How to Compress Rabin Ciphertexts and Signatures (and More)

Craig Gentry

DoCoMo USA Labs
cgentry@docomolabs-usa.com

Abstract. Ordinarily, RSA and Rabin ciphertexts and signatures are $\log N$ bits, where N is a composite modulus; here, we describe how to "compress" Rabin ciphertexts and signatures (among other things) down to about $(2/3) \log N$ bits, while maintaining a tight provable reduction from factoring in the random oracle model. The computational overhead of our compression algorithms is small. We also improve upon Coron's results regarding partial-domain-hash signature schemes, reducing by over 300 bits the hash output size necessary to prove adequate security.

1 Introduction

The hardness of factoring is one of the most fundamental and frequently used assumptions of public-key cryptography; yet cryptosystems that rely on the factoring assumption have relatively poor performance in terms of bandwidth. For example, RSA and Rabin ciphertexts and signatures are typically at least as many bits as the composite modulus N, while recent advances in hardware-based approaches to factoring (e.g., [32]) suggest that N must be more than 1024 bits for strong security. So, factoring-based cryptosystems often do not compare favorably with cryptosystems based on alternative hard problems – e.g., ECC for encryption or DSA for signatures.

Bandwidth consumption is important, in part because fundamental limitations of wireless technology put bandwidth at a premium. For example, Barr and K. Asanović [2] note that wireless transmission of a single bit can cost more than 1000 times as much energy as a 32-bit computation. Since battery efficiency is growing relatively slowly, energy consumption (particularly through wireless transmission) may become a significant bottleneck.

Moreover, signal interference places physical limits on how much data can be transmitted wirelessly in a given region. This was not a problem in wired networks. These limitations are compounded by the lossiness of wireless channels, which necessitates *additional* bandwidth in the form of forward error correction (FEC). FEC is particularly important for cryptographic transmissions, where partial recovery of a ciphertext or digital signature is typically useless.

These considerations make compression algorithms very attractive. In fact, in recent years, substantial progress has been made in constructing "compressed" cryptosystems. For example, XTR [22] and CEILIDH [30] both use "compact representations" of certain elements to achieve a bandwidth savings. There are also a variety of hybrid cryptosystems, such as signcryption and aggregate signature schemes, in which multiple cryptographic functionalities are somehow represented by a single, relatively short string. However, although such hybrid cryptosystems exist for RSA and Rabin, none of them breaks the "($\log N$)-bit barrier."

Our Design Goals. In light of these considerations, we would like to construct a compression algorithm that is broadly applicable to factoring-based schemes, such as RSA and Rabin. Ideally, the compression algorithm should allow RSA and Rabin ciphertexts and signatures to be substantially less than $\log N$ bits without sacrificing any security – i.e., while still using (and retaining the security of) a ($\log N$)-bit modulus. Moreover, the compression algorithm should add minimal computational overhead. If the compression algorithm requires additional computation, this computation should not require use of the secret key, so that it can be performed (more quickly) outside of a "secure environment," such as a smart card.

Our Results. We essentially achieve our design goals, except that our techniques work only for Rabin-type cryptosystems, not for RSA. Along the way, we also substantially improve upon Coron's results on partial-domain-hash Rabin signature schemes (Rabin-PDH).

Coron [16] proved the security of a variant of the Rabin signing scheme (Rabin-PDH) in which the hash function that is used to hash the message outputs strings of length $(\frac{2}{3}+\epsilon)\log N$ bits. It turns out that this ϵ has a large effect in practice; if the simulator in the security proof wishes to generate a distribution of signatures whose statistical distance from uniform is less than 2^{-80}, Coron's method requires that the hash output length be at least at $\frac{2}{3}\log N+364$ bits. We provide a perfectly uniform drawing algorithm that reduces the necessary hash output length to only $\frac{2}{3}\log N+3$ bits; moreover, our security proof is tighter.

Our main result, however, is a compression algorithm that allows a 33% reduction in the bit-length of Rabin signatures and ciphertexts, without any sacrifice in security. (Notice that Coron's result is not a compression algorithm; although the hash output length of Coron's Rabin-PDH scheme may be less than $\log N$ bits, the Rabin-PDH signature itself, which is essentially a modular square root of the hash output, is a ($\log N$)-bit value.) For our improved version of Rabin-PDH signatures, the "entropy" of the hash output is just over $\frac{2}{3}\log N$ bits; thus, it is *theoretically* possible that the signature could also be expressed in about $\frac{2}{3}\log N$ bits. In fact, up to the loss of a few bits, this is precisely what we achieve: a $(\frac{2}{3}\log N+6)$-bit Rabin-PDH signature, with a tight reduction from factoring N.

Our lossless compression algorithm also works for Rabin encryption, but in reverse. A $(\frac{2}{3}\log N)$-bit plaintext is "decompressed" by mapping it to a ($\log N$)-bit number that has a $(\frac{2}{3}\log N+3)$-bit modular square. This modular square is a "compressed" Rabin ciphertext. Numerous other cryptosystems also involve computing square roots modulo a composite modulus N, including Fiat-Shamir, Cocks's identity-based encryption scheme, as well as various schemes enabling ring signatures, signcryption, and so on. Our techniques enable a similar 33% bandwidth reduction for these schemes.

Related Work. Like Coron's work, our techniques build upon Brigitte Vallée's elegant analysis of the distribution, in $\mathbb{Z}/N\mathbb{Z}$, of integers in $B_{N,h,h'} = \{x \in [1, N) : h \leq x^2(\text{mod}N) < h'\}$ for $h' - h \leq 8N^{2/3}$ – i.e., integers with modular squares in a "narrow" interval. We provide a self-contained discussion of her results in section 3.

Some previous work has been done on compressing Rabin and low-exponent RSA signatures – in particular, Bernstein [7] mentions that one can simply remove the $\frac{1}{e}\log_2 N$ least significant bits of any regular Rabin or RSA signature, and the verifier can use Coppersmith's method [17] to recover those bits. Bleichenbacher [8] describes an improvement: the signer can use continued fractions to express the signature s as $a/b(\text{mod}N)$, where a is about $\frac{e-1}{e}\log_2 N$ bits and b is about $\frac{1}{e}\log_2 N$ bits, and send a as the signature. The verifier checks that $c = a^e/H(m)(\text{mod}N)$ is an e^{th} power (namely b^e) over $\mathbb{Z}$. The drawback of these methods, though they arguably reduce Rabin signature length to $\frac{1}{2}\log_2 N$ bits, is that they do not allow message recovery; the verifier needs m before verifying, which effectively adds to the signature length. These methods also do not appear to be very broadly applicable; e.g., they do not appear to lead to low-bit-length encryption, signcryption and aggregate signature schemes.

As mentioned above, Coron [16] uses a "compressed" output space for the hash function in a Rabin signature scheme, but the partial-domain hash signatures themselves are still $\log N$ bits.

Organization of the Paper. This paper is organized as follows. After noting some preliminaries in Section 2, we describe Vallée's distributional observations and her "quasi-uniform" drawing algorithm in section 3. In section 4, we describe our perfectly uniform drawing algorithm, and our improvement upon Coron's results regarding Rabin-PDH. We describe our compression algorithm in section 5, after which we describe compressed Rabin encryption and signature schemes in section 6. Finally, in Section 7, we mention other cryptosystems – such as signcryption, aggregate signature and ring signature schemes – for which our compression algorithm allows a 33% bandwidth reduction.

2 Preliminaries

We gather some mathematical notation here for convenience. Let $\{0,1\}^*$ denote the set of all bit strings, and let $\{0,1\}^n$ denote the set of all bit-strings of length n. For a real number r, $\lceil r \rceil$ denotes the ceiling of r, that is, the smallest integer value greater than or equal to r. Similarly, $\lfloor r \rfloor$ denotes the floor of r, that is, the largest integer value less than or equal to r. Finally, $\lfloor r \rceil$ denotes the closest integer to r. Let the symbol $\|$ denote concatenation.

Throughout, N will denote a suitable integer modulus. To be suitable, N should at least be computationally hard to factor using any modern factoring algorithm. In practice, one often generates N as the product of two large prime numbers p and q – e.g., 512 bits apiece. However, one could choose N differently for our schemes, if desired. For example, setting $N = p^d q$ for $d > 1$ can lead to efficiency advantages, though one should be wary of setting d too large [11].

Let $B_{N,h,h'} = \{x \in [1, N] : h \leq x^2 (\mathrm{mod} N) < h'\}$ for integers h and h' and suitable modulus N – i.e., the set of integers with modular squares in $[h, h')$. Let B be shorthand for $B_{N,h,h'}$ when N, h and h' are understood.

A "lattice" consists of the set of all vectors that can be generated as integer linear combinations of a set of basis vectors. For example, if (a, b) and (c, d) are two basis vectors in two-dimensional space, the lattice that they generate is the set of vectors $\{(k_1 a + k_2 c, k_1 b + k_2 d) : k_1, k_2 \in \mathbb{Z}\}$.

3 Distribution of Numbers with Small Modular Squares

Developing a compressed representation of numbers in $B_{N,h,h'}$ that is efficiently computable and invertible requires an understanding of how numbers in $B_{N,h,h'}$ are distributed in $[0, N/2)$. The compression algorithm works, at a high level, by taking this distribution into account.

In [33], Vallée describes the "global" distribution of $B_{N,h,h'}$ in $[0, N/2)$ in terms of its "local" distribution in each of a set of *Farey intervals* that covers $[0, N/2)$. She then describes each local distribution in terms of points of a lattice that lie in the region between two parabolas. For $h' - h \geq 8N^{2/3}$, the distribution of $B_{N,h,h'}$-elements among the Farey intervals is "quasi-independent," allowing her to construct an algorithm that draws integers from $B_{N,h,h'}$ "quasi-uniformly." Since Vallée's analysis forms the basis of our compression algorithm, we review it in detail in this section.

3.1 Farey Sequences

Some properties of Farey sequences are collected in [20]; we recall them below.

Definition 1 (Farey Sequence). *The Farey sequence* $\mathcal{F}_k$ *of order* k *is the ascending sequence* $(\frac{0}{1}, \frac{1}{k}, \ldots, \frac{1}{1})$ *of fractions* $\frac{a_i}{b_i}$ *with* $1 \leq a_i \leq b_i \leq k$ *and* $gcd(a_i, b_i) = 1$.

The characteristic property of Farey sequences is expressed in the following theorem [20]:

Theorem 1. *If* $\frac{a_i}{b_i}$ *and* $\frac{a_{i+1}}{b_{i+1}}$ *are consecutive in* $\mathcal{F}_k$*, then* $b_i a_{i+1} - a_i b_{i+1} = 1$.

Another useful theorem concerning Farey sequences is the following:

Theorem 2. *If* $\frac{a_i}{b_i}$ *and* $\frac{a_{i+1}}{b_{i+1}}$ *are consecutive in* $\mathcal{F}_k$*, then* $b_i + b_{i+1} > k$.

The latter theorem follows from the fact that $(a_i + a_{i+1})/(b_i + b_{i+1})$, the so-called "mediant" of a_i/b_i and a_{i+1}/b_{i+1}, is between a_i/b_i and a_{i+1}/b_{i+1} and would be in $\mathcal{F}_k$ if $b_i + b_{i+1} \leq k$. Farey sequences lead naturally to the notion of a Farey partition, in which the set of mediants partition the interval $[0, N/2)$ into subintervals. The formal definition is as follows.

Definition 2 (Farey Partition). *The Farey partition of order k of the interval $[0, N/2)$ is the set of intervals $J(a_i, b_i) = [\frac{(a_{i-1}+a_i)N}{2(b_{i-1}+b_i)}, \frac{(a_i+a_{i+1})N}{2(b_i+b_{i+1})})$, where $\frac{a_i}{b_i}$ is the i-th term in $\mathcal{F}_k$.*

So that each "end" of $[0, N/2)$ is covered by the partition, we set $(a_0, b_0) = (a_1, b_1)$ and $(a_{z+1}, b_{z+1}) = (a_z, b_z)$, where $a_z/b_z = 1/1$ is the final fraction in the Farey sequence.

Vallée found it convenient to use another set of intervals $I(a_i, b_i)$, called "Farey intervals," that are related to $J(a_i, b_i)$.

Definition 3 (Farey Interval). *The Farey interval $I(a_i, b_i)$ of order k is the open interval with center $\frac{a_i N}{2b_i}$ and radius $\frac{N}{2kb_i}$, where $\frac{a_i}{b_i}$ is the i-th term in $\mathcal{F}_k$.*

Using Theorems 1 and 2, one can easily prove that $I(a_i, b_i)$ contains $J(a_i, b_i)$, and that the interval $I(a_i, b_i)$ is no more than twice as wide as the interval $J(a_i, b_i)$ [1]. One can also prove that every number in $[0, N/2)$ is covered by at least one, and at most two, Farey intervals – e.g., by showing that, for every i, $I(a_{i-1}, b_{i-1})$ intersects $I(a_i, b_i)$, but neither $I(a_{i-1}, b_{i-1})$ nor $I(a_{i+1}, b_{i+1})$ contains the center of $I(a_i, b_i)$. Vallée probably favored using the Farey intervals rather than the $J(a_i, b_i)$ in her analysis, because (roughly speaking) the fact that each $I(a_i, b_i)$ is symmetric about $a_i N/2b_i$ makes her analysis cleaner. A "Farey Covering," which is analogous to a Farey partition, is then defined as follows.

Definition 4 (Farey Covering). *The Farey covering of order k of the interval $[0, N/2)$ is the set Farey intervals $I(a_i, b_i)$ of order k.*

3.2 The Connection between Farey Sequences and B's Distribution

Although it is far from obvious, Farey sequences have a close connection with the distribution in $\mathbb{Z}/N\mathbb{Z}$ of integers in $B_{N,h,h'}$. Vallée observed that the gaps between consecutive integers in B vary widely close to the rationals $a_i N/2b_i$ of small denominator b_i. Close to these rationals, the distribution might be called "clumpy," with large gaps separating sequences of small gaps. However, as one considers wider intervals centered at $a_i N/2b_i$, the distribution of B-elements provably "evens out" – i.e., the ratio of the number of B-elements in the interval, versus the number one would expect if the B-elements were distributed uniformly, approaches 1. Roughly speaking, the width of interval needed before the "clumpiness" can be disregarded is inversely proportional to b_i. This is one reason why Farey intervals are useful for analyzing B's distribution; the diameter of $I(a_i, b_i)$ is also inversely proportional to b_i.

Building on the above observations, Vallée ultimately proved that the number of $B_{N,h,h'}$-elements in $I(a_i, b_i)$ is essentially proportional to the width of $I(a_i, b_i)$ (as one would expect), as long as $h' - h$ is large enough. Formally, Vallée proved the following theorem [33].

Theorem 3. *For $-h = h' \geq 4N^{2/3}$ and $k = \frac{N}{h'}$, the subset $B_{N,h,h'}$ and the Farey covering of order k are quasi-independent.*

Vallée defines *quasi-independence* as follows.

Definition 5 (Quasi-Independence). *A subset X and a covering $\mathcal{Y} = \{Y_j\}$ of $\mathbb{Z}_N$ are quasi-independent if, for all j, the sets X and Y_j are (l_1, l_2)-independent for some positive constants l_1 and l_2 – i.e., $l_1 \leq \frac{P(X \cap Y_j)}{P(X)P(Y_j)} \leq l_2$.*

Clearly, this definition is meaningless unless l_1 and l_2 are independent of N. Vallée proves that $l_1 = \frac{1}{5}$ and $l_2 = 4$ suffice when $-h = h' \geq 4N^{2/3}$ and $k = \frac{N}{h'}$. This means that, for these parameters, any given Farey interval has no more than $l_2/l_1 = 20$ times the "density" of $B_{N,h,h'}$-elements than any other Farey interval.

Interestingly, Vallée's proof of Theorem 3 is essentially constructive. To analyze the distribution of $B_{N,h,h'}$-elements in the "local" region $I(a_i, b_i)$, Vallée associates each $B_{N,h,h'}$-element with a point that is in a particular lattice and that lies in the region between two particular parabolas. She then partitions the lattice into a set of parallel lines. The number of lines may be very large – e.g., superpolynomial in $\log N$. Her distribution analysis then becomes "even more local"; she provides upper and lower bounds on how many associated lattice points can occur on each line (except for at most 6 of the lines, for which she only provides upper bounds). These bounds imply similar bounds on the number of $B_{N,h,h'}$-elements in $I(a_i, b_i)$. Her constructive approach results in what one may call a "quasi-enumeration" of $B_{N,h,h'}$-elements in $I(a_i, b_i)$, in which each element is indexed first by the line of its associated lattice point, and then by the lattice point's position on the line. This quasi-enumeration is crucial to Vallée's "quasi-uniform" drawing algorithm (subsection 3.3), to our uniform drawing algorithm (section 4), and to our algorithms for losslessly compressing $B_{N,h,h'}$-elements (section 5).

Before discussing these algorithms, we review the details of Vallée's analysis. Set x_0 to be the closest integer to $\frac{a_i N}{2b_i}$ (the center of the Farey interval). If $x = x_0 + u$ is in $B_{N,h,h'}$, then $h \leq x_0^2 + 2x_0 u + u^2 (\bmod N) < h'$. Now, let $L(x_0)$ be the lattice generated by the vectors $(1, 2x_0)$ and $(0, N)$. Then, $x = x_0 + u$ is in $B_{N,h,h'}$ precisely when there is a w such that $(u, w) \in L(x_0)$ and $h \leq x_0^2 + w + u^2 < h'$. The latter requirement implies that (u, w) is in between the two parabolas defined, in variables u' and w', by the formulas $x_0^2 + w' + u'^2 = h$ and $x_0^2 + w' + u'^2 = h'$. Thus, if we set $u_0 = x_0 - \frac{a_i N}{2b_i}$, then each $x \in B_{N,h,h'} \cap I(a_i, b_i)$ corresponds to a lattice point in:

$$P(a_i, b_i) = \{(u, w) \in L(x_0) \ : \ |u + u_0| \leq \frac{h}{2b_i} \text{ and } h \leq x_0^2 + w + u^2 < h'\}. \quad (1)$$

It may seem like a fairly complicated task to approximate how many lattice points in $L(x_0)$ are between the two parabolas defined above,[1] but, as Vallée describes, it is possible to find a lattice basis of $L(x_0)$ in which the basis vectors are each short, with one basis vector being "quasi-horizontal" and the other

[1] Indeed, finding all of the $L(x_0)$ points on a *single* parabola is equivalent to finding all of a number's modular square roots, which is equivalent to factoring.

being "quasi-vertical." The basis is $(\boldsymbol{r}, \boldsymbol{s})$ with:

$$\boldsymbol{r} = b_i(1, 2x_0) - a_i(0, N) = (b_i, 2b_i u_0) \ , \tag{2}$$

$$\boldsymbol{s} = b_{i-1}(1, 2x_0) - a_{i-1}(0, N) = (b_{i-1}, \frac{N}{b_i} + 2b_{i-1}u_0) \ . \tag{3}$$

Recall that $|u_0| \leq \frac{1}{2}$, and $b_i \leq k$ with $k = \frac{1}{4}N^{1/3}$.

Having computed this short lattice basis, Vallée considers the distribution of $P(a_i, b_i)$-points (and hence B-elements) on individual lines parallel to $vecr$. Each point in $P(a_i, b_i)$ lies on a quasi-horizontal line that intersects the vertical axis at ordinate $w_0 - vN/b_i$ for some rational index $v \in [0, (h'-h)^2/16b_iN + (h'-h)b_i/N]$, where $w_0 = h' - x_0^2 + u_0^2$ and where consecutive indices differ by 1. For lines with indices from $v_1 = \lceil 2(h'-h)b_i/N \rceil$ to $v_2 = \lfloor (h'-h)^2/16b_iN \rfloor$, which intersect the region between the two parabolas in an area she dubs the "legs" (which is in between the "chest" and the "feet"), Vallée proves the following theorem:

Theorem 4. *The number $n(v)$ of points in $P(a_i, b_i)$ on the line with index v in the legs satisfies:* $\frac{1}{2}\frac{(h'-h)}{\sqrt{vb_iN}} \leq n(v) \leq \frac{7}{4}\frac{(h'-h)}{\sqrt{vb_iN}}$.

Her bounds on each individual line in the legs imply lower and upper bounds on the total number of lattice points in the legs, using the inequalities:

$$\sum_{v=v_1}^{v_2} \frac{1}{\sqrt{v}} \geq \int_{v_1}^{v_2+1} \frac{dv}{\sqrt{v}} = 2(\sqrt{v_2+1} - \sqrt{v_1}) \ , \tag{4}$$

$$\sum_{v=v_1}^{v_2} \frac{1}{\sqrt{v}} \leq \frac{1}{\sqrt{v_1}} + \int_{v_1+1}^{v_2} \frac{dv}{\sqrt{v}} = \frac{1}{\sqrt{v_1}} + 2\sqrt{v_2} \ . \tag{5}$$

For lines with indices in $[0, 2(h'-h)b_i/N]$ or $[(h'-h)^2/16b_iN, (h'-h)^2/16b_iN + (h'-h)b_i/N]$ that intersect the "chest" or "feet," Vallée provides no nontrivial *lower* bounds on the number of $P(a_i, b_i)$-points they may contain, only upper bounds. For $h'-h = 8N^{2/3}$, one can verify Vallée's results that there are at most 4 lines in the chest, each with fewer than $\frac{2}{b_i}\sqrt{(v_1-1)N/b_i} + 1$ points, and that there are at most 2 lines in the feet, each with fewer than 8 points. Ultimately, Vallée proves Theorem 3 using her lower bounds for the legs, and upper bounds for the chest, legs and feet.

3.3 Vallée's Quasi-Uniform Drawing Algorithm

Vallée uses the above results, particularly her lower and upper bounds for the legs, to obtain a concrete algorithm for drawing integers from $B_{N,h,h'}$ quasi-uniformly when $h'-h \geq 8N^{2/3}$. For a *quasi-uniform* drawing algorithm, the respective probabilities of any two $B_{N,h,h'}$-elements being drawn are within a constant factor of each other; formally:

Definition 6 (Quasi-Uniform). *A drawing algorithm C, defined over a finite set U and with values in a subset X of $\mathbb{Z}_N$, is said to be (l_1, l_2)-uniform (or quasi-uniform) for constants l_1 and l_2 if, for all $x \in X$,* $\frac{l_1}{|X|} \leq \Pr[u \leftarrow U \mid C(u) = x] \leq \frac{l_2}{|X|}$.

Vallée's algorithm is as follows:

1. *Randomly Select a Starting Point:* Pick random integer $x \in [0, N/2)$ with uniform distribution.
2. *Determine Farey Interval:* Use continued fractions to compute (a_i, b_i) for which $x \in J(a_i, b_i)$.
3. *Evaluate the Number of Points in $P(a_i, b_i)$:* Compute $x_0 = \lfloor \frac{a_i N}{b_i} \rceil$, count exactly the number n_{c+f} of points in the chest and feet, and obtain a lower bound n_l on the number of points in the legs using Vallée's lower bounds (with Equation 4).
4. *Pick a Point from $P(a_i, b_i)$:* Randomly select an integer in $t \in [1, n_{c+f} + n_l]$ with uniform distribution. If $t \leq n_{c+f}$, output the appropriate point from the chest or feet. Else, use Equation 4 to determine which quasi-horizontal line would contain the n_l^{th} point in the legs if each line met Vallée's lower bounds, and randomly choose a point in $P(a_i, b_i)$ on that line with uniform distribution.
5. *Compute x' from the Chosen Point in $P(a_i, b_i)$:* Let (u, w) be the lattice point output by the previous step. Set $x' = x_0 + u$.

Remark 1. In Step 3, one can quickly can get an exact count for how many points are in the chest and the feet by counting the *exact* number of points on each line, using simple geometry. (Recall that there are at most 4 lines in the chest, 2 in the feet.) A line intersects one of the two parabolas in at most 4 locations, possibly cutting the line into two segments that lie in between the parabolas. After finding the first and last lattice points on each segment, extrapolating the total number of points on each segment is easy since the x-coordinates of consecutive lattice points differ by b_i (see Equation 2). Vallée avoids counting the number of points on lines in the legs, since the number of lines in the legs may be super-polynomial in $\log N$.

The drawing algorithm outputs an $x' \in B_{N,h,h'}$ that is in the same $J(a_i, b_i)$ interval as x. A wider interval (recall that $I(a_i, b_i)$ has diameter $\frac{N}{b_i k}$ for $1 \leq b_i \leq k$, and that $J(a_i, b_i)$ is at least half as wide as $I(a_i, b_i)$) has a higher chance of being chosen in the first two steps. However, once an interval is chosen, any given B-element in that interval has a lower probability of being chosen if the interval is wide than if it is narrow. On balance, these factors even out (this is quasi-independence), and the drawing algorithm is quasi-uniform.

In computing l_2/l_1, there are three things to consider. First, different Farey intervals may have different "densities" of $B_{N,h,h'}$-elements; specifically, the ratio may be as much as 20 (see discussion after Theorem 3). Second, in Step 2, we used $J(a_i, b_i)$ rather than $I(a_i, b_i)$; since $I(a_i, b_i)$ is between 1 and 2 times as wide as $J(a_i, b_i)$, this costs us another factor of 2. Finally, within the $J(a_i, b_i)$ interval, different lines may be closer to the lower bounds or closer to the upper bounds, leading to a factor of $\frac{7/4}{1/2} = \frac{7}{2}$. Thus, l_2/l_1 is at most $20 \cdot 2 \cdot \frac{7}{2} = 140$.

4 Improving Vallée's and Coron's Results

In this section, we describe how to modify Vallée's quasi-uniform drawing algorithm to make it perfectly uniform. Our perfectly uniform drawing algorithm gives us an immediate improvement upon Coron's proof of security for Rabin-PDH; in particular, it allows us to reduce the output size of the partial domain hash function (see subsection 4.2). More generally, the fact that a simulator can draw B-elements uniformly in responding to an adversary's hash queries allows us (when combined with the compression schemes of Section 5) to reduce the bandwidth of several signature-related cryptosystems, including aggregate signature schemes, ring signature schemes and signcryption schemes.

4.1 A Perfectly Uniform Drawing Algorithm

Modifying Vallée's quasi-uniform drawing algorithm to make it perfectly uniform is surprisingly simple. Our modification is based on our observation that, for any $B_{N,h,h'}$-element (with $h'-h \geq 8N^{2/3}$, as required by Vallée), anyone can efficiently compute the *exact* probability $P_{x'}$ that Vallée's quasi-uniform drawing algorithm will output x'. For example, a simulator in a security proof can compute this probability (without, of course, needing the factorization of N).

Assume, for now, that we can efficiently compute $P_{x'}$ for any given x'. Let P_{min} be a lower bound on such probabilities over all $B_{N,h,h'}$-elements. Then, the improved drawing algorithm is as follows:

1. Use Vallée's method to pick an $x' \in B_{N,h,h'}$ quasi-uniformly.
2. Compute $P_{x'}$.
3. Goto Step 1 with probability $(P_{x'} - P_{min})/P_{x'}$.
4. Otherwise, output x'.

Since Vallée's drawing algorithm is quasi-uniform, the expected number of "Goto" loops per draw is a small constant; thus, the simulator's estimated time-complexity increases only by a constant factor. The probability that x' is chosen in Step 1 and that it "survives" Step 3 is the same for all x' – namely, $P_{x'} \cdot (1 - \frac{P_{x'} - P_{min}}{P_{x'}}) = P_{min}$; for this reason, and since each run of Vallée's algorithm is independent, the algorithm is perfectly uniform.

Now, given x', how does one (say, a simulator) compute $P_{x'}$? First, the simulator determines the at most two Farey intervals $I(a_i, b_i)$ and $I(a_{i+1}, b_{i+1})$ that contain x'. For $I(a_i, b_i)$, the simulator computes the index v_i of the quasi-horizontal line l_{v_i} that contains the lattice point (u_i, w_i) associated to x', and the exact number $n(v_i)$ of lattice points on l_{v_i}. Similarly, if there is a second Farey interval $I(a_{i+1}, b_{i+1})$ that contains x', the simulator computes v_{i+1}, $l_{v_{i+1}}$, (u_{i+1}, w_{i+1}), and $n(v_{i+1})$. Then, using the variables x and t from Vallée's drawing algorithm, the probability that x' will be chosen is:

$$(Pr[x \in J(a_i, b_i)]) \cdot (Pr[t_i \in l_{v_i} \mid x \in J(a_i, b_i)]) \cdot \left(\frac{1}{n(v_i)}\right) +$$

$$(Pr[x \in J(a_{i+1}, b_{i+1})]) \cdot \left(Pr[t_{i+1} \in l_{v_{i+1}} \mid x \in J(a_{i+1}, b_{i+1})]\right) \cdot \left(\frac{1}{n(v_{i+1})}\right) ,$$

where we use $Pr[t_i \in l_{v_i}]$ to denote the probability that the choice of t in Step 4 of Vallée's algorithm will map to the line l_{v_i}.

Remark 2. So that the above terminology works when (u_i, w_i) (or (u_{i+1}, w_{i+1})) lies in the chest or feet, we can pretend that these n_{c+f} points lie on a single "line."

Focusing on the first summand in the expression above, the simulator can compute each of the two probabilities in this term efficiently. First, the simulator computes the number of integers in $J(a_i, b_i)$; denoting this number by j_i, $Pr[x \in J(a_i, b_i)]$ is simply $j_i/\lceil N/2 \rceil$. Next, for the second probability, suppose that $n_{c+f} + n_l$ is the approximation used in Step 4 of Vallée's algorithm derived from her lower bounds (namely, $n_l = \lceil \frac{(h'-h)}{\sqrt{b_i N}}(\sqrt{v_2+1} - \sqrt{v_1}) \rceil$) for the legs, and that $n_{v_i} = \lceil \frac{(h'-h)}{\sqrt{b_i N}}\sqrt{v_i+1} \rceil - \lceil \frac{(h'-h)}{\sqrt{b_i N}}\sqrt{v_i} \rceil$ is her approximation for the number of points on l_{v_i}. (Warning: our v_i notation collides here with Vallée's definition of v_1 and v_2.) Then, $Pr[t \in l_{v_i} \mid x \in J(a_i, b_i)] = n_{v_i}/(n_{c+f} + n_l)$. In a similar fashion, the simulator can compute the necessary probabilities for $I(a_{i+1}, b_{i+1})$, thereby obtaining a perfectly uniform drawing algorithm.

Vallée was presumably content with finding a quasi-uniform drawing algorithm, since a uniform algorithm would not have improved her result of a provable $\exp(\sqrt{(4/3)\log n \log\log n})$-time factoring algorithm by a significant amount. However, as described below, our uniform drawing algorithm has a significant practical impact on Coron's partial-domain hash variant of Rabin's signature scheme.

4.2 Improving Coron's Results for Rabin-PDH

Coron [16] provided a random-oracle security proof for a partial-domain hash Rabin signature scheme (Rabin-PDH), in which the signature x' is a modular square root (up to a fudge factor) of $\gamma \cdot H(m) + f(m)$, where H is a partial-domain hash with output space $[0, N^\beta]$ for $\frac{2}{3} + \epsilon \leq \beta < 1$, f is a possibly constant function, and γ is a constant. In Rabin signing, a common fudge factor is to accept the signature if $x'^2 \equiv c(\gamma \cdot H(m) + f(m))(\mathrm{mod} N)$ for any $c \in \{-2, -1, 1, 2\}$, when $N = pq$ for $p \equiv 3(\mathrm{mod} 8)$ and $q \equiv 7(\mathrm{mod} 8)$. In this case, x' is an integer in $B_{N,h,h'}$ for $h = cf(m)$ and $h' = h + c\gamma N^\beta$ if $c\gamma$ is positive, or for $h = h' + c\gamma N^\beta$ and $h' = cf(m)$ if $c\gamma$ is negative. Coron's proof requires that γ be very small in magnitude (e.g., 16 or 256) [16], so that $h' - h = |c\gamma N^\beta|$ is sufficiently small. One reason that Rabin-PDH was an interesting problem for Coron to analyze was that partial-domain hashes were *already being used* by standardized encoding schemes. For example, ISO 9796-2 defined the encoding $\mu(m) = 4\mathrm{A}_{16} \| m \| H(m) \| \mathrm{BC}_{16}$.

As mentioned above, Coron provides a proof of security for Rabin-PDH when $h' - h$ is at least $(\frac{2}{3} + \epsilon)\log N$ bits, but this "ϵ" can be quite large in practice. Coron's security proof relies completely on his algorithm for drawing integers from $B_{N,h,h'}$ with a distribution whose distance from uniform is at most $16N^{\frac{-3\epsilon}{13}}$.

This statistical distance must be very small, so that an adversary cannot distinguish a real attack from a simulated attack, in which the simulator uses Coron's drawing algorithm to respond to hash queries. For the statistical distance to be at most 2^{-k}, we must have that $4 - \frac{3\epsilon}{13}\log N \leq -k$, which implies that $\epsilon \geq \frac{13(k+4)}{3\log N}$. This implies that $h' - h$ is at least $(\frac{2}{3} + \epsilon)\log N = \frac{2}{3}\log N + \frac{13(k+4)}{3}$ bits. When $k = 80$, for example, $h' - h$ must be at least $\frac{2}{3}\log N + 364$ bits. This means that, for $k = 80$, Coron's technique does not reduce the minimum output size of the hash function at all, until N is at least $3 \cdot 364 = 1092$ bits!

We get a better, and much more practical, provable security result by using our perfectly uniform drawing algorithm. In particular, since our algorithm allows us to draw $B_{N,h,h'}$-elements uniformly for $h' - h \geq 8N^{2/3}$, we can prove a reduction from factoring to Rabin-PDH when $h' - h$ is only $\frac{2}{3}\log N + 3$ bits, over 300 bits less than Coron's result for $k = 80$! Moreover, the proof of security is tighter than Coron's proof for two reasons: 1) the adversary cannot possibly distinguish the simulated distribution from uniform; and 2) Coron's proof, which adapts his proof for RSA-FDH [15], does not provide a tight reduction from factoring (cf. Bernstein [6]).

For completeness, we prove the security of a specific variant of our improved Rabin-PDH, though it should be clear that our drawing algorithm can work with essentially any variant. We pick the one (succintly) described below for its simplicity. Other variants may have advantages; e.g., Bernstein's [6] security reduction is tighter by a small constant, and Bellare and Rogaway [3] describe an encoding scheme that allows (at least partial) recovery of the message being signed.

Let N be the public key, with $N = pq$ for $p \equiv 3(\mathrm{mod} 8)$ and $q \equiv 7(\mathrm{mod} 8)$. Let $A_{a,b}$ be the unique number modulo N that satisfies $A_{a,b} \equiv a(\mathrm{mod} p)$ and $A_{a,b} \equiv b(\mathrm{mod} q)$. Let $H_1 : \{0,1\}^* \to [h, h')$ be the partial-domain hash function with $h'' = h' - h(\mathrm{mod} N) \geq 8N^{2/3}$, and $H_2 : \{0,1\}^* \to \{A_{\pm 1, \pm 1}\}$ be a keyed hash function, with the key known only to the signer. To sign M, the signer first computes $m = H_1(M)$, and then:

1. Sets $s' = m^{(n-p-q+5)/8} \bmod n$ if $\left(\frac{m}{N}\right) = 1$; else, sets $s' = (m/2)^{(n-p-q+5)/8}$;
2. Sends $s = s' \cdot H_2(m) \bmod n$.

To verify, the recipient checks that either $s^2 \equiv \pm H_1(M)(\mathrm{mod} N)$ or $s^2 \equiv \pm 2 \cdot H_1(M)(\bmod N)$. This scheme can be easily modified, à la Bernstein [6], to avoid the computation of Jacobi symbols.

In Appendix A, we prove the following theorem.

Theorem 5. *Assume that there is a chosen-message attack adversary $\mathcal{A}$ that breaks our Rabin-PDH scheme for modulus N in time t with probability ϵ. Then, in the random oracle model, there is an algorithm $\mathcal{B}$ that factors N in time t' with probability ϵ', where $\epsilon' \geq \frac{1}{2}\epsilon(1 - \frac{1}{h''})$, and $t' = O(t + q_H \log^2 N)$.*

5 The Compression Algorithms

In the previous section, we reduced the permissible output size of the hash function in Rabin-PDH to about $\frac{2}{3}\log N$ bits, but Rabin-PDH *signatures* are still $\log N$ bits. In this section, we describe compression algorithms that allow us to compress not only Rabin-PDH signatures, but also Rabin ciphertexts (not to mention aggregate signatures, ring signatures, signcryptions, and so on).

A prerequisite of any compression algorithm is to understand the distribution of what is being compressed. Vallée gives a constructive characterization of the distribution, in $\mathbb{Z}/N\mathbb{Z}$, of integers in $B_{N,h,h'}$; we leverage her characterization to construct a lossless compression algorithms. Roughly speaking, we associate $B_{N,h,h'}$-elements to strings of about $\log_2(h'-h)$ bits that specify the $B_{N,h,h'}$-element's Farey interval and its "address" (according to Vallée's rough enumeration) within that interval. For a B-element in a wider Farey interval, we use fewer bits of the bit string to specify the Farey interval and more bits to specify its address; on balance, it evens out.

Our compression algorithms involve two *nondeterministic quasi-bijections*, $\theta : B_{N,h,h'} \times \mathcal{D} \rightarrow \{0,1\}^{c_2+\log_2(h'-h)}$ (used in the signature schemes) and $\pi : \{0,1\}^{-c_1+\log_2(h'-h)} \times \mathcal{D} \rightarrow B_{N,h,h'}$ (used in the encryption scheme), for small nonnegative constants c_1 and c_2. These mappings are not actual bijections; we call them "nondeterministic quasi-bijections" since the image of an element under each mapping or its inverse has a small constant cardinality; formally:

Definition 7 (Nondeterministic Quasi-Bijection). *For sets $(\mathcal{X}, \mathcal{D}, \mathcal{Y})$ and constants (l_1, l_2, l_3, l_4), we say $\pi : \mathcal{X} \times \mathcal{D} \rightarrow \mathcal{Y}$ is an (l_1, l_2, l_3, l_4)-nondeterministic-quasi-bijection if:*

1. *For all $x \in \mathcal{X}$, the cardinality of $\{\pi(x,d) : d \in \mathcal{D}\}$ is in $[l_1, l_2]$.*
2. *For all $y \in \mathcal{Y}$, the cardinality of $\{x : \exists d \in \mathcal{D} \text{ with } \pi(x,d) = y\}$ is in $[l_3, l_4]$.*

Above, $\mathcal{D}$ is an auxiliary set – e.g., it may be used as a source of (a small number of) random dummy bits if one wishes to make π randomized. The purpose of $\mathcal{D}$ is simply to make π an actual "mapping," with a single output for a given input (even though for a single $x \in \mathcal{X}$ there may be multiple outputs). Notice that an actual bijection is a $(1,1,1,1)$-quasi-bijection.

Roughly speaking, our signature scheme uses θ to compress, without loss, a Rabin-PDH signature (an element of $B_{N,h,h'}$) to a short bit string. Since the "entropy" of the hash output in Rabin-PDH is about $\frac{2}{3}\log N$ bits, one may hope that a Rabin-PDH signature can also be this short; in fact, within a few bits, this is precisely the case. To verify the compressed signature, it is decompressed to recover the ordinary Rabin-PDH signature, which is then verified in the normal fashion. Our encryption scheme uses π to map encoded bit strings to integers in $B_{N,h,h'}$, which are then squared to create short ciphertexts. Both θ and π are efficiently computable and efficiently invertible – i.e., it is easy to recover x from $\pi(x,d)$ or x' from $\theta(x',d)$ – without any trapdoor information.

Why don't we just replace π with θ^{-1}? Indeed, we could if θ were a bijection, but (unfortunately) θ maps each $B_{N,h,h'}$-element to possibly several short strings; if we used θ^{-1} to map short encoded messages to $B_{N,h,h'}$-elements, multiple plaintexts would correspond to the same ciphertext, which we wish to avoid. Thus, although the only real difference between π and θ^{-1} is that we reduce the size of π's domain to ensure that it is an injection, we find it convenient to keep the notation separate.

5.1 Mapping B-Elements to Short Strings (The θ Quasi-Bijection)

Below, we give one approach to the θ quasi-bijection. Roughly speaking, $\theta(x', d)$ re-expresses a $B_{N,h,h'}$-element x' according to its Farey interval and its "address" (using Vallée's lattice) within the Farey interval. For example, a "naive" way to re-express x' is as (a_i, b_i, v, l), where (a_i, b_i) defines x''s Farey interval, v is the index of the quasi-horizontal line that contains the lattice point associated to x', and l represents the lattice point's position on the line. In this format, x' has at most two representations, one corresponding to each Farey interval that contains x'; the only effect of "d" is to pick one of these representations. We describe a different format below that has tighter compression and does not suffer from the parsing problems of the naive approach.

The θ quasi-bijection below maps $x' \in B_{N,h,h'}$ to a short string in $[0, h'']$, where h'' is a parameter whose value will be calibrated later.

Computing $\theta(x', d)$:

1. Determine (a_i, b_i) for which x' is in $J(a_i, b_i)$.
2. Compute x_{left}, the smallest integer in $[0, h'']$ with $(x_{left} + 1) \cdot \frac{N}{h''}$ in $J(a_i, b_i)$, and x_{right}, the largest integer in $[0, h'']$ with $x_{right} \cdot \frac{N}{h''}$ in $J(a_i, b_i)$.
3. Compute n_{c+f}, the number of lattice points in the chest and feet of $P(a_i, b_i)$, and n_l, an upper bound for the number of points in the legs.
4. Using Vallée's enumeration, select one integer in $x_{right} - x_{left}$ (there may be several) that corresponds to the lattice point (u, w) that is associated to x'. More specifically:
   - If (u, w) is the l^{th} point in the chest or feet, set $c = l$.
   - Otherwise, let s_v be Vallée's upper bound for the number of leg lattice points on quasi-horizontal lines with index at most v. Compute the index v of the line containing (u, w). Let n_v be the actual number of lattice points on the line with index v and let $n'_v = s_v - s_{v-1}$ be Vallée's upper-bound estimate. Suppose that x' is the k^{th} lattice point on the line. Pick an integer $c \in (n_{c+f} + s_{v-1} + n'_v \frac{k-1}{n_v}, n_{c+f} + s_{v-1} + n'_v \frac{k}{n_v}]$.
   - Pick an integer $c' \in ((x_{right} - x_{left}) \frac{c-1}{n_{c+f}+n_l}, (x_{right} - x_{left}) \frac{c}{n_{c+f}+n_l}]$. Set $x = x_{left} + c'$.

Although not mentioned explicitly in the algorithm description above, Vallée's quasi-enumeration, and the steps that use this quasi-enumeration, depend on the values of h and h' (which we assume to be public, and which could be most conveniently be set to 0 and $8N^{2/3}$). Shortly, we will calibrate h'' so that $x_{right} - x_{left}$ is larger than (but within a constant of) $n_{c+f} + n_l$. In computing

$\theta(x', d)$, d is used – either deterministically or as a source of random bits – to pick the values of c and c'. Given $\theta(x', d)$, one can recover the value of x' as follows:

Computing $\theta^{-1}(x)$:

1. Determine (a_i, b_i) for which $x \cdot \frac{N}{h''}$ is in $J(a_i, b_i)$.
2. Compute x_{left}, the smallest integer in $[0, h'']$ with $(x_{left}+1) \cdot \frac{N}{h''}$ in $J(a_i, b_i)$, and x_{right}, the largest integer in $[0, h'']$ with $x_{right} \cdot \frac{N}{h''}$ in $J(a_i, b_i)$.
3. Compute n_{c+f}, the number of lattice points in the chest and feet of $P(a_i, b_i)$, and n_l, an upper bound for the number of points in the legs.
4. Compute $c' = x - x_{left}$. From c' and $n_{c+f} + n_l$, compute the value of c. If $c \leq n_{c+f}$, let (u, w) be the c^{th} point in the chest or feet. Otherwise, compute the index v such that $c \in (n_{c+f} + s_{v-1}, n_{c+f} + s_v]$, as well as the value of k (defined as above), and let (u, w) be the k^{th} point on the quasi-horizontal line with index v.
5. Set $x' = \theta^{-1}(x) = \lfloor \frac{a_i N}{b_i} \rceil + u$.

Now, we calibrate h'' to be as small as possible while still allowing the property that at least one bit string in $[0, h'']$ is uniquely associated to each $B_{N,h,h'}$-element. We can ensure this property if, for every interval, $x_{right} - x_{left} \geq n_{c+f} + n_l$ – i.e., the number of bit strings associated to $J(a_i, b_i)$ is at least the number of points in $P(a_i, b_i)$.

Since $x_{left}\frac{N}{h''}$ and $(x_{right}+1)\frac{N}{h''}$ are separated by a distance greater than the width of $J(a_i, b_i)$, we get that $(x_{right} - x_{left} + 1)\frac{N}{h''} > \frac{h'-h}{4b_i}$, where the latter term is the half of the diameter of $I(a_i, b_i)$; thus, we get $x_{right} - x_{left} + 1 > \frac{h''(h'-h)}{4b_i N}$. To determine an h'' for which $\frac{h''(h'-h)}{4b_i N} \geq n_{c+f} + n_l$, we use an upper bound proven by Vallée [33]: $n_{c+f} + n_l \leq 4\frac{(h'-h)^2}{2b_i N} = \frac{2(h'-h)^2}{b_i N}$. Thus, if $h'' \geq 8(h' - h)$, then $x_{right} - x_{left} + 1 > n_{c+f} + n_l$. As long as the n_l estimate is an integer, this implies that $x_{right} - x_{left} \geq n_{c+f} + n_l$, as desired. So, we can set $h'' = 8(h' - h)$. For this value of h'', the θ mapping compresses $B_{N,h,h'}$-elements to within 3 bits of the theoretical minimum. The reader can verify that θ outputs an answer for every x' (i.e., $l_1 \geq 1$) and that θ^{-1} has exactly one possible output for each x (i.e., $l_3 = l_4 = 1$).

5.2 Mapping Short Strings to B-Elements (The π Quasi-Bijection)

Like θ^{-1}, the π quasi-bijection maps short strings to $B_{N,h,h'}$-elements. However, we would like π to map short strings (e.g., plaintext strings) into $B_{N,h,h'}$ *injectively* (e.g., to allow correct decryption); thus, the set of short strings is smaller than the set of $B_{N,h,h'}$-elements (rather than the reverse). For that reason, π uses Vallée's *lower* bounds (unlike θ). Since π is otherwise similar to θ^{-1}, we relegate a precise description of π to Appendix B.

In terms of performance, all steps of the θ and π quasi-bijections and their inverses are $O(\log^2 N)$, except (possibly) the determination of the Farey interval, which uses continued fractions. However, even the continued fraction step can be computed in $O(\log^2 N)$ time – e.g., using adaptations of techniques from [14].

6 Compressed Rabin-PDH Signing and Compressed Rabin-OAEP+ Encryption

In this section, we describe how to use the θ and π quasi-permutations to achieve a 33% reduction in the size of Rabin signatures and Rabin ciphertexts.

The signature case is easy to describe. Recall that, in Section 4.2, we described how to construct a Rabin-PDH signature s that satisfies either $s^2 \equiv \pm H_1(M)(\mathrm{mod} N)$ or $s^2 \equiv \pm 2 \cdot H_1(M)(\mathrm{mod} N)$ for $H_1 : \{0,1\}^* \rightarrow [h, h')$, where $h' - h \geq 8N^{2/3}$. For simplicity, let's assume that $s^2 \equiv H_1(M)(\mathrm{mod} N)$; the other cases can be handled similarly. In this case, we simply set the compressed Rabin-PDH signature to be $\theta_{N,h,h'}(s, d)$ – i.e., the θ quasi-permutation's compression of s for modulus N and parameters h and h'. To verify the compressed Rabin-PDH signature, the verifier simply recovers s from $\theta_{N,h,h'}(s, d)$, and then verifies s in the normal fashion. Note that anybody can create a compressed Rabin-PDH signature from a (non-compressed) Rabin-PDH signature, and vice versa, without needing trapdoor information – i.e., the compression algorithm is completely separate from the signing process.

The proof of security for compressed Rabin-PDH follows easily from the proof of security for (non-compressed) Rabin-PDH. Specifically, let $\mathcal{A}$ be a chosen-message attack adversary against Compressed Rabin-PDH, and let $\mathcal{B}$ be chosen-message attack adversary against Rabin-PDH that interacts both with a "challenger" and with $\mathcal{A}$. To respond to $\mathcal{A}$'s signature query on M, $\mathcal{B}$ queries the challenger regarding M, receives back Rabin-PDH signature x', and sends x to $\mathcal{A}$, where $x = \theta_{N,h,h'}(x', d)$. Eventually, $\mathcal{A}$ aborts or sends $\mathcal{B}$ a forgery x^* on a message M^* that it has never queried. $\mathcal{B}$ aborts or computes $x'^* = \theta^{-1}_{N,h,h'}(x^*)$ and sends x'^* to the challenger as its forgery.

The encryption case is more complicated, because the compression algorithm cannot be separated from the encryption process. Unfortunately, this fact – together with the fact the encryption scheme is not quite a one-way *permutation* as required by OAEP+, but rather a *quasi-bijection* – requires us redo the entire OAEP+ security proof, albeit with relatively minor modifications. At a high level, encryption and decryption proceed as follows:

Encryption:

1. Compute $x \in [1, h'']$, an encoding of M.
2. Compute $x' = \pi_{N,h,h'}(x, d) \in B_{N,h,h'} \cap [0, N/2)$.
3. Compute $y = x'^2 (\mathrm{mod} N)$.
4. Output $c = y - h$ as the ciphertext.

Decryption:

1. Recover y from c and h.
2. Compute each $x' \in B_{N,h,h'} \cap [0, N/2)$ such that $x'^2 \equiv y(\mathrm{mod} N)$.
3. For each x', compute the values of $x = \pi^{-1}_{N,h,h'}(x', d)$.
4. For each x, undo the message encoding, and confirm that the message M is encoded correctly.

5. If an x is encoded correctly, output the decryption; otherwise, indicate decryption failure.

For Vallée's parameters, for a given x', there are at most two values of x in Step 3 – i.e., $l_4 = 2$ – so the encoding of at most 4 values of x must be checked. As mentioned in section 5 and further discussed in Appendix B, our preferred parameters are $h' - h = 8N^{2/3}$ and $h'' \leq \frac{(h'-h)}{5}$.

Although we could use any of a variety of encoding schemes, we prove that Compressed Rabin-OAEP+ has a tight reduction to factoring. The OAEP+ encoding scheme uses three hash functions:

$$G : \{0,1\}^{k_0} \to \{0,1\}^m, H' : \{0,1\}^{m+k_0} \to \{0,1\}^{k_1}, \text{ and } H : \{0,1\}^{m+k_1} \to \{0,1\}^{k_0}$$

where m, k_0, k_1 are security parameters. The quantities 2^{-k_0} and 2^{-k_1} should be negligible. Let $n = m + k_0 + k_1 = \log h'' < \frac{2}{3} \log N + \frac{8}{5}$. To encode message $M \in \{0,1\}^m$, the sender:

1. Picks a random $r \in \{0,1\}^{k_0}$.
2. Sets $s \leftarrow (G(r) \oplus M) \| H'(r \| M)$ and $t \leftarrow H(s) \oplus r$.
3. Sets $x \leftarrow s \| t$, an n-bit integer.

In Step 4 of Decryption, the recipient decodes by parsing each candidate x into $s_i \| t_i$ for $s_i \in \{0,1\}^{m+k_1}$ and $t_i \in \{0,1\}^{k_0}$, and then parsing s_i into for $s'_i \| s''_i$ for $s'_i \in \{0,1\}^m$ and $s''_i \in \{0,1\}^{k_1}$. For each i, the recipient computes $r_i \leftarrow t_i \oplus H(s_i)$ and $M_i \leftarrow s'_i \oplus G(r_i)$, and tests whether $s''_i = H'(r_i \| M_i)$. If there is a unique i for which the condition is satisfied, the recipient outputs M_i as the correct plaintext; otherwise, it indicates a decryption failure. For technical reasons in the security proof, we require that $d = r$ – i.e., that the encrypter use r as the random bits in the computation of $\pi_{N,h,h'}(x, d)$ – and that the decrypter indicate a decryption failure if this is not done. For compressed Rabin-OAEP+, we prove the following theorem in Appendix C.

Theorem 6. *Let $\mathcal{A}$ be an IND-CCA2 adversary that breaks Compressed Rabin-OAEP+ in time t with advantage ϵ for modulus N. Then $\epsilon \leq \frac{l_2}{l_1}\epsilon' + (q_{H'} + q_D)/2^{k_1} + (q_D + 1)q_G/2^{k_0}$, where ϵ' is the success probability that a particular algorithm $\mathcal{B}$ can factor, $t' = O(t + q_G q_H T_f + (q_G + q_{H'} + q_H + q_D)\log N)$, and T_f is the complexity of encryption.*

7 Extensions

In the full version of the paper, we describe compressed signcryption, aggregate signature and ring signature schemes, in which we achieve a 33% bandwidth reduction in comparison to Rabin-variants of the schemes in [24], [23] and [29]. We also note that our compression algorithms can be applied to allow shorter identity-based secret and public keys for the Fiat-Shamir signature scheme and Cocks' identity-based encryption scheme.

References

1. T.M. Apostol, *Modular Functions and Dirichlet Series in Number Theory*, Springer-Verlag (1976).
2. K. Barr and K. Asanović, *Energy Aware Lossless Data Compression*, in Proc. of MobiSys 2003.
3. M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures – How to Sign with RSA and Rabin*, in Proc. of Eurocrypt 1996, LNCS 1070, pages 399–416. Springer-Verlag, 1996.
4. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption – How to Encrypt with RSA*, in Proc. of Eurocrypt 1994, LNCS 950, pages 92–111. Springer-Verlag, 1994.
5. D.J. Bernstein, *A Secure Public-Key Signature System with Extremely Fast Verification*, 2000. Available at `http://cr.yp.to/djb.html`.
6. D.J. Bernstein, *Proving Tight Security for Standard Rabin-Williams Signatures*, 2003. Available at `http://cr.yp.to/djb.html`.
7. D.J. Bernstein, *Reducing Lattice Bases to Find Small-Height Values of Univariate Polynomials*, 2003. Available at `http://cr.yp.to/djb.html`.
8. D. Bleichenbacher, *Compressed Rabin Signatures*, in Proc. of CT-RSA 2004.
9. D. Boneh, *Simplified OAEP for the RSA and Rabin Functions*, in Proc. of Crypto 2001, LNCS 2139, pages 275–291. Springer-Verlag, 2001.
10. D. Boneh, C. Gentry, B. Lynn, and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, in Proc. of Eurocrypt 2003, LNCS 2656, pages 416-432. Springer-Verlag, 2003.
11. D. Boneh, G. Durfee, and N. Howgrave-Graham, *Factoring $N = p^r q$ for Large r*, in Proc. of Crypto 1999, LNCS 1666, pages 326–337. Springer-Verlag, 1999.
12. D. Boneh and R. Venkatesan, *Breaking RSA May Not Be Equivalent to Factoring*, in Proc. of Eurocrypt 1998, LNCS 1233, pages 59–71. Springer-Verlag, 1998.
13. C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, in Proc. of Cryptography and Coding 2001, LNCS 2260, Springer (2001). Available at `http://www.cesg.gov.uk/technology/id-pkc/media/ciren.pdf`.
14. H. Cohen, *A Course in Computational Algebraic Number Theory*, 4th ed., Graduate Texts in Mathematics, Springer, 2000.
15. J.S. Coron, *On the Exact Security of Full Domain Hash*, in Proc. of Crypto 2000, LNCS 1880, pages 229-235. Springer-Verlag, 2000.
16. J.-S. Coron, *Security Proof for Partial-Domain Hash Signature Schemes*, In Proc. of Crypto 2002, LNCS 2442, pages 613–626. Springer-Verlag, 2002.
17. D. Coppersmith, *Finding a Small Root of a Univariate Modular Equation*, in Proc. of Eurocrypt 1996, LNCS 1070, pages 155-165. Springer-Verlag, 1996.
18. U. Feige, A. Fiat, A. Shamir, *Zero-Knowledge Proofs of Identity*, in Jour. of Cryptology (1), pp. 77–94 (1988).
19. A. Fiat, A. Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, in Proc. of Crypto 1986, LNCS 263, pp. 186-194. Springer (1986).
20. G.H. Hardy and E.M. Wright, *An Introduction to the Theory of Numbers*, Oxford Science Publications (5^{th} edition).
21. J. Jonsson, *A OAEP Variant with a Tight Security Proof*, 2003. Available at `http://www.math.kth.se/~jakobj/crypto.html`.
22. A.K. Lenstra and E.R. Verheul, *The XTR Public Key System*, In Proc. of Crypto 2000, LNCS 1880, pages 1–20. Springer-Verlag, 2000.

23. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham, *Sequential Aggregate Signatures from Trapdoor Homomorphic Permutations*, in Proc. of Eurocrypt 2004, LNCS 3027, pages 74–90. Springer-Verlag, 2004.
24. J. Malone-Lee and W. Mao, *Two Birds One Stone: Signcryption Using RSA*, 2002. Available at `http://www.hpl.hp.com/techreports/2002/HPL-2002-293.html`.
25. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
26. S. Micali, A. Shamir, *An Improvement of the Fiat-Shamir Identification and Signature Scheme*, in Proc. of Crypto 1988, LNCS 403, pp. 244–247. Springer-Verlag (1990).
27. H. Ong, C.P. Schnorr, *Fast Signature Generation with a Fiat Shamir - Like Scheme*, in Proc. of Eurocrypt 1990, LNCS 473, pp. 432–440. Springer-Verlag (1990).
28. M.O. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.
29. R.L. Rivest, A. Shamir, and Y. Tauman, *How to Leak a Secret*, in Proc. of Asiacrypt 2001, LNCS 2248, pages 552–565. Springer-Verlag, 2001.
30. K. Rubin and A. Silverberg, *Torus-based Cryptography*, in Proc. of Crypto 2003, LNCS 2729, pages 349–365. Springer-Verlag, 2003.
31. V. Shoup, *OAEP Reconsidered*, in Proc. of Crypto 2001, LNCS 2139, pages 239-259. Springer-Verlag, 2001.
32. A.K. Lenstra, A. Shamir, J. Tomlinson and E. Tromer, *Analysis of Bernstein's Factorization Circuit*, in Proc. of Asiacrypt 2002, LNCS 2501, pages 1–26. Springer-Verlag, 2002.
33. B. Vallée, *Provably Fast Integer Factoring with Quasi-Uniform Small Quadratic Residues*, In Proc. of STOC 1989, pages 98–106.
34. B. Vallée, *Generation of Elements with Small Modular Squares and Provably Fast Integer Factoring Algorithms*, Mathematics of Computation, vol. 56, no. 194, pages 823–849, 1991.

A Security Proof for Improved Rabin-PDH

To prove our scheme secure against existential forgery under chosen-message attacks, we construct the following game:

Setup: $\mathcal{B}$ gives $\mathcal{A}$ the public key N, retaining H_1 for use as a random oracle.

Hash Queries: $\mathcal{A}$ can make a query M_i to the H_1-oracle at any time. If $\mathcal{B}$ has received an identical query before, it responds as it did before. Otherwise, $\mathcal{B}$ responds by first generating a random value $c_i \in \{-2, -1, 1, 2\}$ with uniform distribution. It then generates a number $s_i \in B_{N,c_i h,c_i h'}$ with uniform distribution and sets $H_1(M_i) = s_i^2/c_i(\text{mod}N)$. It logs (M_i, s_i) into its H_1-list. (When $c_i = \pm 2$, there is a small complication – namely, s_i must be chosen s.t. not only $c_i h(\text{mod}n) \leq s_i^2(\text{mod}n) < c_i h'(\text{mod}n)$, but also $s_i^2(\text{mod}n) \in \{c_i h(\text{mod}n), \ldots, c_i(h'-1)(\text{mod}n)\}$. The simulator can accomplish this easily simply by discarding the sampled s_i's that don't satisfy the latter inequality (50% of them for $|c_i| = 2$).)

Signature Queries: $\mathcal{A}$ can make a query M_i to the H_1-oracle at any time. $\mathcal{B}$ responds by using M_i to recover s_i from its H_1-list; it then sends s_i.

Forgery: Eventually, the adversary either aborts or outputs a signature s on a message M for which it has not made a signature query.

One can easily confirm that $\mathcal{B}$'s H_1-query responses, as well as its signature responses, are indistinguishable from uniform; in fact, they are perfectly uniform.

Any forgery that $\mathcal{A}$ manages to generate for message M must satisfy $s^2 \equiv \pm H_1(M)(\mathrm{mod} N)$ or $s^2 \equiv \pm 2 \cdot H_1(M)(\mathrm{mod} N)$. If $\mathcal{A}$ made no H_1-query at M, then its probability of success is at most $1/h''$. If $\mathcal{A}$ did make an H_1-query at M, then $\mathcal{B}$ recovers the value s' associated to M from its H_1-list. With probability $\frac{1}{2}$, $\gcd(s - s', N)$ gives a nontrivial factor of N. Thus, $\epsilon' \geq \frac{1}{2}\epsilon(1 - \frac{1}{h''})$, and $t' = O(t + q_H \log^2 N)$.

B Details of the π Quasi-Bijection

Let $x \in [0, h'']$, where h'' is a parameter whose value will be calibrated later. The π quasi-permutations sends x to an element of $B_{N,h,h'}$, as follows.

Computing $\pi(x, d)$:

1. Compute $x \cdot \frac{N}{h''}$, and determine (a_i, b_i) for which the result is in $J(a_i, b_i)$.
2. Compute x_{left}, the smallest integer in $[0, h'']$ with $(x_{left} + 1) \cdot \frac{N}{h''}$ in $I(a_i, b_i)$, and x_{right}, the largest integer in $[0, h'']$ with $x_{right} \cdot \frac{N}{h''}$ in $I(a_i, b_i)$.
3. Compute n_{c+f}, the number of lattice points in the chest and feet of $P(a_i, b_i)$, and n_l, a lower bound for the number of points in the legs.
4. Using Vallée's enumeration, select one lattice point (u, w) (there may be several) that corresponds to $x - x_{left}$. More specifically:
   – Pick an integer in $c \in ((n_{c+f} + n_l)\frac{x - x_{left} - 1}{x_{right} - x_{left}}, (n_{c+f} + n_l)\frac{x - x_{left}}{x_{right} - x_{left}}]$
   – If $c \leq n_c + n_f$, pick the lattice point (u, w) that has enumeration c in the chest or feet.
   – Otherwise, let s_v be Vallée's lower-bound for the number of leg lattice points on quasi-horizontal lines with index at most v. Compute v such that $s_{v-1} < c - n_{c+f} \leq s_v$. Let n_v be the number of lattice points on the line with index v and let n'_v be Vallée's lower-bound estimate. Pick an integer $c' \in (n_v(\frac{c - n_{c+f} - s_{v-1} - 1}{n'_v}), n_v(\frac{c - n_{c+f} - s_{v-1}}{n'_v})]$, and set (u, w) to be the c'^{th} point in $P(a_i, b_i)$ on the line.
5. Set $x' = x_0 + u$, where $x_0 = \lfloor \frac{a_i N}{b_i} \rceil$. Output x'.

We omit the description of $\pi^{-1}(x')$, since it should be clear from the above. Now, we mention some of the properties of the π quasi-permutation.

Choosing the parameters such that $0 < x_{right} - x_{left} \leq n_{c+f} + n_l$ – i.e., such that the lower bound on the number of points in $P(a_i, b_i)$ is greater than the number of bit strings associated to $I(a_i, b_i)$ – ensures that l_1 is at least 1, since one can always find a value for c in the computation of π. Notice that $(x_{right} - x_{left} - 1)\frac{N}{h''} < \frac{h' - h}{2b_i}$, where the latter term is the diameter of $I(a_i, b_i)$. This implies that $x_{right} - x_{left} - 1 < \frac{h''(h' - h)}{2b_i N}$. Now, consider the parameters used by Vallée. Vallée considered the case $-h = h' = 4N^{2/3}$, so that $h' - h = 8N^{2/3}$.

For this value of $h' - h$, Vallée proved a lower bound of $n_{c+f} + n_l \geq \frac{(h'-h)^2}{10 b_i N}$ (see [33]). Thus, if $h'' \leq \frac{(h'-h)}{5}$, then $x_{right} - x_{left} - 1 < n_{c+f} + n_l$. As long as the n_l estimate is an integer, this implies that $x_{right} - x_{left} \leq n_{c+f} + n_l$, as desired. To ensure that $x_{right} - x_{left}$ is never zero, we want that $\frac{N}{h''} \leq \frac{N}{k^2} \Rightarrow h'' \geq k^2 = N^{2/3}/16 = \frac{(h'-h)}{128}$, where the latter is the diameter of the narrowest Farey interval. So, we can set h'' to be anything between $\frac{(h'-h)}{128}$ and $\frac{(h'-h)}{5}$; values closer to the latter involve less ciphertext expansion.

On the other hand, we would like l_2 and l_4 to be small positive constants. This ensures that picking x (and d) uniformly and outputting $\pi(x, d)$ is a *quasi-uniform drawing algorithm* for $B_{N,h,h'}$ (this helps get a tight security proof for the encryption scheme). The computation of $\pi^{-1}(x')$ outputs up to two values of x, exactly one for each Farey interval that contains x'; thus $l_4 = 2$. We use Vallée's upper bounds to bound l_2. Specifically, Vallée's computations allow $n_{c+f} + n_l$ to be upper bounded by $(1.004 + 0.125 + \frac{4-\sqrt{5}}{8})\frac{(h'-h)^2}{2b_i N} < \frac{.7(h'-h)^2}{b_i N}$, allowing us to upper bound the number of possible values of c by 4, for h''. Also, there are at most $\lceil \frac{7}{2} \rceil = 4$ (see Vallée's Leg Theorem) possible values of c', so l_2 is at most $4 \times 4 = 16$. Accordingly, for $h' - h = 8N^{2/3}$ and $h'' = \lfloor \frac{(h'-h)}{5} \rfloor$, one gets a $(1, 16, 1, 2)$ quasi-bijection.

C Security Proof for Compressed Rabin-OAEP+

Recall the standard definition of security against adaptive chosen-ciphertext attack. An algorithm $\mathcal{A}$ "breaks" the encryption scheme if, in the following game, it outputs the correct value of b in the final stage with more than negligible advantage:

Setup: The challenger generates a Rabin modulus N and hash functions G, H' and H, defined as above. It sends (N, G, H', H) to $\mathcal{A}$.
Phase 1: $\mathcal{A}$ requests the challenger to decrypt ciphertexts of $\mathcal{A}$'s choosing.
Challenge: $\mathcal{A}$ chooses two plaintexts M_0 and M_1 and sends them to the challenger. The challenger randomly chooses bit $b \in \{0, 1\}$, encrypts M_b, and sends the ciphertext c to $\mathcal{A}$.
Phase 2: $\mathcal{A}$ again requests the challenger to decrypt ciphertexts of $\mathcal{A}$'s choosing, other than the Challenge ciphertext.
Output: Finally, $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

We define $\mathcal{A}$'s advantage as: $\mathrm{Adv}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|$.

In the game above, algorithm $\mathcal{B}$ plays the part of the challenger, using its its control over the random oracles G, H' and H to respond to $\mathcal{A}$'s decryption queries. We say that the system is $(t, \epsilon, q_D, q_G, q_{H'}, q_H)$-secure if no attacker limited to time t, to q_D decryption queries, to q_G G-queries, to $q_{H'}$ H'-queries, and to q_H H-queries, has advantage more than ϵ. Now, we define aspects of the game more precisely.

Hash queries: $\mathcal{A}$ can query G, H' or H at any time. In responding to these queries, $\mathcal{B}$ maintains a G-list, H'-list and H-list logging queries and responses. If

$\mathcal{A}$ makes a query that is contained in one of $\mathcal{B}$'s lists, $\mathcal{B}$ responds the same way it did before. Otherwise, for G, it generates a random m-bit string with uniform distribution, sends this to $\mathcal{A}$ as its G-query response, and logs $\mathcal{A}$'s G-query and its response on its G-list. It responds similarly to H'-queries and H-queries. We use the convention that before $\mathcal{A}$ makes an H'-query on (r_i, M_i), it makes a G-query on r_i and an H-query on $s_i = (G(r_i) \oplus M_i) \| H'(r_i \| M_i)$.

Challenge: At some point, $\mathcal{A}$ produces two plaintexts $M_0, M_1 \in \{0,1\}^m$ on which it wishes to be challenged. $\mathcal{B}$ picks a random $b \in \{0,1\}$ and encrypts M_b in the usual way. Let c^* be the resulting ciphertext, and let s'^*,s''^*,t^*,r^*, and M^* denote the values corresponding to c^* that would be obtained through the decryption process.

Decryption queries and Probability Analysis: $\mathcal{A}$ can make decryption queries at any time, subject to the constraint that it cannot query the Challenge ciphertext in Phase 2. Our treatment of decryption queries closely tracks Shoup's analysis for trapdoor *permutations* encoded using OAEP+. Shoup's analysis consists of a sequence of games G_i for $0 \leq i \leq 5$, each game a slight modification of the previous one, where G_0 represents the attack on the encryption scheme, and G_5 is a certain attack in which an adversary obviously has no advantage. Shoup bounds $|\Pr[S_{i-1}] - \Pr[S_i]|$ for $1 \leq i \leq 5$, where $\Pr[S_i]$ is an adversary's probability of success in game G_i, thereby bounding an adversary's advantage in G_0. To reduce space, our proof draws heavily from Shoup's proof.

In game G_1, the decryption oracle decrypts ciphertext c_i as usual, recovering s'_i,s''_i,t_i,r_i, and M_i in the process. The decryption oracle is identical to G_0 (e.g., it can find modular square roots) except that the decryption oracle in G_1 rejects whenever r_i is not on its G-list. Let F_1 be the event that a ciphertext rejected in G_1 would not have been rejected in G_0. Consider a ciphertext $c \neq c^*$ submitted to the decryption oracle. If $r = r^*$ and $M = M^*$, then since there is only a single legitimate ciphertext generated from r^* and M^* (recall that we use r as the random bits in the π quasi-bijection), G_0 would also have rejected. Our analysis of the case of $r_i \neq r^*$ or $M_i \neq M^*$ is identical to Shoup's, leading to the conclusion that $|\Pr[S_0] - \Pr[S_1]| \leq q_D/2^{k_1}$.

In game G_2, the decryption oracle is identical to that of G_1, except it rejects when s_i is not on its H-list. Let F_2 be the event that a ciphertext rejected in G_2 would not have been rejected in G_1. For ciphertext $c_i \neq c^*$ with s_i not on the H-list, we consider two cases:
Case 1: $s_i = s^*$. Now, $s_i = s^*$ and $c_i \neq c^*$ implies $t_i \neq t^*$ (again because we made π deterministic given r). Shoup's remaining analysis of this case also works for our situation.
Case 2: $s_i \neq s^*$. Our analysis here is again identical.
Like Shoup, we obtain $|\Pr[S_1] - \Pr[S_2]| = \Pr[F_2] \leq q_{H'}/2^{k_1} + q_D q_G/2^{k_0}$.

In game G_3 the decryption oracle does not have access to a trapdoor, but instead maintains a ciphertext-list. After receiving an H'-query (r_i, M_i), it computes all possible values of $x'_i = \pi_{N,h,h'}(s_i \| t_i, r_i)$ and $c_i = x'^2_{i,j} - h (\mathrm{mod} N)$. It logs these ciphertexts in its ciphertext-list. Shoup's probability analysis applies to our case: $\Pr[S_2] = \Pr[S_3]$. His time-complexity analysis also applies: over the course of G_3, the decryption oracle's complexity is $O(\min(q_{H'}, q_H))T_f + (q_G + q_{H'} + q_H + q_D)\log N)$, where T_f is the complexity of the encryption function.

Game G_4, in which Shoup replaces the original random oracles with different but identically distribute variables, also works in our case. (See [31] for details of G_4.) Note the new encryption oracle in G_4 is identically distributed to the old one, even though "f" is not a permutation in our case, since Shoup's changes only affect f's input, not f itself. $\Pr[S_3] = \Pr[S_4]$.

Game G_5 is the same as G_3 (we skipped describing G_4) except that the encryption oracle chooses random strings $r^+ \in \{0,1\}^{k_0}$ and $g^+ \in \{0,1\}^m$, and it uses these values in the computation of the ciphertext, as described in [31]. Since g^+ is only used to mask M^*, $\Pr[S_5] = \frac{1}{2}$. Like Shoup, we also obtain in our case that $\Pr[S_4] - \Pr[S_5] \leq \Pr[F_5]$, where F_5 is the event that $\mathcal{A}$ queries G at r^*. However, our proofs finally diverge significantly at this point. Shoup describes an auxiliary game G_5' in which the encryption oracle is modified again to simply output a random number c^+ in the ciphertext space (in our case, $B_{N,h,h'} \cap [0, N/2)$), and then he uses the fact that, for a permutation, c^+ comes from a distribution identical to c^*. We cannot do this, since the π quasi-bijection chooses from $B_{N,h,h'} \cap [0, N/2)$ – and thus from the ciphertext space – only quasi-uniformly.

Instead, we define our c^+ as $f(w)$ for $w \in \{0,1\}^n$ chosen randomly with uniform distribution, and (as always) r^* and s^* are defined with respect to this ciphertext. Then, for reasons analogous to those used by Shoup, if we define F_5' to be the event that $\mathcal{A}$ queries G at r^* in game G_5', we have $\Pr[F_5] = \Pr[F_5']$. Letting F_5'' be the event that $\mathcal{A}$ queries H at s^* in game G_5', we have that $\Pr[F_5'] = \Pr[F_5' \wedge F_5''] + \Pr[F_5' \wedge \neg F_5'']$.

Now, we claim that, if π is an (l_1, l_2, l_3, l_4) quasi-bijection, then $\Pr[F_5' \wedge F_5''] \leq \frac{l_2}{l_1}\mathrm{Adv}(\mathcal{B})$. For brevity, denote the probability $\Pr[F_5' \wedge F_5'' | w']$ – i.e., the probability F_5' and F_5'' occur given the value $w' = \pi_{N,h,h'}(w, r)$ for w as chosen above – by $P_{w'}$, where w' will be treated as a random variable. Notice that, for any w', there exists a v' such that $w'^2 \equiv v'^2 (\mathrm{mod} N)$ and $\gcd(w', v', N)$ is a nontrivial factor of N; in fact, we can "pair off" the numbers in $B_{N,h,h'}$, so that each w' corresponds to exactly one v'. Suppose that $r'' \in \{0,1\}^{k_0}$ and $s'' \in \{0,1\}^{m+k_1}$ correspond to v'. If $\mathcal{A}$ queries $r'' \in \{0,1\}^{k_0}$ and $s'' \in \{0,1\}^{m+k_1}$ (which occurs with probability $P_{v'}$), then $\mathcal{B}$ can use $w' = f(w)$ to find a nontrivial factor of N by taking every pair r_i, s_i queried by $\mathcal{A}$, deriving the corresponding t'', computing $x'' = \pi_{N,h,h'}(s_i \| t_i)$, and checking whether $\gcd(x'', w', N)$ is a nontrivial factor.

Overall, we have that $\Pr[F_5' \wedge F_5''] = \sum_{w'} \Pr[F_5' \wedge F_5'' | w'] \cdot \Pr[w']$. This probabilility is less than $\frac{l_2}{l_1} \sum_{w'} \Pr[F_5' \wedge F_5'' | w'] \cdot \Pr[v']$ by quasi-uniformity, where each w' is paired off with a v' that gives a nontrivial factor. However, the probability $\sum_{w'} \Pr[F_5' \wedge F_5'' | w'] \cdot \Pr[v']$ is less than $\mathcal{B}$'s probability of success, which proves the claim.

For the same reason as in [31], $\Pr[F_5' \wedge \neg F_5''] \leq q_G / 2^{k_0}$. Thus, we get $\Pr[F_5'] \leq \frac{l_2}{l_1}\mathrm{Adv}(\mathcal{B}) + q_G / 2^{k_0}$. Collecting all of the results, we get the time and complexity stated in the theorem.