Published in *Cryptography and Commnunications*, **7**(1):71–90, 2015.

The Distributions of Individual Bits in the Output of Multiplicative Operations

Michael Tunstall · Marc Joye

Abstract A difference-of-means test applied to acquisitions of the instantaneous power consumption has been shown to be a suitable means of distinguishing a multiplication from a squaring operation *over the integers*. This has been attributed to the difference in expected Hamming weight of the output of these operations but few details are present in the literature. In this paper we define how this difference occurs and show that, somewhat surprisingly, a difference can, for some moduli, still be observed after a *modular reduction*. Moreover, we show that this difference leads to a practical attack under reasonable assumptions where a modulus is blinded. The presented attack goes beyond the cryptographic primitive and applies to concrete provably secure implementations, including RSA-PSS for signature generation or RSA-OAEP for encryption that uses side-channel countermeasures.

Keywords Side-Channel Analysis · Exponentiation Algorithms

1 Introduction

Side-channel analysis was first introduced to the cryptographic community by Kocher [17] who noted that a private key could be extracted from implementations of RSA [25] by observing the time required to compute a modular exponentiation. Further work by Kocher et al. [18] demonstrated that a

This work was conducted while the first author was employed by the Cryptography Group at the University of Bristol.

Michael Tunstall
Cryptography Research Inc.
425 Market Street, San Francisco, CA 94105, USA
E-mail: michael.tunstall@cryptography.com

Marc Joye
Technicolor
175 South San Antonio Road, Los Altos, CA 94022, USA
E-mail: marc.joye@technicolor.com

secret key could be extracted by a statistical treatment of numerous power consumption traces acquired during the computation of a block cipher. The initial attack used a difference-of-means test to determine key values, but alternatives such a Pearson's correlation coefficient [6], mutual information [13] or the Kolomogorov-Smirnov test [29] have since been proposed.

It is widely accepted that the instantaneous power consumption of a microprocessor typically corresponds to the Hamming weight or Hamming distance model [19]. That is, the instantaneous power consumption is typically proportional to the Hamming weight of the data being manipulated at a given point in time, or proportional to the Hamming weight of the data being manipulated at a given point in time XORed with some previous state. The Hamming weight model is typically assumed to correspond to bus lines or memory locations that are set to zero before being overwritten, whereas the Hamming distance model is typically assumed to correspond to transistors or RAM cells, for example, changing state. However, not everything is always visible since some operations, within an ALU for example, may not correspond to these models and hence cannot easily be exploited by an adversary.

The difference-of-means test has been shown to be sufficient to distinguish a multiplication from a squaring operation by comparing the instantaneous power consumption during two group operations. Akishita and Takagi [1] demonstrated that a mean power consumption trace of an FPGA implementation of a multiplication and squaring operation differs. A similar result has been presented by Amiel et al. [2], who observed that the expected Hamming weight of the result of a multiplication is different to the result of a squaring operation. This difference in expected Hamming weight was shown to be visible in acquisitions of the instantaneous power consumption taken while an ARM7 microprocessor was computing a multiplication or a squaring operation. Clavier et al. [8] proposed an improvement where the information from all the points where a difference could be observed were combined. Simulations suggested that this would allow an attacker to derive the same information from a single trace. A similar approach was taken Hanley et al. [14] who showed that this difference could be detected using a small number of traces by building suitable templates [7]. However, building these templates is somewhat problematic and a discussion of this topic is beyond the scope we wish to cover here.

The attacks described in the literature [1,2,8,14] do not require any knowledge of the input to an exponentiation, other than that it is random and uniformly distributed. These attacks are therefore applicable to cases where the input has been padded and/or blinded. However, no previous work provides much detail on why the difference in expected Hamming weight occurs.

In this paper we describe how the expected difference in Hamming weight is produced and show that a, much reduced, expected difference can still be present *after a modular reduction.* This difference is also shown to be present in the result of a Montgomery multiplication [20]. Furthermore, we show that the number of acquisitions required to conduct an attack after a modular reduction is reasonable.

We have structured our work as follows. We begin by defining an attack model that we will refer to in the rest of the paper. We continue in Section 3 by defining the difference in the expected Hamming weight for multiplications and squaring operations in $\mathbb{Z}$ and the expected number of traces required to observe this difference. We then show how this can be extended to $\mathbb{Z}_m$ in Section 4 and how this affects the required number of acquisitions. We describe some of the applications of such attacks in Section 5. We conclude in Section 6.

2 Attack Model

In this paper we consider an adversary who is able to measure the power consumption of a microprocessor while it is computing a modular exponentiation. The adversary wishes to determine the exponent and the ability to distinguish a multiplication from a squaring operation will allow individual bits of the exponent to be determined. We refer the interested reader to Kocher et al. [18] for a more detailed description. We assume that the instance of the modular exponentiation uses the binary exponentiation algorithm implemented such that an adversary cannot distinguish a modular multiplication and a modular squaring operation by inspecting a single trace.[1]

An adversary can take as many acquisitions as deemed necessary to conduct an attack. The input to the modular exponentiation is assumed to be random and unknown to an adversary. For example, using an implementation of RSA-PSS [5] or RSA-OAEP [4]. The adversary is able to take each trace acquired and divide it up into sub-traces, where each sub-trace corresponds to one group operation.

We consider two different side channels that could be available to an adversary:

1. The power consumption measured immediately after a multiplication is proportional to the Hamming weight of the result. For convenience, we assume that an adversary can directly determine the Hamming weight.
2. The power consumption measured immediately after a modular multiplication is proportional to the Hamming weight of the result. Given that the multiplication of large numbers will typically occur in dedicated hardware the Hamming weight of the multiplication may not be visible. Again, for convenience, we assume that an adversary can directly determine the Hamming weight.

The majority of modular multiplication algorithms used to compute the product of large numbers consist of interleaved multiplications and modular reductions. To achieve an efficient implementation the interleaved multiplications are typically with one computer word whose bit length is dictated by the architecture of a given microprocessor. We shall consider 32- and 64-bit microprocessors where an adversary can measure the power consumption at the end of each modular reduction step. Clavier et al. [9] have shown that one can

[1] Otherwise a private exponent could be determined by simple power analysis [18].

extract small power consumption traces corresponding to individual multiplications from a power consumption trace taken during the computation of a multi-precision multiplication, thus allowing an adversary to choose multiple points where the Hamming weight at each step in a multiplication, resp. modular multiplication, can be determined. We will consider these cases alongside the two models given above.

3 Defining the Distributions in $\mathbb{Z}$

We shall define an approximation to the distribution of the individual bits of the result of a multiplication and a squaring operation, corresponding to the analysis of the first type of side-channel leakage defined in Section 2.

3.1 Multiplication in $\mathbb{Z}$

In this section we shall define a function that returns the probability that the i-th most significant bit of the result of a multiplication is set to one for random multiplicands. Given r-bit multiplicands, we determine a precise result for the r least significant bits of the output by starting with a strong assumption and then show that the assumption is irrelevant. For the r most significant bits we use a more heuristic approach.

We consider $a, b, c \in \mathbb{Z}_{\geq 0}$ where $c = a\,b$. For convenience we shall assume that a and b have the same bit length r. Given a and b the simplest method to compute c is the grid method (a.k.a. schoolboy method) where we only consider digits in base 2 rather than base 10. Hence, for r-bit values of a and b we consider a repeated addition of shifted values of one of a and b. This is defined in Algorithm 1.

Algorithm 1: Grid Multiplication

```
  Input: $a, b \in \mathbb{Z}_{\geq 0}$ of bit length $r$
  Output: $c = a\,b \in \mathbb{Z}_{\geq 0}$
1 $z = 0$ ;
2 for $i = 0$ to $r - 1$ do
3     if $\lfloor b/2^i \rfloor \wedge 1 = 1$ then
4         $z = z + 2^i\,a$ ;
5     end
6 end
7 return $z$
```

This gives a series of additions of the form

```
X X X X X X X X X X X X X X X X X X X       +
  X X X X X X X X X X X X X X X X X X X     +
    X X X X X X X X X X X X X X X X X X X   +
      X X X X X X X X X X X X X X X X X X X +
. . .
```

where either $\nexists$X or X $\in \{0,1\}$. In terms of Algorithm 1, a line will exist if a certain bit of b is set to 1, otherwise the line forms a shifted bitwise representation of a. If we consider one column, the corresponding bit in the result c will be set to 1 if the sum of the column (and any carry bits) is odd.

We define Y_s as the sum of the bits of the s-th column, and W_s as the number of lines present in the addition described above, i.e. the Hamming weight of the s least significant bits of b. Then

$$\Pr[Y_s = y \mid W_s = w] = \binom{w}{y}\frac{1}{2^w} \quad \text{and} \quad \Pr(W_s = w) = \binom{s}{w}\frac{1}{2^s} \ .$$

There will be a possible combination for Y_s, for any $W_s \geq Y_s$. Hence:

$$\Pr[Y_s = y] = \sum_{i=y}^{s} \Pr[Y_s = y \mid W_s = i]\Pr[W_s = i] \ .$$

We define Z_s as the sum of the s-th column modulo 2 giving

$$\Pr[Z_s = 1] = \sum_{i=0}^{\lfloor s/2 \rfloor} \sum_{j=i}^{\lfloor s/2 \rfloor} \Pr[Y_s = 2\,j+1 \mid W_s = 2\,i+1]\Pr[W_s = 2\,i+1] \ .$$

The sum of the bits of a given column may exceed 1, so we have to consider the carry produced when performing the same operation on the next column. That is, if we define D_s as the carry produced from the s-th column then

$$\Pr[D_s = d] = \Pr[Y_s = 2\,d] + \Pr[Y_s = 2\,d+1] \,,$$

and

$$\Pr[Y_s = y] = \sum_{i=0}^{y} \Pr[D_{s-1} = y - i] \sum_{j=i}^{s} \Pr[Y_s = i \mid W_s = j]\Pr[W_s = j] \ .$$

Let $\kappa = \sum_{i=0}^{\lfloor s/2 \rfloor} \Pr[D_{s-1} = 2\,i]$, then

$$\begin{aligned}\Pr[Z_s = 1] = \kappa \sum_{i=0}^{\lfloor s/2 \rfloor} \sum_{j=2\,i+1}^{s} & \Pr[Y_s = 2\,i+1 \mid W_s = j]\Pr[W_s = j] \\ & + (1-\kappa) \sum_{i=0}^{\lfloor s/2 \rfloor} \sum_{j=2\,i}^{s} \Pr[Y_s = 2\,i \mid W_s = j]\Pr[W_s = j] \ .\end{aligned}$$

From which it can easily be shown that

$$\Pr[Z_s = 1] = \frac{1}{2} - \frac{1}{2^{s+1}}$$

(see Appendix A). Given that $\{\frac{1}{2^{s+1}}\}$ is a basic null sequence, $\Pr[Z_s = 1] \longrightarrow \frac{1}{2}$ as $s \longrightarrow \infty$ under the assumption that $s < r$.

Computing the remaining probabilities $\Pr[Z_s = 1]$ for $r < s \leq 2\,r$ cannot be reduced to such a simple expression since the impact of the carry cannot be eliminated in the same way. Results can be achieved by observing that the probability $\Pr[Z_s = 1]$ for the most significant bit of the result of a multiplication converges to a fixed value as r increases. Likewise, the second most significant bit converges to a fixed value as r increases, etc.

Lemma 1 *The probability that the n-th most significant bit of the result of a multiplication is set to one, given random multiplicands, will converge on a certain value as the bit lengths of the multiplicands tends to infinity.*

Proof We consider n-bit multiplicands and define the i-th most significant bit of the multiplicand used to make the row in the grid (as defined above) to be X_i, and R_i to be the i-th most significant bit of the output of a multiplication. We define c_i to be the sum of all the carry bits up to the $(i-1)$-th most significant bit of the output. Then the effect of a given column from the grid on the most significant bit of the output will be at most:

$$\frac{X_1 + c_1}{2}, \frac{\sum_{i=1}^{2} X_i + c_2}{2^2}, \frac{\sum_{i=1}^{3} X_i + c_3}{2^3}, \ldots, \frac{\sum_{i=1}^{j} X_i + c_j}{2^j}, \ldots$$

or, given $c_j \leq j$, equivalently,

$$\frac{2}{2}, \frac{4}{2^2}, \frac{6 + c_3}{2^3}, \ldots, \frac{2\,j}{2^j}, \ldots$$

Given that the dominant term will form the basic null sequence $\{\frac{1}{2^j}\}$, then $\frac{\sum_{i=1}^{j} X_i + c_j}{2^j} \longrightarrow 0$ as $j \longrightarrow \infty$. Hence, the probability that the most significant bit of the output is set to one converges in probability as a function of j as $j \longrightarrow \infty$. It is straightforward to modify this argument to most-significant r bits of the output. Hence, for random r-bit multiplicands the probability that individual bits in the most-significant r bits of the output are set to one converges. □

For example, the probabilities of the most significant bit being set to one, as the bit length increases, are given below to four significant figures for multiplicands of one to thirteen bits:

$$\{0, 0.0625, 0.09375, 0.125, 0.1377, 0.1460, 0.1491,$$
$$0.1513, 0.1524, 0.1529, 0.1532, 0.1533, 0.1533, 0.1533\}$$

We can see that for multiplicands of bit length greater than 11 the probability that the most significant bit is set to one is constant to four significant figures. While somewhat imprecise, this method is adequate for our requirements.

Hence, we define the set β as a given number of probabilities for the $|\beta|$ most significant bits of the result of a multiplication determined using the method described above. We define h_i as the i-th element of β, where i corresponds to the i-th most significant bit of the result of a multiplication. Then

$$f_m : {\mathbb{Z}_{>0}}^2 \longrightarrow \mathbb{Q}^+_{\leq 1} : f_m(s,r) \longmapsto \begin{cases} \frac{1}{2} - \frac{1}{2^{s+1}} & \text{if } s \leq r \\ \frac{1}{2} & \text{if } r < s \leq 2\,r - |\beta| \\ h_{2\,r-s+1} & \text{if } 2\,r - |\beta| < s \leq 2\,r \\ 0 & \text{if } s > 2\,r \end{cases}$$

defines a function f_m for computing $\Pr[Z_s = 1]$ for $1 \leq s \leq 2\,r$.

3.2 Squaring Operation in $\mathbb{Z}$

As above, we define Y_s as the sum of the bits of the s-th column, and W_s as the number of lines present in the addition described above, i.e. the Hamming weight of the s least significant bits of b. Again, we consider that the result of a series of additions of the form

```
X X X X X X X X X X X X X X X X X X X          +
  X X X X X X X X X X X X X X X X X X X        +
    X X X X X X X X X X X X X X X X X X X      +
      X X X X X X X X X X X X X X X X X X X    +
...
```

where either $\nexists \mathtt{X}$ or $\mathtt{X} \in \{0,1\}$. If we consider one column of the grid as n bits

$$\mathtt{X}_1, \mathtt{X}_2, \ldots, \mathtt{X}_n$$

where either $\nexists \mathtt{X}_i$ or $\mathtt{X}_i \in \{0,1\}$ for all $i \in \{1,\ldots,n\}$, then there will be relationships between elements of the vector. Generally, we can state

$$(\mathtt{X}_{i+1} = 1) \iff (\mathtt{X}_{n-i} = 1)$$

for all $i \in \{0,\ldots,\lfloor n/2 \rfloor - 1\}$. If n is odd then we can also state that

$$\exists \mathtt{X}_{\lceil n/2 \rceil} \iff (\mathtt{X}_{\lceil n/2 \rceil} = 1) \ .$$

This is a widely known result (see, for example, [24]). Then

$$\Pr[W_s = 2\,w] = \binom{\lfloor s/2 \rfloor}{w} \frac{1}{2^{\lfloor s/2 \rfloor}} \ ,$$

and if s is odd $\Pr[W_s = s] = \frac{1}{2^{\lceil s/2 \rceil}}$.

Without loss of generality we shall assume that s is even, since if s is even $\Pr[W_s = s] = \Pr[W_{s-1} = s-1]$. Hence,

$$\Pr[Y_s = 2\,y \mid W_s = 2\,w] = \binom{w}{y} \frac{1}{2^w} \ .$$

Again, we define D_s as the carry produced from the s-th column and let $\kappa = \sum_{i=0}^{\lfloor s/2 \rfloor} \Pr[D_{s-1} = 2\,i]$.

We note that the result of the sum of a given column will be even, and the result will impact the next column, then it can easily be shown that:

$$\begin{aligned}
\Pr[Z_s = 1] &= \kappa \sum_{j=0}^{\lfloor (s-1)/2 \rfloor} \sum_{k=2\,j+1}^{\lfloor (s-1)/2 \rfloor} \Pr[Y_s = 2\,j+1 \mid W_s = k] \Pr[W_s = k] \\
&\quad + (1-\kappa) \sum_{j=0}^{\lfloor (s-1)/2 \rfloor} \sum_{k=2\,j}^{\lfloor (s-1)/2 \rfloor} \Pr[Y_s = 2\,j \mid W_s = k] \Pr[W_s = k] \\
&= \frac{1}{2} - \frac{1}{2^{\lfloor (s-1)/2 \rfloor + 1}}
\end{aligned}$$

(see Appendix B). As previously, $\{\frac{1}{2^{\lfloor (s-1)/2 \rfloor+1}}\}$ is a basic null sequence and hence $\Pr[Z_s = 1] \longrightarrow \frac{1}{2}$ as $s \longrightarrow \infty$ under the assumption that $s < \frac{r}{2}$. The probability defined above is valid for $1 < s \leq r$. However we note that it is incorrect for $s = 1$ since $\Pr[Z_1 = 1] = \frac{1}{2}$.

Given Lemma 1 we state the following corollary:

Corollary 1 *The probability that the n-th most significant bit of the result of a squaring operation is set to one, given random multiplicands, will converge on a certain value as the bit lengths of the multiplicands tends to infinity.*

As previously, we define the set γ as a given number of probabilities for the $|\gamma|$ most significant bits of the result of a squaring operation determined using the method described above. We define g_i as the i-th element of γ, where i corresponds to the i-th most significant bit of the result of a squaring operation. Then

$$f_q : {\mathbb{Z}_{>0}}^2 \longrightarrow \mathbb{Q}^+_{\leq 1} : f_q(s,r) \longmapsto \begin{cases} \frac{1}{2} & \text{if } s = 1 \\ \frac{1}{2} - \frac{1}{2^{\lfloor (s-1)/2 \rfloor+1}} & \text{if } 1 < s \leq r \\ \frac{1}{2} & \text{if } r < s \leq 2\,r - |\gamma| \\ g_{2\,r-s+1} & \text{if } 2\,r - |\gamma| < s \leq 2\,r \\ 0 & \text{if } s > 2\,r \end{cases}$$

defines a function f_q for computing $\Pr[Z_s = 1]$ for $1 \leq s \leq 2\,r$.

3.3 Using the Difference to Distinguish a Multiplication from a Squaring Operation

As defined in Section 2, we assume that the power consumption of a device is proportional to the Hamming weight of the values being manipulated at a given point in time. The distributions defined for a multiplication and a squaring operation can be used to describe an attack by computing a difference of means as described by Kocher et al. [18]. In the following we determine the minimum sample size required to conduct such an attack using a straightforward variation of the method described by Mangard et al. [19].

Given a series of probabilities that the s-th bit is equal to 1, i.e. $\Pr[Z_s = 1]$ using the notation given above. The probabilities of individual bits being set to one, for a given operation on r-bit operands, is an instance of the Poisson binomial distribution. We define h as the Hamming weight of the output, then the expected Hamming weight $\mathbb{E}(h)$ and the variance $\mathrm{var}(h)$ are defined as

$$\mathbb{E}(h) = \sum_{i=1}^{2\,r} \Pr(Z_i = 1) \quad \text{and} \quad \mathrm{var}(h) = \sum_{i=1}^{2\,r} (1 - \Pr[Z_i = 1]) \Pr[Z_i = 1] \ .$$

The Poisson binomial distribution is non-normal. However, the majority of probabilities will be equal to $\frac{1}{2}$ (i.e. the distribution will be close to a standard binomial) and, given the effect of the Central Limit Theorem, it is reasonable to assume that the Poisson binomial distribution is normal in this instance.

We define μ_q and σ_q^2 to be the expected Hamming weight and variance of the output of a squaring operation, and μ_m and σ_m^2 to be the expected Hamming weight and variance of the output of a multiplication. Under the assumption that $\mu_q = \mu_m$ then the difference

$$\frac{\bar{\mu}_q - \bar{\mu}_m}{\sqrt{\frac{\bar{\sigma}_q^2}{n} + \frac{\bar{\sigma}_m^2}{n}}} \sim N(0,1)$$

where $\bar{\mu}_q$ and $\bar{\mu}_m$ are the sample means, from n samples, for the Hamming weight of a multiplication and squaring operations respectively, with sample variances $\bar{\sigma}_q^2$ and $\bar{\sigma}_m^2$. We also use the asymptotic result that Student's t-distribution is equivalent to the standard normal distribution.

We wish to show that the null hypothesis $\mu_q = \mu_m$ is incorrect to demonstrate that two sample means are not the same (and hence the difference implies a multiplication has been compared with a squaring operation). Let $-Q_{\alpha/2}$ and $Q_{\alpha/2}$ be the value on the cumulative density function of the standard normal distribution, where the cumulative probability is equal to $\alpha/2$ and $1 - \alpha/2$ respectively. Then for a chosen value of α one would expect that if $\mu_q = \mu_m$ then

$$Q_{1-\alpha/2} \geq \left| \frac{\mu_q - \mu_m}{\sqrt{\frac{\sigma_q^2}{n} + \frac{\sigma_m^2}{n}}} \right| \ , \text{ and hence } \quad n = \frac{\sigma_q^2 + \sigma_m^2}{(\mu_q - \mu_m)^2} {Q_{1-\alpha/2}}^2$$

will be the minimum sample size to provide evidence against the null hypothesis that $\mu_q \neq \mu_m$ if it is false.

If we set $\alpha = 0.001$ then the probability of incorrectly determining that a given difference is consistent with the null hypothesis is $1/1000$, and similar for incorrectly determining that a given difference is not consistent with the null hypothesis. If we apply this to the distributions defined for the multiplication and squaring operation the number of samples required is shown in Table 1.

Table 1 The required number of traces to distinguish a multiplication and a squaring operation in $\mathbb{Z}$ where $\alpha = 0.001$.

Bit Length	Operation	$\mathbb{E}(h)$	var(h)	Acquisitions Required
512	Squaring	509.81	255.39	5.53×10^3
	Multiplication	510.81	255.73	
1024	Squaring	1021.81	511.39	1.11×10^4
	Multiplication	1022.81	511.73	
2048	Squaring	2045.81	1023.39	2.22×10^4
	Multiplication	2046.81	1023.73	

The model given in Section 2 assumes that an adversary is able to determine the Hamming weight at a given point in time from a power consumption trace. In practice this information will be noisy where the noise level depends on the microprocessor under attack. The effects of noise on an attack based on a difference-of-means is covered in detail by Mangard et al. [19] and will not be discussed here.

4 Defining the Distribution in $\mathbb{Z}_m$

In this section we define an approximation to the distribution of the individual bits of the result of a modular multiplication and a modular squaring operation, corresponding to the analysis of the second type of side channel leakage defined in Section 2.

4.1 Modular Multiplication

In the previous section we discuss the distribution for operations in $\mathbb{Z}$. We consider $a, b, c \in \mathbb{Z}_m$ where $c = a\,b$ and $m \in \mathbb{Z}_{>0}$. We consider that the result is reduced modulo m of bit length r. Then the result

$$c \equiv a\,b \pmod{m}$$

can be computed using

$$c - k\,m = a\,b \bmod m$$

for some $k \in \mathbb{Z}_m$. For convenience, we will assume that m is a random variable of a fixed bit length that is independent to a and b. We note that k is not independent to a and b but the bitwise representation of $k\,m$ will be random assuming that the longest run of zeros or ones in m is less than the longest run of zeros or ones in k [26].

We have defined the distribution of the individual bits of c in the previous section, and $k\,m$ will have the same distribution as the result of a multiplication. The effect of this subtraction can be readily computed. We define $Z_{t,i}$ as the probability the i-th bit of the targeted operation (either a multiplication or squaring operation) is equal to 1 and $Z_{m,i}$ as the probability the i-th bit of the result of a multiplication is equal to 1. Furthermore, we define b_i to the be value of the borrow at the i-th bit and $Z_{x,i}$ the probability the i-th bit of the result is equal to 1. Then

$$\begin{aligned}\Pr[Z_{x,i} = 1] = \Pr[b_i = 0]\,((\Pr[Z_{t,i} = 0]\Pr[Z_{m,i} = 1])\,(\Pr[Z_{t,i} = 1]\Pr[Z_{m,i} = 0])) \\ + \Pr[b_i = 1]\,((\Pr[Z_{t,i} = 0]\Pr[Z_{m,i} = 0]) + (\Pr[Z_{t,i} = 1]\Pr[Z_{m,i} = 1]))\end{aligned}$$

and

$$\begin{aligned}\Pr[b_{i+1} = 1] = \Pr[b_i = 0]\,(\Pr[Z_{t,i} = 1]\Pr[Z_{m,i} = 1]) \\ + \Pr[b_i = 1]\,(1 - (\Pr[Z_{t,i} = 0]\Pr[Z_{m,i} = 0]))\ ,\end{aligned}$$

which can be used to determine the number of traces required to distinguish a multiplication from a squaring operation. A described in Section 2, the number of traces required to conduct an attack can be reduced where single-precision multiplications and squaring operations can be compared in a multi-precision multiplication.

If we assume that an adversary is analysing an implementation with interleaved multiplications and modular reductions, as described in Section 2, the expected Hamming weight can be computed as described above where one operand has a limited bit length. Since the probability of a given bit being set to one rapidly converges to $\frac{1}{2}$, the resulting expectation and variance are the same at each stage of the operation. This remains true at each stage of the multiplication since the first operation will be a multiplication or a squaring operation where the first x bits are potentially equal. The second multiplication will correspond to a multiplication or a squaring operation where the first $2\,x$ bits are potentially equal, and this will continue until a full multi-precision multiplication has been computed. A number of samples can therefore be taken from each acquisition, where the word-size of a given microprocessor will dictate how many points are of interest.

In Table 2 we summarise the required number of traces required to distinguish a multiplication from a squaring operation. We give results based on the word size of the processor considered, which are: large (i.e. only one observation will be possible), 32-bit and 64-bit.

Table 2 The required number of traces to distinguish a multiplication from a squaring operation in $\mathbb{Z}_n$ where $\alpha = 0.001$.

Bit Length	Operation	$\mathbb{E}(h)$	var(h)	Acquisitions Required		
				Large	32-bit	64-bit
512	Squaring Multiplication	255.91 255.85	127.99 127.98	9.20×10^5	5.75×10^4	1.15×10^5
1024	Squaring Multiplication	511.91 511.85	255.99 255.98	1.84×10^6	5.75×10^4	1.15×10^5
2048	Squaring Multiplication	1023.91 1023.85	511.99 511.98	3.68×10^6	5.75×10^4	1.15×10^5

The above analysis assumes that the modulus is different for each computation of a multiplication or a squaring operation, i.e. $\Pr[Z_{m,i} = 1] = \frac{1}{2}$ for all $1 \leq i \leq \log_2 m$. The results based on this analysis are therefore the expected difference for a fixed modulus, or the expected observable difference assuming that the modulus is randomised by adding some random multiple of the modulus before computing an exponentiation. For a fixed modulus the difference may exist but the expected difference will be different in each instance, and may not be statistically significant. Moreover, in some cases a modular squaring operation will produce results with a higher expected Hamming weight than a modular multiplication. This means that it is essential that one uses a two-tailed test, as described in Section 3.3, when evaluating the significance of an observed difference. The results given in Table 2, and the results described below, are therefore only indicative and will be different for an arbitrary fixed modulus.

Simulating an attack, one cannot produce a statistically significant difference with a randomly generated RSA modulus (such as used for RSA-PSS [5] and RSA-OAEP [4]) or for a prime modulus defined by NIST for use in ECDSA [22]. However, a difference can be readily observed where a modulus is blinded [11], i.e. where one typically multiplies the modulus by a random value to provide a redundant representation for intermediate values. The reader is referred to Smart et al. [26] for a thorough discussion of randomised representations. We also note that if m is a power of 2, then the resulting difference will correspond to the difference before modular reduction, allowing an attack to be conducted with a similar number of acquisitions to that required to distinguish a multiplication and a squaring operation in $\mathbb{Z}$.

The number of traces required to conduct the attack where $\alpha = 0.001$ is quite high, and will be impractical in some circumstances. One could, therefore, increase the chosen value of α so that the error rate increases. An adversary would then be obliged to solve a discrete logarithm problem. We summarise the results for a variety of values of α in Table 3 using Stinson's algorithm for exponents of low Hamming weights [27], where for an exponent of bit length η with an unknown t-bit error the exponent can be derived with time complexity $\mathcal{O}\left(\eta \sum_{n=0}^{\lceil t/2 \rceil} \binom{\eta/2}{n}\right)$, as described in Appendix C.

Remark 1 At first sight our results may seem to be wrong. One might (incorrectly) conclude that seeing a difference in the distributions of squaring operations and multiplications modulo m contradicts the quadratic residuosity (QR) assumption. This would be because of the following: Let N be an RSA modulus; i.e., $N = p\,q$ is the product of two large primes. Loosely speaking, the QR assumption conjectures that squares in $\mathbb{Z}_N^*$ cannot be distinguished from pseudo-squares—that is, non-squares in $\mathbb{Z}_N^*$ *with Jacobi symbol 1*. This section states that the expected Hamming weight of squared elements is different to the product of two distinct elements. For two elements $a, b \in \mathbb{Z}_N^*$, $a \neq b$, we note that if $w := a/b \bmod N$ is a square then the multiplication $a\,b \bmod N$ can, equivalently, be obtained as $(b\,w)b \bmod N$ or as $x^2 \bmod N$ where $x = b\,z \bmod N$ with z a square-root of w. Hence the seemingly possible

Table 3 The required number of traces to distinguish a multiplication from a squaring operation in $\mathbb{Z}_m$ for different values of α.

Bit Length	α	Acquisitions Required			Expected Error (bits)	Time Complexity
		Large	32-bit	64-bit		
512	0.001	9.20×10^5	5.75×10^4	1.15×10^5	1	2^{17}
	0.005	6.69×10^5	4.18×10^4	8.37×10^4	3	2^{24}
	0.01	5.64×10^5	3.52×10^4	7.05×10^4	6	2^{30}
1024	0.001	1.84×10^6	5.75×10^4	1.15×10^5	2	2^{19}
	0.005	1.34×10^6	4.18×10^4	8.37×10^4	6	2^{34}
	0.01	1.13×10^6	3.52×10^4	7.05×10^4	11	2^{54}
2048	0.001	3.68×10^6	5.75×10^4	1.15×10^5	3	2^{30}
	0.005	2.68×10^6	4.18×10^4	8.37×10^4	11	2^{61}
	0.01	2.26×10^6	3.52×10^4	7.05×10^4	21	2^{94}

contradiction. However, our results were obtained for the *whole* group $\mathbb{Z}_N^*$, and not for the set of elements with Jacobi symbol 1. Our results do not hold in this case. We also note that the QR assumption does not hold in the whole group $\mathbb{Z}_N^*$: using the Jacobi symbol as a distinguisher, the advantage is of 1/4.

4.2 Montgomery Multiplication

Montgomery multiplication functions differently to most modular multiplication algorithms. Again, we consider $a, b, c \in \mathbb{Z}_m$ where $c = a\,b$ and $m \in \mathbb{Z}_{>0}$ (m odd). The Montgomery multiplication will result in

$$c \equiv a\,b\,2^{-\delta} \pmod{m},$$

where we define $\delta = \lceil \log_2 m \rceil$ to be the bit length of m. Equivalently, one can compute $c = a\,b$ and one then adds a multiple of m, e.g. $k\,m$ for some $k \in \mathbb{Z}_{>0}$, such that the least significant δ bits of the result are set to zero. That is:

$$\frac{a\,b + k\,m}{2^\delta} = a\,b\,2^{-\delta} \bmod m$$

As for the strategy described in Section 4.1, if we know the distribution of the most significant bits of a multiplication and squaring operations the resulting distribution can be determined.

The result of the Montgomery multiplication algorithm, and the interleaved reduction steps, will be between 0 and $2\,m$ which leads to a slightly higher value for $\mathbb{E}(h)$. The expected number of traces that one requires to conduct an attack is given in Table 4, and the expected number of required traces for different values of α is given in Table 5. It is interesting to note that the required number of traces is approximately half that required for a standard modular multiplication.

The values given in Table 4 and Table 5 are only indicative and will be different for an arbitrary fixed modulus. As with the results described in Section 4.1, the results based on this analysis are therefore the expected difference

Table 4 The required number of traces to distinguish a multiplication from a squaring operation in $\mathbb{Z}_n$ using Montgomery multiplication and $\alpha = 0.001$.

Bit Length	Operation	$\mathbb{E}(h)$	var(h)	Acquisitions Required Full	32-bit	64-bit
512	Squaring Multiplication	255.97 255.89	128.11 128.09	4.13×10^5	2.58×10^4	5.16×10^4
1024	Squaring Multiplication	511.97 511.89	256.11 256.09	8.26×10^5	2.58×10^4	5.16×10^4
2048	Squaring Multiplication	1023.97 1023.89	512.11 512.09	1.65×10^6	2.58×10^4	5.16×10^4

Table 5 The required number of traces to distinguish a multiplication and a squaring operation in $\mathbb{Z}_n$ using Montgomery multiplication for different values of α.

Bit Length	α	Acquisitions Required Full	32-bit	64-bit	Expected Error (bits)	Time Complexity
512	0.001	4.13×10^5	2.58×10^4	5.16×10^4	1	2^{17}
	0.005	3.01×10^5	1.88×10^4	3.76×10^4	3	2^{24}
	0.01	2.53×10^5	1.58×10^4	3.16×10^4	6	2^{30}
1024	0.001	8.26×10^5	2.58×10^4	5.16×10^4	2	2^{19}
	0.005	0.60×10^6	1.88×10^4	3.76×10^4	6	2^{34}
	0.01	0.51×10^6	1.58×10^4	3.16×10^4	11	2^{54}
2048	0.001	1.65×10^6	2.58×10^4	5.16×10^4	3	2^{30}
	0.005	1.20×10^6	1.88×10^4	3.76×10^4	11	2^{61}
	0.01	1.01×10^6	1.58×10^4	3.16×10^4	21	2^{94}

for a fixed modulus, or the expected observable difference assuming that the modulus is randomised by adding some random multiple of the modulus before computing an exponentiation.

5 Applications of the Attack

In the attack described above we assume that an adversary is seeking to determine a private exponent used in the binary exponentiation algorithm. Furthermore, we assume that the input to the exponentiation algorithm is random and unknown to an adversary. Indeed, to satisfy the requirement that the output of group operations are distributed as defined above, the input has to be at least somewhat random. The attack is therefore applicable to implementations of RSA-PSS [5] and RSA-OAEP [4] where an adversary cannot derive the input to an exponentiation algorithm. However, our analysis suggests that, unless simulated results suggest otherwise, an attack requires that an implementation uses a blinded modulus [11].

We recall that the attacks described in the literature [1,2,8,14] function by distinguishing a multiplication from a squaring operation *over the integers*. Clavier et al. [10] describe an algorithm that only uses squaring operations to compute a multiplication, based on the observation that

$$x\,y = \frac{(x+y)^2 - x^2 - y^2}{2}$$

and hence

$$x\,y = \left(\frac{x+y}{2}\right)^2 - \left(\frac{x-y}{2}\right)^2 \ .$$

Clavier et al. show that using this countermeasure is a practical solution given that the time required to compute a squaring operation can be $\frac{4}{5}$ that of a multiplication. We can note that the result of any operation equivalent to $x\,y$ will still have the distribution described in Section 4. The described analysis would therefore apply, but an adversary would only have one point that could be analysed, rather than a point after each multiplication of interleaved multiplication and modular reduction steps.

Countermeasures to the attacks described in the analysis given above can be readily found in the literature. The simplest approach would be to use a regular exponentiation algorithm so that the required information is not available to an adversary. Some examples of such algorithms for constrained devices include the Montgomery powering ladder [16,21] and Joye's highly regular exponentiation algorithms [15]. One could also use a redundant representation when computing a squaring operation by using two different representations for the same value [11,12,26]. The probability distributions for multiplication and squaring operations would then be the same.

Another approach would be to randomise the structure of an algorithm such that the information is spread over numerous points. This could be done by changing the exponentiation algorithm as, for example, proposed by Oswald and Aigner [23]. Alternatively, one could compute a multi-precision multiplication in a random order as described by Bauer et al. [3]. Typically, one would say that these countermeasures only require that an attacker take more acquisitions to compensate (see Mangard et al. [19] for more details). However, given the large number of acquisitions that are required, these countermeasures may be sufficient.

6 Conclusion

A difference-of-means test applied to acquisitions of the instantaneous power consumption has been shown to be a suitable means of distinguishing a multiplication from a squaring operation [1,2,8,14]. These attacks are based on the difference in the expected Hamming weight of the result of a multiplication and a squaring operation. While these attacks rely on the exponent remaining unchanged, and will not apply to all moduli, the input to the exponentiation can be unknown, although it is necessary that the input is different in each execution of the exponentiation. These attacks are therefore applicable to cases where the input has been padded and/or blinded.

However, the literature contains no details on why this difference occurs which we address in this paper. We describe how the expected difference in Hamming weight occurs and show that a, much reduced, expected difference is still present after a modular reduction. This difference is also shown to be present in the result of a Montgomery multiplication. We define the expected minimum number of acquisitions required to conduct an attack before and after a modular reduction, which are practical in some cases. If an adversary is able to extract individual multiplications, as described by Clavier et al. [9], the number of acquisitions is low enough to be practical in many of the cases considered. In each case we define the time complexity of analysing a hypothesis to determine the exponent used showing the majority of the cases considered are practical. However, we do not consider the impact of acquisition noise on the number of acquisitions required to conduct an attack. This will be an important consideration in any implementation of these attacks since it will increase the number of acquisitions that are required. A discussion of this topic is beyond the scope of this paper and the interested reader is referred to Mangard et al. [19].

For a fixed and known modulus one can estimate the difference in the expected Hamming weight of the result of a multiplication and a squaring operation using the Monte Carlo method. One can then determine whether a statistically significant difference could be observed given a reasonable number of traces. Simulating an attack, it appears that one cannot produce a statistically significant difference with a randomly generated RSA modulus (such as used for RSA-PSS [5] and RSA-OAEP [4]) or for a prime modulus defined by NIST for use in ECDSA [22].

Acknowledgements The authors would like to thank the anonymous referees for their detailed and perceptive comments. The work described in this paper has also been supported in part the European Commission through the ICT Programme under Contract ICT-2007-216676 ECRYPT II and the EPSRC via grant EP/I005226/1.

References

1. Akishita, T., Takagi, T.: Power analysis to ECC using differential power between multiplication and squaring. In: J. Domingo-Ferrer, J. Posegga, D. Schreckling (eds.) CARDIS 2006, *LNCS*, vol. 3928, pp. 151–164. Springer (2006)
2. Amiel, F., Feix, B., Tunstall, M., Whelan, C., Marnane, W.P.: Distinguishing multiplications from squaring operations. In: H. Youm, M. Yung (eds.) SAC 2008, *LNCS*, vol. 5932, pp. 148–162. Springer (2009)
3. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal and vertical side-channel analysis against secure RSA implementations. In: E. Dawson (ed.) CT-RSA 2013, *LNCS*, vol. 7779, pp. 1–17. Springer (2013)
4. Bellare, M., Rogaway, P.: Optimal asymmetric encryption — how to encrypt with RSA. In: A.D. Santis (ed.) EUROCRYPT '94, *LNCS*, vol. 950, pp. 92–111. Springer (1994)
5. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: U. Maurer (ed.) EUROCRYPT '96, *LNCS*, vol. 1070, pp. 399–416. Springer (1996)
6. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: M. Joye, J.J. Quisquater (eds.) CHES 2004, *LNCS*, vol. 3156, pp. 16–29. Springer (2004)

7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: B.S.K. Jr., C.K. Koç, C. Paar (eds.) CHES 2002, *LNCS*, vol. 2523, pp. 13–28. Springer (2002)
8. Clavier, C., Feix, B., Gagnerot, G., Giraud, C., Roussellet, M., Verneuil, V.: ROSETTA for single trace analysis. In: S. Galbratih, M. Nandi (eds.) INDOCRYPT 2012, *LNCS*, vol. 7668, pp. 140–155. Springer (2012)
9. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: M. Soriano, S. Qing, J. López (eds.) ICICS 2010, *LNCS*, vol. 6476, pp. 46–61. Springer (2010)
10. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Square always exponentiation. In: D.J. Bernstein, S. Chatterjee (eds.) INDOCRYPT 2011, *LNCS*, vol. 7107, pp. 40–57. Springer (2011)
11. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: C.K. Koç, C. Paar (eds.) CHES 1999, *LNCS*, vol. 1717, pp. 292–302. Springer (1999)
12. Dupaquis, V., Venelli, A.: Redundant modular reduction algorithms. In: E. Prouff (ed.) CARDIS 2011, *LNCS*, vol. 7079, pp. 102–114. Springer (2011)
13. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: E. Oswald, P. Rohatgi (eds.) CHES 2008, *LNCS*, vol. 5154, pp. 426–442. Springer (2008)
14. Hanley, N., Tunstall, M., Marnane, W.P.: Using templates to distinguish multiplications from squaring operations. International Journal of Information Security **10**(4), 255–266 (2011)
15. Joye, M.: Highly regular right-to-left algorithms for scalar multiplication. In: P. Paillier, I. Verbauwhede (eds.) CHES 2007, *LNCS*, vol. 4727, pp. 135–147. Springer (2007)
16. Joye, M., Yen, S.M.: The Montgomery powering ladder. In: B.S.K. Jr., Ç. K. Koç, C. Paar (eds.) CHES 2002, *LNCS*, vol. 2523, pp. 291–302. Springer (2003)
17. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: N. Koblitz (ed.) CRYPTO '96, *LNCS*, vol. 1109, pp. 104–113. Springer (1996)
18. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: M.J. Wiener (ed.) CRYPTO '99, *LNCS*, vol. 1666, pp. 388–397. Springer (1999)
19. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks — Revealing the Secrets of Smart Cards. Springer (2007)
20. Montgomery, P.: Modular multiplication without trial division. Mathematics of Computation **44**, 519–521 (1985)
21. Montgomery, P.: Speeding the Pollard and elliptic curve methods of factorization. Mathematics of Computation **48**(177), 243–264 (1987)
22. National Institute of Standards and Technology (NIST): Recommended elliptic curves for federal government use. In the appendix of FIPS 186-3, available from `http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf` (2009)
23. Oswald, E., Aigner, M.: Randomized addition-subtraction chains as a countermeasure against power attacks. In: Ç. K. Koç, D. Naccache, C. Paar (eds.) CHES 2001, *LNCS*, vol. 2162, pp. 39–50. Springer (2001)
24. Parhami, B.: Computer Arithmetic. Oxford University Press (2000)
25. Rivest, R., Shamir, A., Adleman, L.M.: Method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
26. Smart, N., Oswald, E., Page, D.: Randomised representations. IET Proceedings on Information Security **2**(2), 19–27 (2008)
27. Stinson, D.: Some baby-step giant-step algorithms for the low Hamming weight discrete logarithm problem. Mathematics of Computation **71**(237), 379–391 (2002)
28. Teske, E.: New algorithms for finite abelian groups. Ph.D. thesis, Technische Universität Darmstadt (1998)
29. Whitnall, C., Oswald, E., Mather, L.: An exploration of the Kolmogorov-Smirnov test as a competitor to mutual information analysis. In: E. Prouff (ed.) CARDIS 2011, *LNCS*, vol. 7079, pp. 234–251. Springer (2011)

A $\Pr[Z_s = 1]$ for Multiplication in $\mathbb{Z}$

Following the notation we define in Section 3.1. We define Y_s as the sum of the bits of the s-th column, and W_s as the number of lines present in the addition described above, i.e. the Hamming weight of the s least significant bits of the result of a multiplication.

$$\Pr[Y_s = y \mid W_s = w] = \binom{w}{y}\frac{1}{2^w} \quad \text{and} \quad \Pr(W_s = w) = \binom{s}{w}\frac{1}{2^s} \ .$$

We also define D_{s-1} as the carry produced from the $(s-1)$-th column then

$$\Pr[D_{s-1} = d] = \Pr[Y_{s-1} = 2\,d] + \Pr[Y_{s-1} = 2\,d + 1]\,,$$

and

$$\Pr[Y_s = y] = \sum_{i=0}^{y} \Pr[D_{s-1} = y - i] \sum_{j=i}^{s} \Pr[Y_s = i \mid W_s = j] \Pr[W_s = j] \ .$$

Let $\kappa = \sum_{i=0}^{\lfloor s/2 \rfloor} \Pr[D_{s-1} = 2\,i]$, then

$$\begin{aligned}
\Pr[Z_s = 1] &= \kappa \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j+1}^{s} \Pr[Y_s = 2\,j + 1 \mid W_s = k] \Pr[W_s = k] \\
&\qquad + (1-\kappa) \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j}^{s} \Pr[Y_s = 2\,j \mid W_s = k] \Pr[W_s = k] \\
&= \kappa \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j+1}^{s} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{s}{k}\frac{1}{2^s} + (1-\kappa) \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j}^{s} \binom{k}{2\,j}\frac{1}{2^k}\binom{s}{k}\frac{1}{2^s} \\
&= (\kappa + (1-\kappa)) \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j+1}^{s} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{s}{k}\frac{1}{2^s} \quad \text{(by Lemma 2)} \\
&= \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j+1}^{s} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{s}{k}\frac{1}{2^s} \\
&= \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j+1}^{s} \Pr[Y_s = 2\,j + 1 \mid W_s = k] \Pr[W_s = k] \ .
\end{aligned}$$

Hence, one can compute $\Pr[Z_s = 1]$ without needing to compute the carry at each step.

Lemma 2 *Given the binomial numbers $\binom{n}{r}$ for $r \in \{0, \dots, n\}$ for some $n \in \mathbb{Z}_{>0}$, then*

$$\sum_{r \ odd} \binom{n}{r} = \sum_{r \ even} \binom{n}{r} \ .$$

Proof This follows from the binomial formula by noting that

$$0 = (1-1)^n = \sum_{0 \le r \le n} \binom{n}{r} 1^{n-r}(-1)^r = \sum_{\substack{0 \le r \le n \\ r \text{ even}}} \binom{n}{r} - \sum_{\substack{0 \le r \le n \\ r \text{ odd}}} \binom{n}{r} \, .$$

□

Furthermore, given that

$$\Pr[Z_s = 1] = \sum_{j=0}^{\lfloor s/2 \rfloor} \sum_{k=2\,j+1}^{s} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{s}{k}\frac{1}{2^s} \ ,$$

for any k,

$$\sum_{\substack{1\le j\le k,\\ j \text{ odd}}} \binom{k}{j}\frac{1}{2^k} = \frac{1}{2} \quad \text{(by Lemma 1).}$$

Hence,

$$\Pr[Z_s = 1] = \sum_{k=1}^{s} \frac{1}{2}\binom{s}{k}\frac{1}{2^s} = \frac{1}{2} - \frac{1}{2}\binom{s}{0}\frac{1}{2^s} = \frac{1}{2} - \frac{1}{2^{s+1}} \,.$$

B $\Pr[Z_s = 1]$ for Squaring Operation in $\mathbb{Z}$

Without loss of generality we shall assume that s is even, since if s is even $\Pr[W_s = s] = \Pr[W_{s-1} = s-1]$. Hence,

$$\Pr[Y_s = 2\,y \mid W_s = 2\,w] = \binom{w}{y}\frac{1}{2^w}$$

Again, we define D_{s-2} as the carry produced from the $(s-2)$-th column and let $\kappa = \sum_{i=0}^{\lfloor (s-2)/2\rfloor} \Pr[D_{s-2} = 2\,i]$. We note that the result of the sum of a given column will be even, and the result will impact the next column, then

$$\begin{aligned}
\Pr[Z_s = 1] &= \kappa \sum_{j=0}^{\lfloor (s-1)/2\rfloor} \sum_{k=2\,j+1}^{\lfloor (s-1)/2\rfloor} \Pr[Y_s = 2\,j+1 \mid W_s = k]\Pr[W_s = k] \\
&\quad + (1-\kappa) \sum_{j=0}^{\lfloor (s-1)/2\rfloor} \sum_{k=2\,j}^{\lfloor (s-1)/2\rfloor} \Pr[Y_s = 2\,j \mid W_s = k]\Pr[W_s = k] \\
&= \kappa \sum_{j=0}^{\lfloor (s-1)/2\rfloor} \sum_{k=2\,j+1}^{\lfloor (s-1)/2\rfloor} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{\lfloor s/2\rfloor}{k}\frac{1}{2^{\lfloor s/2\rfloor}} \\
&\quad + (1-\kappa) \sum_{j=0}^{\lfloor (s-1)/2\rfloor} \sum_{k=2\,j}^{\lfloor (s-1)/2\rfloor} \binom{k}{2\,j}\frac{1}{2^k}\binom{\lfloor s/2\rfloor}{k}\frac{1}{2^{\lfloor s/2\rfloor}} \\
&= (\kappa + (1-\kappa)) \sum_{j=0}^{\lfloor (s-1)/2\rfloor} \sum_{k=2\,j+1}^{\lfloor (s-1)/2\rfloor} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{\lfloor s/2\rfloor}{k}\frac{1}{2^{\lfloor s/2\rfloor}} \quad \text{(by Lemma 2)} \\
&= \sum_{j=0}^{\lfloor s/2\rfloor} \sum_{k=2\,j+1}^{\lfloor s/2\rfloor} \binom{k}{2\,j+1}\frac{1}{2^k}\binom{\lfloor s/2\rfloor}{k}\frac{1}{2^{\lfloor s/2\rfloor}} \\
&= \sum_{k=1}^{\lfloor (s-1)/2\rfloor} \frac{1}{2}\binom{\lfloor (s-1)/2\rfloor}{k}\frac{1}{2^{\lfloor (s-1)/2\rfloor}} \\
&= \frac{1}{2} - \frac{1}{2}\binom{\lfloor (s-1)/2\rfloor}{0}\frac{1}{2^{\lfloor (s-1)/2\rfloor}} = \frac{1}{2} - \frac{1}{2^{\lfloor (s-1)/2\rfloor+1}} \,.
\end{aligned}$$

C The Discrete Logarithm Problem

We recall the discrete logarithm problem:

Definition 1 Let $\alpha \in G$, for some Abelian group G, and suppose $\alpha \in \langle\beta\rangle$. The discrete logarithm $\log_\alpha \beta$ is the unique integer x such that $0 \le x \le \mathrm{ord}(\alpha) - 1$ and $\alpha^x = \beta$. The Discrete Logarithm Problem (DLP) is to compute $\log_\alpha \beta$, given α and β.

In a side-channel analysis of a given instance of an exponentiation algorithm the results can only give the best guess of the exponent. Stinson describes a variant of the Baby-Step/Giant-Step algorithm where it is assumed that the exponent has a small Hamming weight [27]. Stinson's algorithm requires the existence of a means of splitting a string of bits into two sets of equal Hamming weight.

Lemma 3 *We consider an integer of bit length m, as a string of bits of length $m \in 2\,\mathbb{Z}$ and Hamming weight $0 < t < m$. There will exist a set of contiguous bits with Hamming weight $\lfloor t/2 \rfloor$.*

We present a somewhat simplified version of Stinson's proof:

Proof We begin with the case where t is even. Let X be an string of bits of length m with Hamming weight $t \in 2\,\mathbb{Z}$. Let each Y_i for $i \in \{1, \ldots, m/2\}$ represent one of the $m/2$ sets of contiguous bits starting from the i-th bit of the string. Let H be a function that returns the Hamming weight, then $H(Y_1) = t - H(Y_{m/2})$. Given that $H(Y_i) - H(Y_{i+1})$ will be in $\{-1, 0, 1\}$ there will be some set of contiguous bits with Hamming weight $m/2$. If t is odd then the first bit can be ignored as it will be set to one given the bit length is known putting us the case described above. Hence, one can find one set of Hamming weight $\lfloor m/2 \rfloor$ and the other of $\lceil m/2 \rceil$. □

This is sufficient for our requirements. We refer the reader to Stinson for versions of this proof where m is odd [27].

Given an estimate for the exponent x' where $x = x' \oplus e$, for some unknown e of Hamming weight t, we can attempt to determine x by guessing e. We let z_i denote the ith bit of z for an n-bit number z. Given an n-bit number z we define the vector $\mathring{z}$ as follows

$$\mathring{z}_i = \begin{cases} 0 & \text{If } z_i = 0\,, \\ 1 & \text{If } z_i = 1 \text{ and } x'_i = 0\,, \\ -1 & \text{If } z_i = 1 \text{ and } x'_i = 1\,. \end{cases}$$

For a vector $\mathring{z}$ we define

$$g^{\mathring{z}} = \prod_{i=1}^{n} g^{\mathring{z}_i \cdot 2^{n-i}} \quad .$$

If we set $\beta' = \alpha^{x'}$, then given a proposed value of e, such that $x = x' \oplus e$, we can test whether it is correct by checking whether we have $\beta = \beta' \cdot \alpha^{\mathring{e}}$. The error e can be divided into two sets e_1 and e_2, where e_1 and e_2 have a Hamming weight of $t/2$ given by a splitting algorithm. We also define a and b as two integers such that $x' = a + b$ and the only bits that can be set to one for a and b are at the indexes defined by the splitting algorithm for e_1 and e_2 respectively. Then $\alpha^x = (\alpha^a\, \alpha^{\mathring{e}_1})(\alpha^b\, \alpha^{\mathring{e}_2})$.

We produce a list of error vectors of Hamming weight $t/2$ where we define the i-th error from the set of possible errors e_1 as $e_{i,1}$. We define the Giant-Steps to be the table which consists of all pairs $\left(\frac{\beta}{\alpha^a\, \alpha^{\mathring{e}_{i,1}}}, a + \mathring{e}_{i,1}\right)$, for all $e_{i,1}$. We define the Baby-Steps as pairs $\left(\alpha^b\, \alpha^{\mathring{e}_{j,2}}, b + \mathring{e}_{j,2}\right)$, for all $e_{j,2}$. As in the Baby-Step/Giant-Step method we can terminate the method when a collision is found between $\left(\frac{\beta}{\alpha^a\, \alpha^{\mathring{e}_{i,1}}}\right)$ and $\left(\alpha^b\, \alpha^{\mathring{e}_{j,2}}\right)$ for a given i, j. We can then derive the exponent as $x = (a + \mathring{e}_{i,1}) + (b + \mathring{e}_{j,2})$.

For an m-bit exponent one would be required to compute $\binom{m}{t/2}$ Giant-Steps and $\binom{m}{t/2}$ Baby-Steps for an error of Hamming weight t. The above assumes that t is even. If t is odd then the extra bit can be assigned, arbitrarily, to the computation of baby steps. The required computation then becomes $\binom{m}{\lfloor t/2 \rfloor}$ Giant-Steps and $\binom{m}{\lfloor t/2 \rfloor + 1}$ Baby-Steps for an error of Hamming weight t.

Other than the inclusion of an initial guess this algorithm is defined by Stinson [27], and has time complexity of $\mathcal{O}\left(m \binom{m/2}{t/2}\right)$. However, this assumes that t is known.

Typically, t is not known and an adversary has to start with $t = 1$ and increase the Hamming weight until t is found. One would expect the resulting time complexity to be $\mathcal{O}\left(m \sum_{n=0}^{t} \binom{m/2}{n/2}\right)$. However, by Lemma 3 we can ignore the cases where n is odd. Since the required baby and giant steps will be computed for the cases $n-1$ and $n+1$. The resulting time complexity is therefore $\mathcal{O}\left(m\sum_{n=0}^{\lceil t/2\rceil}\binom{m/2}{n}\right)$ when t is unknown.

To derive a private exponent used in RSA [25] the order is not known and the above analysis cannot be applied directly. If we define γ to be the maximum possible bit length of $\mathrm{ord}(\alpha)$. Then the problem can be rewritten as $\alpha^{\gamma+1}\,\alpha^{x} = \alpha^{\gamma+1}\,\beta$. Then the inverse of α^{b} can be replaced by $\alpha^{\gamma+1-b}$ [28].