varepsilon'|$ in solving the above problem where ε' is $\mathcal{F}$'s probability of success for MTVY with the modified key generation. But then from the fact discussed above we have $|\varepsilon^2/(nq_s) - \delta_2 - \varepsilon'| \leq \delta_2$ and so $\varepsilon' \geq \varepsilon^2/(nq_s) - 2\delta_2$, as claimed. □

Before we can prove the final lemma used in the security proof, we need to state a theorem from [11] about random subspaces. It will be helpful to recall the notion of statistical distance we discussed in Remark 4.3.1.

Theorem 4.5.7 ([11])**.** *Let $\ell \geq 5$, Q be a finite set, $\mathcal{K}$ an ℓ-dimensional vector space over $\mathbb{Z}_q$, and $f : \mathcal{K} \to Q$ be an arbitrary function. Let $Z = (\mathbf{z_1}, \ldots, \mathbf{z_{\ell-3}}) \in_R \mathcal{K}^{\ell-3}$ and set $\mathcal{Z} := \operatorname{span}(\mathbf{z_1}, \ldots, \mathbf{z_{\ell-3}})$. Let $\mathbf{v} \in_R \mathcal{Z}$ and $\mathbf{u} \in_R \mathcal{K}$. Then*

$$\operatorname{dist}((Z, f(\mathbf{v})), (Z, f(\mathbf{u}))) \leq \delta$$

whenever $|Q| \leq q^{\ell-4}\delta^2$.

Note that this is actually a special case of the theorem in [11] which is slightly more general. We will use this theorem to show that the following "subspace game" is hard.

Lemma 4.5.8. *We will use the same notation as in Theorem 4.5.7. Additionally, let $\mathbf{v_i} \in_R \mathcal{Z}$ and $\mathbf{u_i} \in_R \mathcal{K}$ for $1 \leq i \leq q_u$. A challenger selects a random bit and if it is 1 sets $\mathbf{w_i} := \mathbf{u_i}$ and otherwise sets $\mathbf{w_i} := \mathbf{v_i}$. The adversary adaptively selects leakage queries $f_1, \ldots, f_{q_u}$ and receives $f_i(\mathbf{w_i})$ for all i. The adversary then outputs a bit; its advantage is the difference between the probability it outputs 1 in the two different cases. The advantage of any (unbounded) adversary for this game is at most δ whenever $|Q| \leq q^{\ell-4}(\delta/q_u)^2$.*

Sketch of Proof. The idea is to feed an adversary $\mathcal{A}$ for the subspace game the result of its query on a random element of $\mathcal{K}$ for the first $i-1$ queries and a random element of $\mathcal{Z}$ for the remaining queries. To ease notation we will say that for $1 \leq j \leq i-1$ the query occurs on $\mathbf{w_j} \in_R \mathcal{K}$ and for $i \leq j \leq q_u$ the query occurs on $\mathbf{w'_j} \in_R \mathcal{Z}$. Let f_j be the j^{th} query made by $\mathcal{A}$. Then we set D_i to be the distribution

$$((Z, f_1(\mathbf{w_1})), \ldots, (Z, f_{i-1}(\mathbf{w_{i-1}})), (Z, f_i(\mathbf{w'_i})), \ldots, (Z, f_{q_u}(\mathbf{w'_{q_u}}))).$$

Suppose that $\mathcal{A}$ succeeds with probability ε. We then have that

$$\varepsilon \leq \operatorname{dist}(D_1, D_{q_u+1}) \leq \sum_{i=1}^{q_u} \operatorname{dist}(D_i, D_{i+1})$$

by the triangle inequality. Thus, for some i we must have that $\operatorname{dist}(D_i, D_{i+1}) \geq \varepsilon/q_u$. Notice, however, that the only difference in the two distributions D_i and D_{i+1} is that in the first case f_i is given an element of $\mathcal{K}$ and in the second it is given an element of $\mathcal{Z}$. But from Theorem 4.5.7, this must mean that the two distributions are at most $\sqrt{|Q|/q^{\ell-4}}$ apart. Since $|Q| \leq q^{\ell-4}(\delta/q_u)^2$ this means that $\varepsilon/q_u \leq \operatorname{dist}(D_i, D_{i+1}) \leq \delta/q_u$ and so $\varepsilon \leq \delta$. □

Lemma 4.5.9. *The signature scheme defined in Lemma 4.5.6 is statistically indistinguishable from one in which the key is chosen as a random element from the set of all preimages of t (i.e. $\mathbf{w} \in_R \mathcal{S} := \{\mathbf{w} \mid e(\mathbf{v}, \mathbf{w}) = t\}$) under the assumption that $\mathcal{F}$ produces a successful forgery.*

Sketch of proof. Here we give the main ideas of the proof although we will omit many details. We will use a forger $\mathcal{F}$ to create an adversary $\mathcal{A}$ for the subspace game. Suppose $\mathcal{F}$ succeeds with probability ε when $\mathbf{w}$ is chosen randomly from $\mathcal{W}$ and ε' when it is chosen randomly from $\mathcal{S}$. The main idea we will use is that since $\mathcal{A}$ is unbounded, it will able to be generate signatures without knowledge of a secret key. To generate leakage queries it will rely on the leakage oracle for the subspace game. It can do this since, by construction of the special Waters hash and the requirements on $\mathcal{F}$, signature queries requested by $\mathcal{F}$ always use a non-DDH CRS. In particular, this means the corresponding proofs are perfectly witness indistinguishable. Thus for any choice of witness and randomness $(\mathbf{w}, R)$ producing a proof σ, for every other witness $\mathbf{w}'$ there exists *unique* randomness R' so the proof using $(\mathbf{w}', R')$ and the same CRS also equals σ.

$\mathcal{A}$ generates all parameters for the modified MTVY scheme except the vector $(\mathbf{z_1}, \ldots, \mathbf{z}_{\ell-\mathbf{3}})$ and sends them to $\mathcal{F}$. Let us first imagine a case where $\mathcal{F}$ can only make one signature query (m, f) per update. In this case $\mathcal{A}$ computes the CRS normally and selects a random element σ from the set of all proofs using that CRS. To answer the leakage query it has the challenger for the subspace game:

1. Generate the appropriate vector $\mathbf{w_i}$ for the round.
2. Compute $\mathbf{w} := \mathbf{s} + \mathbf{w_i}$ and find the randomness R so that the proof computed using $\mathbf{w}, R$, and the CRS is σ.
3. Output $f(\mathbf{w}, R)$.

We will attempt to give an idea of what to do in the general case (note that Malkin et al. do not describe what to do in this situation saying only that it "follows in a straightforward way"). Unfortunately, the method we were able to devise is not very straightforward due to the possibility of leakage and signature queries being adaptive and the fact that $\mathcal{A}$ can only query the challenger once per update. Very roughly speaking, in the general case, $\mathcal{A}$ examines the description of $\mathcal{F}$ and determines what messages could be possibly signed between a particular pair of key updates. It computes signatures for all these messages as described above. It then feeds these signatures into the challenger as well as the description of $\mathcal{F}$ for that time period. It has the challenger generate the key for that round as before. The challenger can calculate the output of the various leakage functions (calculating the randomness for the signatures as described above). At the end it outputs the concatenation of the appropriate results of the leakage functions. By examining this output and the description of $\mathcal{F}$, $\mathcal{A}$ can determine which signatures were requested and send them to $\mathcal{F}$ one-by-one along with the appropriate results of leakage queries.

Since no leakage can occur during key updates, $\mathcal{A}$ does not need to do anything for update queries. In the end it outputs 1 depending on whether or not $\mathcal{F}$ was successful. Further, by our observation above about witness indistinguishability for Groth-Sahai proofs the distribution of $(\sigma, f(\mathbf{w}, R))$ generated by $\mathcal{A}$ for $\mathcal{F}$ is correct. Furthermore, we see that whenever the challenger's random bit is 0, the secret keys are always chosen from $\mathcal{S}$, and whenever it is 1, they are always chosen from $\mathcal{W}$. Therefore $\mathcal{A}$ succeeds at the subspace game with advantage $\delta := |\varepsilon - \varepsilon'|$.

By assumption on the forger, the number of bits requested from the challenger at each round is $(1-(2+\gamma)/\ell)\cdot\ell\cdot\log q$. Thus $|Q| = q^{\ell-2-\gamma} = q^{\ell-4}q^{2-\gamma}$. So we can see that setting $q^{\ell-4}(\delta/q_u)^2 = q^{\ell-4}q^{2-\gamma}$ we have $\delta = q^{1-\gamma/2}\cdot q_u$, which is negligible for any $\gamma > 2$ since q_u is polynomial in k and q is exponential in k. Thus, by definition of δ, the difference in the success probability of $\mathcal{F}$ in the two cases is negligible, as required. □

4.6 Something Out of Nothing

In this section we will take an in-depth look at the lemma that allows leakage during key generation and updates. In particular we will consider what it means in practice, especially as it relates to the MTVY scheme. The lemma (although not explicitly stated) first appeared as a part of the security proof for the encryption scheme of Brakerski et al. [11]. We will state the lemma in terms of a signature scheme, but the authors of [18] cite personal communication with Waters stating that it holds for any one-way relation.

Lemma 4.6.1 ([11, 52])**.** *Any signature scheme that is secure in the continual memory bounded-length leakage model with leakage parameters $(0, 0, L_S, L_M)$, security parameter k, and secret key size S, is also secure in the same model with leakage parameters*

$$\left(\frac{c\cdot\log k}{S}, \frac{c\cdot\log k}{S}, L_S, L_M\right)$$

for any $c > 0$ provided that L_M is $\omega(\log k/|SK|)$.

Notice that this lemma allows for $c\cdot\log k$ *bits* of leakage during key generation and update. We will not give the details of the proof since they are not much more enlightening than the proof idea. The idea is to take a forger $\mathcal{F}$ which is permitted to issue leakage queries during key generation and update and turn it into a forger $\mathcal{A}$ which does not. We run $\mathcal{F}$ and each time it asks for leakage during key generation or update we record its state and make the following memory leakage query to the challenger for $\mathcal{A}$:

1. Let λ be one of the possible outputs of $\mathcal{F}$'s leakage query. Run another copy $\mathcal{F}'$ of $\mathcal{F}$ using the recorded state and λ as the answer to its leakage query. Answer further leakage queries normally (so $\mathcal{F}'$ is run to completion). Repeat this process (i.e. running $\mathcal{F}'$ with λ as the answer to its leakage query) k times using fresh randomness.
2. Repeat step 1 for each of the $2^{c\log k} = k^c$ possibly values of λ.
3. Output the leakage value that leads to the highest number of successful forgeries.

Notice that since c is constant, k^c is polynomial in k, and so this leakage function is polytime-computable. The rest of the proof is a statistical argument that the new forger $\mathcal{A}$ succeeds with probability at least one half that of $\mathcal{F}$.

The first thing to notice about this proof is that the result for key generation is trivial. In particular for leakage that only happens once, one trying to convert $\mathcal{F}$ into $\mathcal{A}$ can

simply guess a value for this leakage. The reason why this is trivial requires some further terminology. The *tightness* of a proof (of security) refers to how close the probability of breaking the scheme is to the probability of breaking the underlying hard problem. That is, suppose that if an attacker can break a scheme with probability ε, then she can solve, say, DLOG with probability $f(\varepsilon)$ (in the same number of steps). A proof is tight if $f(\varepsilon)$ is very close to ε. In principle, if a proof is non-tight the key size should be increased so that the difficulty, based on the security proof, of solving the underlying hard problem meets the requirements of the security parameter. In practice, this is never done and so non-tight proofs could lead to schemes which are insecure. Fortunately, such proofs are almost always secure, in the sense that there does not appear to be a way to exploit the lack of tightness.

All that being said, the loss of tightness that results from guessing all possible values of leakage during key generation is meaningful. Otherwise, any scheme could reduce its key size by $c \cdot \log k$ bits without incurring any loss in security (by revealing $c \cdot \log k$ bits of the secret key). Thus the need to perform k^{c+1} operations to permit leakage of $c \cdot \log k$ bits is a meaningful loss in tightness. However, the key update leakage is not subject to this argument. That is, if we simply guess all values for the leakage in key updates, our new procedure would be guaranteed to succeed with probability k^{-cq_u} times the probability of success of $\mathcal{F}$, which is negligible. Nonetheless, we feel that the very nature of a secure (signature) scheme with key updates may trivially allow this key update leakage.

We may assume that any secret information which an attacker learns over time through leakage is initially given to her (that is, any unchanging secret values can be given to the attacker in the first place). Further, it is reasonable to expect that leakage about the secret key SK_i reveals no information about the secret key SK_{i+1}. Basically, the intuition is that old leakage is almost entirely unhelpful after the secret key has been updated. Thus it can be viewed as if each time the secret key is updated the entire system is re-keyed. While this overstates the security imbued by key updates, something almost as strong *must* hold, otherwise the system would get weaker and weaker after each update occurred.[1] But, if key updates are equivalent to rekeying, then the leakages allowed under Lemma 4.6.1 are, in some sense, independent. In other words, a wrong guess for one leakage value will only alter the probability of success of $\mathcal{F}$ in the current round. This conclusion makes sense since, because of the availability of memory leakages, the only advantage $\mathcal{F}$ gets by receiving leakage during key updates is the ability to learn some function of the randomness used for those particular updates. In other words, the update leakages only reveal information about a particular key and not about the set of keys corresponding to the public key.

Combining these assertions with our observations about the leakage during key generation, we see that once again the loss of tightness needed to achieve provable security despite key update leakage is meaningful. Put another way, the proofs of Lemma 4.6.1 found in [11, 52] provide no assurances that leakage should actually be allowed during key generation or update.

We might not be the first to have these concerns either. In [18], the authors note that their leakage parameters could include leakage during key generation and updates by

[1]For MTVY we see that the number of updates only alters security if q_u is almost as big as $q^{1-\gamma/2}$.

applying the lemma. Boyle et al. [10] and Lewko et al. [51] note the existence of the lemma but specifically do not apply it to their schemes. While we are not sure of the reasoning behind the authors' decisions to not include the lemma, it is curious that they would not take the opportunity to add "free leakage" if they felt it was meaningful to do so.

Even if we were to disregard the above, what does Lemma 4.6.1 tell us in practice? To make things concrete we will look at how it applies to the MTVY scheme at the 128-bit security level. The first thing we need to do is determine what value of c is appropriate. Brakerski et al. suggest that $c = 1$ is a reasonable choice [11], and we are not inclined to make it much higher because even for $c = 2$ the proof described above results in a tightness loss by a factor of over a million. When $c = 1$ and $k = 128$ the lemma shows MTVY to be secure even if it leaks 7 bits during key generation and update. It is hard to imagine a scenario in which an implementer is worried about these procedures leaking information, but not more than 7 bits. We will discuss the ramifications of only allowing such few bits of leakage in Section 4.7.1.

There may be another very subtle possible problem with using Lemma 4.6.1 in conjunction with MTVY. In particular, if δ_i is the advantage any PPT algorithm has in solving DDH in G_i and ε is the probability of success of some PPT forger $\mathcal{F}$, then the proof of security tells us that $\delta_1 \geq f(\varepsilon) - 2\delta_2$ for some non-negligible function f. However, an adversary who can observe the selection of DDH parameters (i.e. one that gets leakage) can trivially solve DDH if this observation reveals whether or not the output is a DDH tuple. Notice that even a single bit of leakage suffices to totally compromise DDH security. Furthermore, the security proof for MTVY essentially consists of using $\mathcal{F}$ to construct an algorithm for solving DDH in G_1. This algorithm only ever gives to $\mathcal{F}$ values that depend on the first two elements of a DDH challenge. Thus, even if $\mathcal{F}$ had access to leakage during key generation, it would not be able to tell if the challenge was a DDH tuple. However, the proof uses the fact that DDH is hard in G_2. Moreover, it does this in such a way that with leakage, an adversary would be able to tell whether or not certain parameters it was given form a DDH tuple. Thus, we cannot use the security proof directly to show that some leakage can be allowed during key generation for MTVY.

So now suppose $\mathcal{F}'$ is a forger that requires a single bit of leakage during key generation. Further, suppose $\mathcal{F}'$ creates a special forgery with probability ε'. We now create a forger $\mathcal{F}$ that runs $\mathcal{F}'$ making a random guess for the value of the leakage bit and outputs the signature produced by $\mathcal{F}$. It is clear that $\mathcal{F}$ produces a special forgery with probability $\varepsilon \geq \varepsilon'/2$. In those cases where $\mathcal{F}$ succeeds, it can still be used to solve DDH in G_1, but this does not necessarily mean that ε must be negligible. In particular, there seems to be no way of knowing that $\mathcal{F}$ does not solve DDH in G_2 in those cases where its guess for the bit is correct. Notice that if this were the case $\mathcal{F}$ may only have negligible advantage in solving DDH in G_2, but could still produce a forgery with non-negligible probability. It is very hard to tell whether the loss in tightness incurred by the proof is enough to make its conclusions meaningless when paired with the DDH assumption in this way. Further research is required to determine the applicability of Lemma 4.6.1 to schemes which use decisional hardness assumptions in this way.

While the proof *may* not say anything in the case where leakage occurs during key

generation and update, we do not necessarily think that this scheme is insecure when this extra amount of leakage is allowed. In particular, solving DDH in G_2 does not immediately lead to an attack on the scheme. This is normal for schemes based on decisional problems like DDH. Even with MTVY, the proof of security ends up using the forger to solve a computational problem which is likely harder than DDH (in the sense that it could not be solved even with access to a DDH oracle). We again emphasize that theoreticians should use extreme caution when dealing with (decisional) indistinguishability and leakage. The argument above shows both that it is possible to base leakage resilient schemes (even ones with leakage during key generation) on decisional hardness problems like DDH, but also that extreme care should be used when doing so.

4.7 Performance Analysis

We will now attempt to analyze the performance characteristics of MTVY, including its leakage resilience. We will do so as if Lemma 4.6.1 made sense for this scheme. In particular we will allow a small amount of leakage during key generation and update. Doing so allows us to examine the security of the scheme as expected by Malkin et al. [52]. As usual we will assume that $|q| = 2k$. In practice, asymmetric pairings at the 128-bit security level are implemented on Barreto-Naehrig (BN) curves. With these curves one input group has coordinates of size $|q|$ and the other of size $|q^2|$. The output group has elements of size $|q^{12}|$. So for this scheme we may assume that elements of G_1 can be represented with $|q^2| = 4k$ bits, elements of G_2 can be represented using $|q| = 2k$ bits, and elements of G_T can be represented using $|q^{12}| = 24k$ bits.

The public key consists of $\mathbf{g}$ which is two elements of G_2, H which is $2n$ elements of G_2, $\mathbf{v}$ and $\mathbf{u}$ which are ℓ group elements each in G_1 and G_2 respectively, and t which is 1 G_T element. Since, in practice, messages are hashed by a collision-resistant hash function before they are signed, we can set n, the size of the message space, equal to $2k$. In total, the public key is comprised of ℓ elements in G_1, $4k + \ell + 2$ elements of G_2 and 1 element of G_T. The secret key consists of ℓ elements of G_2. A total of $(4 + 4k + 3\ell)2k$ bits of randomness are needed for key generation as well as ℓ point multiplications each in G_1 and G_2 and one pairing computation.

In key update, $2k$ random bits are needed and ℓ point multiplications in G_2 are computed. Signing requires $4\ell k$ random bits, 2ℓ point multiplications in G_1, and 4ℓ point multiplications in G_2. The signature consists of 2 elements in G_1 and 2ℓ elements in G_2. The verification algorithm consists of $2\ell + 4$ pairing computations. When evaluating efficiency, we will use the rough estimate that 1 exponentiation in G_1 costs 4 exponentiations in G_2.

In terms of leakage, we have seen that we desire $q^{(1-\gamma/2)} \cdot q_u \ll \varepsilon^2/(2kq_s)$. For $k = 128$ and $q_s = q_u = 1/\varepsilon = 2^{32}$ we can see that this happens when γ is a little more than 3. So we will choose $\gamma = 3.25$. As discussed earlier, we will set $c = 1$ and so then the *number of bits* that can be leaked at each stage is

$$(\log k, \log k, 2k(\ell - 5.25), 2k(\ell - 5.25)) .$$

4.7.1 Comparison with Schnorr-ℓ

The next natural step is to compare MTVY to the schemes we have already looked at. Table 4.1 shows the various parameters of MTVY and Schnorr-ℓ for a few different values of ℓ. In the efficiency columns M refers to a point multiplication and P a pairing computation.

ℓ	**Scheme**	**Size**			**Master leakage**		**Efficiency**	
		PK	SK	Signature	Bits	Rate	Sign	Verify
6	MTVY	139264	1536	4096	192	0.125	72M	16P
	Schnorr-ℓ	1792	1536	1792	576	0.375	6M	7M
7	MTVY	140032	1792	4608	448	0.25	84M	18P
	Schnorr-ℓ	2048	1792	2048	704	0.3929	7M	8M
8	MTVY	140800	2048	5102	704	0.34375	96M	20P
	Schnorr-ℓ	2304	2048	2304	832	0.4063	8M	9M
50	MTVY	173056	12800	26624	11456	0.895	600M	104P
	Schnorr-ℓ	13056	12800	13056	6208	0.485	50M	51M
100	MTVY	211456	25600	52224	24256	0.9475	1200M	204P
	Schnorr-ℓ	23856	25600	23856	12608	0.4925	100M	101M

Table 4.1: Comparison of the MTVY and Schnorr-ℓ signature schemes at the 128-bit security level.

The values in this column for Schnorr-ℓ assume that Shamir's multi-exponentiation trick is not used. Similarly, the values in the column for MTVY assume no methods are used for accelerating products of pairings. Notice that in MTVY the public key is always very large due to the need to include the Waters hash. Perhaps the most striking thing to note is that for small values of ℓ, Schnorr-ℓ can actually leak more bits between updates than MTVY. Furthermore, there is no restriction on the time periods where leakage can occur in Schnorr-ℓ whereas in MTVY only 7 bits can be leaked during key generation or update. However, since the key cannot be updated in Schnorr-ℓ, the overall number of bits that can be leaked in MTVY is much larger. In other words, for (almost) practical values of ℓ, Schnorr-ℓ is slightly more secure against one-shot attacks like cold boot, and MTVY is much more likely to be secure against attacks that require many measurements like DPA.

Other than this security tradeoff, signing in (unoptimized) MTVY is also 12 times slower than it is in (unoptimized) Schnorr-ℓ, which is already ℓ times slower than a practical scheme like Schnorr. Comparing the verification algorithms is a little more difficult, since pairing computations and point multiplications are not as nicely comparable. Nonetheless, it is safe to say that pairing computations are significantly slower than point multiplications and so verification is more than twice as slow in MTVY compared to (unoptimized) Schnorr-ℓ. Further, the enormous public key makes the scheme unusable in applications with constrained hardware such as smart cards. However, it may be possible to replace the Waters hash with a random oracle and so a more practical implementation of the scheme using a standard hash function is probably possible. Further, since MTVY does not rely on a random oracle, its security guarantees can be seen as more concrete, in some sense,

than those of Schnorr-ℓ. That being said, there are no known attacks against schemes that rely on the random oracle assumption using a cryptographic hash function in practice. Additionally, Chatterjee and Sarkar [14] show that storage requirements for the Waters hash can be reduced at the cost of needing more computations during signing and verification and a loss in tightness.

Another important drawback of MTVY is the inability to leak more than 7 bits during key update. While it is plausible that the key is securely generated, i.e. in an environment where it is unlikely that leakage will occur, the same cannot be said for key update. In particular, in the most likely use case key update and signing are done in the exact same environment and so are equally prone to leak. The effective need for a safe haven to perform key updates makes the key update procedure much more costly.

There is a way the scheme could be used without such a safe haven while still maintaing (provable) security. Suppose an implementer is only worried about point multiplications as sources of leakage. Further, she knows that at most 7 bits will leak every ℓ point multiplications (no matter which group they were performed in). Thus she can be sure that key updates satisfy the leakage bound. Since signing consists of 6ℓ point multiplications, she can also be sure that at most 42 bits leak per signature. When $\ell = 6$, this means she must update her key after every 4 signatures so that the scheme will remain provably secure. The question remains: how could an implementer ever obtain such an assurance of the leakiness of point multiplications? Is it reasonable to expect that each point multiplication leaks only around one bit of information? Of course, if no leakage can be tolerated (which we argue in two different ways above), then not even this limited scenario can be achieved.

We emphasize that an implementer should not take the leakage bounds during key generation and update lightly. Recall that the fact that no leakage occurred during either period was used several times in the proof. While it is certainly not clear that an attack exists against this scheme when these conditions are violated, it is also not clear that such attacks do not exist. At the very least there is a possibility that leakage during key generation and update significantly weakens the security of the scheme.

Of course, the ability to leak an overall unbounded number of bits is a significant advantage MTVY has over Schnorr-ℓ. This allows it the possibility of being secure against attacks that overall use a very large amount of information like DPA or lattice-type attacks. However, as noted in Section 2.2.2, the model may not be able to tolerate any style of power analysis attacks. Furthermore, it is not known how to apply DPA and lattice-type attacks to any schemes that employ key update, even ones which are not provably leakage resilient.

One thing to note about MTVY is that while it makes use of pairings, pairing computations do not need to be secured against side channel attacks. This is because the only time a pairing is computed using a secret value as input is during key generation when it is assumed that (almost) no leakage occurs. That is, to guarantee that no leakage occurs during key generation in practice, it would likely need to be implemented in a safe environment where side channel attacks could not be performed (and so there would be no need for secure implementations of any of the primitives used during key generation). If leakage was permitted when pairings were computed using secret values, however, research into side channel attacks against pairings would need to be done. We mention this, since

it is possible that other leakage resilient schemes make use of pairings in this way. Without a secure implementation of pairing computations, these schemes would be useless in protecting against side channel attacks.

Chapter 5

Conclusions

In this thesis we have examined leakage resilience, the response by the theoretical community to the problem of side channel attacks. We have argued that while some ideas and techniques used in leakage resilient public key schemes may prove useful in eventually constructing side channel resistant schemes, current leakage models and schemes are insufficient to guarantee security against side channel attacks. Most importantly, practitioners need to implement all the same countermeasures for the new schemes as they would for traditional schemes. Additionally, it is possible that new countermeasures need to be developed or that these new schemes are harder to protect against side channel attacks and so, in practice, they may actually fare worse against side channel attacks than traditional schemes.

We also demonstrated a disconnect between authors' claims about the utility of their schemes and the practicality of these schemes. By disguising theoretical results as practical, the claims help exacerbate the division between practitioners and theoreticians. We feel that this division is particularly damaging, since only by the combined work of theoreticians and practitioners can we hope to achieve security against side channel attacks. Further, we claim that the current direction of the field is one which will not be helpful in protecting against side channel attacks. That is, if anything, the field is becoming more theoretical, even though a substantial effort is already required to assess the benefits of proposed schemes in practice.

We have made several proposals in an effort to steer the field towards practical results. First, we suggest that authors give a practical analysis of the running time and leakage bounds of their schemes. For schemes that are clearly impractical, authors should make mention of this fact and hopefully propose some methods for overcoming these impracticalities. Further, authors should note the ways in which their schemes are vulnerable to existing side channel attacks. By doing so, they can identify components of their schemes for which new countermeasures need to be implemented. In the end, they should give some assessment of how difficult it is to implement countermeasures for their schemes (in comparison to traditional schemes). We also noted that schemes which tolerate only a small amount of leakage but have practical efficiency are much more useful than those that tolerate a large amount of leakage, but are impractical.

Ultimately, to fully analyze these schemes we need to be able to understand their security guarantees. To that end, we suggest research into several areas which is needed to determine which attacks are indeed covered by current leakage models. In particular, some notion of how many bits are leaked (or how much entropy is lost) when a given side channel measurement is performed is required for many of these models to be meaningful. Additionally, we noted that indistinguishability of ciphertexts may be hard to achieve for leakage resilient encryption schemes. Thus, we feel that current definitions which do not allow leakage during the encryption of the challenge (or allow only "entropic leakage") are inadequate. Therefore, we recommend that research is done into alternate security definitions for these schemes.

We also provided an in-depth look at two different leakage resilient signature schemes. We showed that Schnorr-ℓ, one of the most efficient schemes proposed so far, is significantly less efficient than standard Schnorr. Furthermore, for (almost) practical instantiations of the scheme we saw that only a (relatively) small number of bits could be leaked. In particular, we found it was hard to imagine a scenario in which the security guarantee was meaningful, especially given that no attacks are known on standard Schnorr when such little leakage occurs. That being said, we demonstrated that when around half the key bits are expected to leak Schnorr-2 is a better scheme to use than Schnorr with its key size doubled (note that *neither* scheme is provably secure in this scenario). We further explored a type of attack on Schnorr which extends to Schnorr-ℓ and showed that such attacks are not permitted under the (memory) bounded length leakage model.

Finally, we examined the MTVY signature scheme. In addition to the scheme being very slow (but still faster than other proposed schemes), we demonstrated that the tiny amount of leakage it allows during key updates makes its security guarantees of limited value in practice. Additionally, for reasonable instantiations of the scheme, we saw that it is actually less secure against cold boot attacks than Schnorr-ℓ. We also argued that that loss of tightness in Lemma 4.6.1 that is used by this and several other schemes to permit (some) leakage during key generation and update may make the lemma meaningless. We also showed that, ignoring the previous argument, the lemma might not actually be applicable to MTVY. Here the particular reliance of the security proof on a DDH problem may make the security guarantee of the scheme meaningless after applying the lemma. We suggested further research into this area to better understand the relation between Lemma 4.6.1 and decisional hardness assumptions. We used this example to once again argue that extreme care needs to be taken when using indistinguishability arguments in leakage resilient schemes.

The examination of the two schemes showed that it is unclear whether leakage resilient schemes offer any additional protection. That is, there are traditional schemes which are not known to be insecure even after leakage occurs. Even schemes in continual models are not necessarily more secure than other schemes with key updates. Further, the price of the ability to tolerate leakage is a rather large efficiency loss. Given that side channel attacks are much harder to mount than a number of other practical attacks, there is little evidence to suggest that the added protection of these schemes is worth the efficiency loss.

References

[1] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *6th Theory of Cryptography Conference*, TCC 2009, pages 474–495. Springer-Verlag.

[2] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *Advances in Cryptology*, Eurocrypto 2010. Springer-Verlag.

[3] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology*, Crypto 2009, pages 36–54. Springer-Verlag.

[4] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[5] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for Waters' IBE scheme. In *Advances in Cryptology*, Eurocrypt 2009, pages 407–424. Springer-Verlag.

[6] Dan J. Bernstein, Ian Goldberg, Nadia Heninger, Kevin S. McCurley, and Moti Yung. Leakage. *J. Craptology*, 8, 2011.

[7] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology*, Crypto 1997, pages 513–525. Springer-Verlag.

[8] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In *Advances in Cryptology*, Eurocrypt 1997, pages 37–51. Springer-Verlag.

[9] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology*, Crypto 2008, pages 108–125. Springer-Verlag.

[10] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *Advances in Cryptology*, Eurocrypt 2011, pages 89–108. Springer-Verlag.

[11] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *51st Annual Symposium on Foundations of Computer Science*, FOCS 2010, pages 501–510. IEEE Computer Society. Full version available at `http://eprint.iacr.org/2010/278`.

[12] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In *Advances in Cryptology*, Crypto 2000, pages 453–469. Springer-Verlag.

[13] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Advances in Cryptology*, Crypto 1999, pages 398–412. Springer-Verlag.

[14] Sanjit Chatterjee and Palash Sarkar. On (hierarchical) identity based encryption protocols with short public parameters (with an exposition of Waters' artificial abort technique). Cryptology ePrint Archive, Report 2006/279, 2006. `http://eprint.iacr.org/2006/279`.

[15] Sherman S. M. Chow, Yevgeniy Dodis, Yannis Rouselakis, and Brent Waters. Practical leakage-resilient identity-based encryption from simple assumptions. In *17th ACM Conference on Computer and Communications Security*, CCS 2010, pages 152–161. ACM.

[16] Yevgeniy Dodis. *Exposure-Resilient Cryptography*. PhD thesis, Massachusetts Institute of Technology, 2000.

[17] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *7th Theory of Cryptography Conference*, TCC 2010, pages 361–381. Springer-Verlag.

[18] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51th Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2010, pages 511–520. IEEE Computer Society.

[19] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *Advances in Cryptology*, Asiacrypt 2010, pages 613–631.

[20] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *41st Annual ACM Symposium on Theory of Computing*, STOC 2009, pages 621–630. ACM.

[21] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[22] Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In *Advances in Cryptology*, Eurocrypt 2000, pages 301–324. Springer-Verlag.

[23] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS 2008, pages 293–302. IEEE Computer Society.

[24] Thomas Eisenbarth, Timo Kasper, Amir Moradi, Christof Paar, Mahmoud Salmasizadeh, and Mohammad T. Manzuri Shalmani. On the power of power analysis in the real world: A complete break of the KeeLoq code hopping scheme. In *Advances in Cryptology*, Crypto 2008, pages 203–220. Springer-Verlag.

[25] Junfeng Fan, Xu Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede. State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures. In *International Symposium on Hardware-Oriented Security and Trust*, HOST 2010, pages 76–87, June.

[26] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *6th Theory of Cryptography Conference*, TCC 2010, pages 343–360. Springer-Verlag.

[27] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *Advances in Cryptology*, Eurocrypt 2010, pages 135–156. Springer-Verlag.

[28] Sebastian Faust, Leonid Reyzin, and Eran Tromer. Protecting circuits from computationally-bounded leakage. Cryptology ePrint Archive, Report 2009/379, 2009. `http://eprint.iacr.org/2009/379`.

[29] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology*, Crypto 1986, pages 186–194. Springer-Verlag.

[30] Jeffrey Friedman. TEMPEST: A signal problem. *NSA Cryptologic Spectrum*, Summer, 1972. Available at `http://www.nsa.gov/public_info/_files/cryptologic_spectrum/tempest.pdf`.

[31] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Third International Workshop on Cryptographic Hardware and Embedded Systems*, number Generators in CHES 2001, pages 251–261. Springer-Verlag.

[32] Catherine H. Gebotys and Robert J. Gebotys. Secure elliptic curve implementations: An analysis of resistance to power-attacks in a DSP processor. In *4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES 2002, pages 114–128. Springer-verlag.

[33] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology*, Eurocrypt 2008, pages 415–432. Springer-Verlag.

[34] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *17th USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

[35] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In *8th Theory of Cryptography Conference*, TCC 2011, pages 107–124. Springer-Verlag.

[36] Nadia Heninger and Hovav Shacham. Reconstructing rsa private keys from random key bits. In *Advances in Cryptology*, Crypto 2009, pages 1–17.

[37] N. A. Howgrave-Graham and N. P. Smart. Lattice attacks on digital signature schemes. *Designs, Codes and Cryptography*, 23:283–290, August 2001.

[38] J. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

[39] Marc Joye. Defences against side-channel analysis. In *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes Series*, pages 87–100. Cambridge University Press, 2005.

[40] Marc Joye and Sung-Ming Yen. The Montgomery powering ladder. In *4th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES 2002, pages 291–302. Springer-Verlag.

[41] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *Advances in Cryptology*, Crypto 2010, pages 41–58. Springer-Verlag.

[42] Jonathan Katz. Signature schemes with bounded leakage resilience. Cryptology ePrint Archive, Report 2009/220, 2009. `http://eprint.iacr.org/2009/220`.

[43] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology*, Asiacrypt 2009, pages 703–720. Springer-Verlag.

[44] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient ElGamal encryption. In *Asiacrypt 2010*, Advances in Cryptology, pages 595–612. Springer-Verlag.

[45] Neal Koblitz and Alfred Menezes. Another look at security definitions. Cryptology ePrint Archive, Report 2011/343, 2011. `http://eprint.iacr.org/2011/343`.

[46] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology*, Crypto 1996, pages 104–113. Springer-Verlag.

[47] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology*, Crypto 1999, pages 388–397. Springer-Verlag.

[48] Arjen Lenstra, Hendrik Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[49] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, Whit is right. Cryptology ePrint Archive, Report 2012/064, 2012. `http://eprint.iacr.org/2012/064`.

[50] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *43rd ACM Symposium on Theory of Computing*, STOC 2011, pages 725–734. Springer-Verlag.

[51] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *8th Theory of Cryptography Conference*, TCC 2011, pages 70–88. Springer-Verlag.

[52] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC 2011*, 8th Theory of Cryptography Conference, pages 89–106. Springer-Verlag. Full version available at `http://eprint.iacr.org/2010/522`.

[53] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., 2007.

[54] Marcel Medwed. Template-based SPA attacks on 32-bit ECDSA implementations. Master's thesis, Graz University of Technology, 2007.

[55] Alfred Menezes. *An Introduction to Pairing-Based Cryptography*, volume 477 of *Contemporary Mathematics*. AMS–RSME, 2009. Available at `http://cacr.uwaterloo.ca/~ajmeneze/publications/pairings.pdf`.

[56] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, 1st edition, 1996. `www.cacr.math.uwaterloo.ca/hac/`.

[57] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC 2004*, 1st Theory of Cryptography Conference, pages 278–296. Springer-Verlag.

[58] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[59] Morita Tech/AIST. Power analysis attacks on SASEBO. Technical Report, 2010. `http://www.morita-tech.co.jp/SASEBO/pdf/SASEBO_PA_Report_English.pdf`.

[60] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology*, Crypto 2009, pages 18–35. Springer-Verlag. Full version available at `http://eprint.iacr.org/2009/105`.

[61] Phong Q. Nguyen. Can we trust cryptographic software? Cryptographic flaws in GNU privacy guard v1.2.3. In *Advances in Cryptology*, Eurocrypt 2004, pages 555–570. Springer-Verlag.

[62] Phong Q. Nguyen and Igor Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *J. Cryptology*, 15:151–176, 2002.

[63] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology*, Crypto 1992, pages 31–53. Springer-Verlag.

[64] Elisabeth Oswald. Side-channel analysis. In *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes Series*, pages 69–86. Cambridge University Press, 2005.

[65] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards. In *International Conference on Research in Smart Cards*, E-smart 2001, pages 200–210. Springer-Verlag.

[66] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology*, Crypto 1989, pages 239–252. Springer-Verlag.

[67] François-Xavier Standaert. How leaky is an extractor? In *Progress in Cryptology*, Latincrypt 2010, pages 294–304. Springer-Verlag.

[68] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology*, Eurocrypt 2009, pages 443–461. Springer-Verlag.

[69] Franois-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In *Towards Hardware Intrinsic Security: Foundation and Practice*, pages 105–139. Springer, 2010.

[70] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology*, Eurocrypt 2005, pages 114–127. Springer-Verlag.

[71] Daniel Wichs. *Cryptographic Resilience to Continual Information Leakage*. PhD thesis, New York University, 2011.

[72] Peter Wright. *Spycatcher: The Candid Autobiography of a Senior Intelligence Officer*. Viking Penguin, 1987.