Fiat-Shamir With Aborts: Applications to Lattice and Factoring-Based Signatures

Vadim Lyubashevsky [*]

Department of Computer Science
Tel-Aviv University
Tel Aviv 69978, Israel
vlyubash@cs.ucsd.edu

Abstract. We demonstrate how the framework that is used for creating efficient number-theoretic ID and signature schemes can be transferred into the setting of lattices. This results in constructions of the most efficient to-date identification and signature schemes with security based on the worst-case hardness of problems in ideal lattices. In particular, our ID scheme has communication complexity of around $65,000$ bits and the length of the signatures produced by our signature scheme is about $50,000$ bits. All prior lattice-based identification schemes required on the order of millions of bits to be transferred, while all previous lattice-based signature schemes were either stateful, too inefficient, or produced signatures whose lengths were also on the order of millions of bits. The security of our identification scheme is based on the hardness of finding the approximate shortest vector to within a factor of $\tilde{O}(n^2)$ in the standard model, while the security of the signature scheme is based on the same assumption in the random oracle model. Our protocols are very efficient, with all operations requiring $\tilde{O}(n)$ time.

We also show that the technique for constructing our lattice-based schemes can be used to improve certain number-theoretic schemes. In particular, we are able to shorten the length of the signatures that are produced by Girault's factoring-based digital signature scheme ([10, 11, 31]).

1 Introduction

The appeal of building cryptographic primitives based on the hardness of lattice problems began with the seminal work of Ajtai who showed that one-way functions could be built with security based on the worst-case hardness of certain lattice problems [2]. Unfortunately, cryptographic primitives that were built with this very strong security property were extremely inefficient for practical applications. For example, evaluating one-way and collision-resistant hash functions required $\tilde{O}(n^2)$ time and space [2, 24], and in public-key cryptosystems, the keys

[*] Supported by the Israel Science Foundation and by a European Research Council (ERC) Starting Grant. Some of the work in this paper was performed while the author was a student at the University of California, San Diego.

were on the order of megabytes [3, 33, 34, 28] (also see [25] for concrete parameter proposals for the scheme in [34]). Therefore some new ideas were required in order to make provably-secure lattice-based primitives a realistic alternative to ones based on number-theory.

A promising approach for improving efficiency is to use lattices that possess extra algebraic structure, and it is precisely this extra structure that makes the NTRU cryptosystem [14] (which unfortunately does not have a proof of security) very efficient in practice. A step in the direction of building *provably-secure* lattice-based primitives was taken by Micciancio [23], who showed that one could build *efficient* ($\tilde{O}(n)$ evaluation time) one-way functions with security based on the worst-case instances of problems pertaining to cyclic lattices (cyclic lattices are lattices that correspond to ideals in the ring $\mathbb{Z}[\mathbf{x}]/\langle\mathbf{x}^n - 1\rangle$)). This result was later extended to give constructions of collision-resistant hash functions by either restricting the domain [29] or changing the ring [18] in Micciancio's scheme. These works then led to constructions and implementations of collision-resistant hash functions [20] with security based on worst-case problems in lattices corresponding to ideals in $\mathbb{Z}[\mathbf{x}]/\langle\mathbf{x}^n + 1\rangle$ whose performance was comparable to the performance of ad-hoc hash functions that are currently in use today. And because there is a very close connection between collision-resistant hash functions and more sophisticated primitives such as ID schemes and digital signatures, it was very natural to ask whether these primitives also had efficient lattice-based constructions. There has been some recent work in this direction, which we will now describe.

Lyubashevsky and Micciancio constructed a one-time signature in which signing and verification can be performed in time $\tilde{O}(n)$ [19]. Using standard techniques, the one-time signature can be transformed into a full-fledged signature scheme using a signature-tree [21, 22] with only an additional work factor of $O(\log n)$. While this combination results in a very theoretically-appealing scheme where all the operations take time $\tilde{O}(n)$, it does require the use of a tree, which is a somewhat unwanted feature in practice. Another signature scheme was proposed by Gentry et al. in [9]. Their signature scheme follows the hash-and-sign paradigm, and when instantiated with algebraic lattices [37], verification takes time $\tilde{O}(n)$, but $\tilde{O}(n^4)$ time is needed to do the signing (it is plausible that the signing time could be reduced to $\tilde{O}(n^2)$ with a more careful analysis).

A different way of constructing digital signature schemes is to first construct an identification scheme of a certain form and then convert it to a signature scheme using the Fiat-Shamir transform [7, 32, 1]. The identification schemes of Micciancio and Vadhan [26], Lyubashevsky [17], and Kawachi et al. [15] can all be instantiated such that the secret and public keys are of size $\tilde{O}(n)$, and the entire interaction takes $\tilde{O}(n)$ time as well. While these constructions seem essentially optimal, they contain a common inefficiency. The ID schemes all have the form of standard commit-challenge-response protocols (see Figure 1, for an example of one where Y is the commitment, c is the challenge, and z is the response), and the inefficiency lies in the fact that for each challenge bit, the response consists of $\tilde{O}(n)$ bits. Since the security of the protocol is directly connected to the number of challenge bits sent by the verifier, it means that for every bit of security, $\tilde{O}(n)$ bits need to be transmitted. Theoretically, this does not cause a problem because one only needs $\omega(\log n)$ bits of security in order for the protocol to be considered secure against polynomial-time adversaries, and then the total running time of the above protocols is still $\tilde{O}(n)$. But in practice, this is a rather unsatisfactory solution because one wants some concrete security guarantee, say 80 bits, and then the communication complexity of the ID scheme will be about 80 times larger (the size of the signature in the derived signature scheme would be 160 times larger) than possibly necessary. This is in sharp contrast to number-theoretic ID schemes where the response of the prover is longer than the challenge by only a small factor.

What allows number-theoretic ID schemes like Schnorr [35], GQ [13], Girault [10], Okamoto [27], etc. to be so "compact" is that the challenge string in these protocols is not treated as a sequence of independent 0's and 1's, but instead the entire string is interpreted as an integer from a certain domain. This can be done because there is a lot of underlying algebraic structure upon which these schemes are built. On the other hand, lattices do not seem to have as much algebraic underpinning, and so the schemes based on them are very combinatorial in nature which is why the challenge strings are treated simply as a sequence of independent challenges much like in generic zero-knowledge proofs for NP. The main accomplishment of the current work is to show how to exploit the limited algebraic structure of ideal lattices in order to use the challenge bits collectively rather than individually, which ends up greatly improving the practical efficiency of lattice-based identification and signature schemes.

1.1 Contributions and Comparisons

Lattice-based constructions. We construct a lattice-based ID scheme in which the challenge string is treated as a polynomial in a certain ring, and one correct response to it from the prover is enough for authentication. The caveat is that some constant fraction of the time, the prover cannot respond to the challenge from the verifier and must abort the protocol. The result of this is that the "commit" and "challenge" steps of the ID scheme now must be repeated several times to ensure that a valid prover is accepted with some decent probability. But using standard techniques, one can significantly shorten the length of the "commit" part of the protocol, and because of the structure of our scheme, the challenge can always be the same. Therefore the number of transmitted bits is dominated by the length of the single "response".

Even more optimizations are possible when converting the ID scheme into a signature scheme using the Fiat-Shamir transform. In the resulting signature scheme, there is of course no longer any interaction until the signer outputs the signature. And therefore there is no need for the signer to output the attempts in which he failed to sign (which correspond to the times he couldn't answer the challenge in the ID scheme). So while the failures do cost time, the length of the final signature is as short as it would have been if the signer only attempted to sign once and succeeded. And because the probability of failure is a small constant ($\approx 2/3$), we only expect to repeat the signature protocol 3 times before succeeding.

All operations in our scheme take time $\tilde{O}(n)$ and we prove that the ID and signature schemes are secure based on the worst-case hardness of approximating the shortest vector to within a factor of $\tilde{O}(n^2)$ in lattices corresponding to ideals in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ (the security of the signature scheme is in the random oracle model). Compared to previous works, our asymptotic hardness assumption is the same as that in [19, 17] (although the scheme of [19] is secure in the standard model), but is worse than that in [26, 9, 37] (where the factor is $\tilde{O}(n^{1.5})$) and in [15] (where the factor is $\tilde{O}(n)$).

Based on the work of Gama and Nguyen [8] who worked out the effectiveness of current state-of-the-art lattice reduction algorithms, we present some concrete parameters with which our schemes can be instantiated. On the low end, the outputted signatures are about 50000 bits in length (the ID scheme requires about 65000 bits to be transmitted). While the scheme of [15] has better asymptotic security, the response to each challenge bit seems to require at least 10000 bits. So if we would like the challenge to be 160 bits for security purposes, the response (and therefore the signature size) will be over a million bits. The signature schemes of [26] and [17] look like they would have their signatures be about 160 times longer than ours (the ID schemes would require communications that are about 80 times longer), again because the responses are done separately to every challenge bit. So even though our ID and signature schemes have worse asymptotic security, their structure makes them much more practically efficient.

At this point it is not possible to give an accurate comparison of our signature scheme to the hash-and-sign signature schemes [9, 37] because no concrete parameters were given for those schemes. But independent of the signature sizes, our scheme will still have the advantage in that signing can be done in time $\tilde{O}(n)$ rather than $\tilde{O}(n^4)$.

The signature length of the one-time signature in [19] may actually be a little shorter than in our scheme, but this advantage is lost when the one-time signature gets converted to a general stateless signature scheme. If a signature tree is used in the conversion, then the signature length may go up by a factor of the tree depth, which would make it much less efficient. On the other hand, one could build a hash tree using any collision-resistant hash function, and then the signatures would only increase by the product of the tree depth and the hash function output. If the scheme is to be completely stateless and support about 2^{60} signatures, and we use SHA-256 as the hash function, then the length of the one-time signature in [19] would increase by about $15,000$ bits, which would make it somewhat longer than our signature. The similarity between the signature sizes of our scheme and the scheme in [19] is no coincidence, and we further discuss the relationship between the two schemes in Section 1.2.

Factoring-based signatures. We show that the ideas used to construct our lattice-based digital signature can also be used for shortening the length of some number-theoretic schemes. The signature scheme originally proposed by Girault

[10], and analyzed in [11, 31] is a scheme whose security, in the random oracle model, is based on the hardness of factorization. What is particularly attractive about it is that if the signer can do some pre-computing before receiving the message, then signing can be done with just one random oracle query, one multiplication, and one addition over the integers (no modular reduction is required). We show how to reduce the length of the signature in an instantiation of the scheme due to Pointcheval [31] from 488 bits to 425.

1.2 Techniques

There is a pattern that emerges when looking at constructions of certain ID and signature schemes based on the hardness of factoring and discrete log. The informal chain of reductions from the hard problem to the signature scheme looks as follows:

$$\text{Hard Problem} \leq \text{CRHF} \leq \text{One-time signature} \leq \text{ID scheme} \leq \text{Signature}$$

For example, finding collisions in the hash function $h(x) = g^x \bmod N$ implies being able to factor N. This can be converted into a one-time signature with the secret key being some pair of integers (x, y), public keys being $h(x), h(y)$, and the signature of a message c being $xc + y$. The one-time signature can then be converted into an ID scheme by simply picking a new y every time (Figure 1) and c now being a challenge chosen by the verifier. The ID scheme can then be converted to a signature scheme by using the Fiat-Shamir transform which replaces the verifier with a random oracle (Figure 5) [10, 11, 31]. The same idea can be used with the hash function $h(x_1, x_2) = (g_1^{x_1} g_2^{x_2} \bmod p)$, in which finding collisions implies solving the discrete log problem. The ID and signature schemes resulting from that hash function are due to Okamoto [27].

It turns out that a somewhat similar approach can be used to build lattice-based primitives as well. The works of [29, 18], showed a reduction from the *worst-case* problem of finding short vectors in algebraic lattices to finding collisions in hash functions. The work of [19] can be viewed as a transformation of the hash function to a one-time signature, and this current work can then be seen as the continuation of this chain of reductions where the one-time signature of [19] is converted into an ID scheme and then into a signature scheme.

But what prevents the techniques used in number-theoretic schemes to be directly extended to lattice-based ones, is that lattices allow for much less algebraic structure. For example, the domains in number-theoretic hash functions are rings, while in lattice-based ones, the domain is just a subset of a ring (in particular, those elements in the ring that have small Euclidean norm) that is neither closed under addition nor multiplication. This is very related to the fact that the factoring and discrete log problems can be reduced to finding an element in the kernel of some homomorphic function, while finding short vectors in lattices reduces to the problem of finding *small* elements in the kernel of a homomorphism. This difference is what seems to give lattice problems resistance against polynomial-time quantum algorithms that solve factoring and discrete log [36], but at the same time it also hinders constructions of lattice-based primitives.

Secret key: $s \overset{\$}{\leftarrow} D_s$
Public key: N, g, and $S \leftarrow g^s \bmod N$

$$\begin{array}{lcl}
\underline{\text{Prover}} & & \underline{\text{Verifier}}\\
y \overset{\$}{\leftarrow} D_y,\ Y \leftarrow g^y \bmod N & \xrightarrow{\quad Y \quad} & \\
 & \xleftarrow{\quad \mathbf{c} \quad} & c \overset{\$}{\leftarrow} D_c\\
z \leftarrow sc + y & & \\
[\text{if } z \notin G \text{ then } z \leftarrow\perp\ ] & \xrightarrow{\quad z \quad} & \\
 & & \text{Accept iff } g^z \equiv YS^c(\bmod N)
\end{array}$$

Fig. 1. Factoring-Based Identification Schemes.
The parameters for this scheme are in Figure 6. The line in [ ] is only performed in the aborting version of the scheme.

In overcoming this limitation, the one-time signature of [19] had to leak parts of its secret key. While it wasn't a problem in that setting because the secret key is only used once, in ID schemes the same secret key is used over and over, and so leaking a part of the secret key every time would result in complete insecurity. In this paper, we solve this difficulty by using an aborting technique that was introduced in [17]. The idea behind aborting is that the prover can elect to abort the protocol in order to protect some information about his secret key (mainly, the protocol needs to remain witness-indistinguishable). In this work, we are able to relax the conditions that were needed for witness-indistinguishability in [17], and this allows us to construct much more efficient lattice-based protocols as well as extend the technique to other contexts, such as the factoring-based scheme described in Section 1.1. We essentially show that all that is needed for the aborting technique to be applicable is a collision-resistant homomorphic hash function that has small elements in its kernel. We believe that this technique can find further applications.

1.3 Intuition for Aborting

Understanding where aborting might be useful is best accomplished with an example. Consider the protocol in Figure 1 (for this discussion, it is not necessary to understand why the protocol works), which has the form of a typical 3-round commit-challenge-response ID scheme. The secret key is some s and the public key is $h(s)$ where h is a function that happens to be $h(s) = g^s \bmod N$ in our example. In the first step of the protocol, the prover picks a parameter y, and sends $h(y)$, to the verifier. The verifier picks a random "challenge" c, and sends it to the prover. The third step of the protocol consists of a response of the prover to the challenge. This response must somehow use the secret key, and in our example, the secret key s is multiplied by c and then added to y. Notice that sending sc without adding it to y would completely reveal s, and so the job of y is to somehow mask the exact value of sc. If the operation sc takes place in some finite group, then a natural idea for masking would be to pick y uniformly at random from that group. The intuition is that if nothing about y is known, then the value $y + sc$ is also completely random (of course, something is known about y when the prover sends $h(y)$ to the verifier, but we gloss over that here). And this is exactly what is done in well-known ID schemes such as Schnorr [35], GQ [13], Okamoto [27], etc..

But sometimes it is infeasible to pick y uniformly at random from the group. In Girault's ID-scheme [10, 11, 31] (Figure 1), the multiplication sc is performed over the integers, which is an infinite group. A way to do masking in this scheme is to pick a y in a range that is much larger than the range of sc. So for example, if $0 \leq sc \leq R$, then one could pick a random y from the range $[0, ..., 2^{64}R]$. Then, with very high probability, the value of $sc + y$ will be in $[R, ..., 2^{64}R]$, in which case it will be impossible to determine anything about sc if nothing is known about y.

In constructing our lattice-based ID scheme, the same difficulty is encountered as in Girault's scheme, except we do not have the luxury of picking y (or something analogous to y in the lattice-based scheme) from such a large range because doing so would require us to make a much stronger complexity assumption which would significantly decrease the efficiency of the protocol (we would have to assume that it is hard to find a super-polynomial approximation of the shortest vector instead of just an $\tilde{O}(n^2)$ approximation). Our solution is to instead pick y from a much smaller set, something analogous to $[0, \ldots, 2R]$, but only reveal $sc + y$ if it falls into the range $[R, \ldots, 2R]$. If the range is picked carefully and the function h is a homomorphism that has "small" elements in its kernel, then one can show that if the prover only reveals values in this range and aborts otherwise, the protocol will be perfectly witness-indistinguishable. The witness-indistinguishability is then used to prove security of the protocol by showing that a forger can be used for extracting collisions in h.

The same technique can also be applied to Girault's scheme. Notice that if we pick y uniformly at random from the range $[0, ..., 2R]$ instead of from $[0, ..., 2^{64}R]$, the length of $sc + y$ will be 63 bits shorter. We point out that our aborting factoring-based ID scheme in Figure 1 which uses this idea is actually worse than the corresponding non-aborting one because the savings gained by shortening $sc + y$ are lost in case the prover has to abort and the ID protocol has to be repeated. But the advantage of aborting does show itself when the ID protocol is converted into a signature scheme using the Fiat-Shamir paradigm (Figure 5). In a signature scheme, there is no interaction, and therefore there is no need for the signer to ever include the aborted signing attempts into the final signature. So if the signer needs to abort, he simply reruns the protocol until he gets a signature in the correct range. The end result is that the eventual signature is shorter than it would have been in schemes such as [10, 11, 31] where the signer does not have the option to abort.

2 Preliminaries

2.1 Notation

We will denote vectors by bold letters. For convenience, vectors of vectors will be denoted by a bold letter with a hat. For example, if $\mathbf{a}_1, \mathbf{a}_2$ are elements of

$\mathbb{Z}^n$, then we can write $\widehat{\mathbf{a}} = (\mathbf{a}_1, \mathbf{a}_2)$. The ℓ_∞ norm of $\mathbf{a}$ is written as $\|\mathbf{a}\|_\infty$, and $\|\widehat{\mathbf{a}}\|_\infty$ for $\widehat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ is defined as $\max_i(\|\mathbf{a_i}\|_\infty)$. If S is a set, then $a \overset{\$}{\leftarrow} S$ means that a is chosen uniformly at random from S. All logarithms are assumed to be base 2.

2.2 Lattices and Algebra

An integer lattice Λ is a subgroup of $\mathbb{Z}^n$. The approximate Shortest Vector Problem ($\mathrm{SVP}_\gamma(\Lambda)$) asks to find a vector $\mathbf{v}$ in Λ such that $\|\mathbf{v}\|_\infty$ is no more than γ times larger than the vector in Λ with the smallest ℓ_∞ norm. In this work, we will be interested in lattices that exhibit an additional algebraic property – in particular, they correspond to ideals in the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$. We will say that a lattice Λ is an $(\mathbf{x}^n + 1)$-cyclic lattice if for every vector $(v_0, \ldots, v_{n-2}, v_{n-1}) \in \Lambda$, the vector $(-v_{n-1}, v_0, \ldots, v_{n-2})$ is also in Λ. If we look at the vectors as polynomials (i.e. $(v_0, \ldots, v_{n-2}, v_{n-1})$ as $v_0 + \ldots + v_{n-2}\mathbf{x}^{n-2} + v_{n-1}\mathbf{x}^{n-1}$), then an $(\mathbf{x}^n + 1)$-cyclic lattice is an ideal in $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ because in this ring,

$$(v_0 + \ldots + v_{n-2}\mathbf{x}^{n-2} + v_{n-1}\mathbf{x}^{n-1}) \cdot \mathbf{x} = -v_{n-1} + v_0\mathbf{x} + \ldots + v_{n-2}\mathbf{x}^{n-1}.$$

The ring that will be most important to us throughout the paper is the ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ where p is some odd positive integer. The elements in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ will be represented by polynomials of degree $n-1$ having coefficients in the range $\left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$. Throughout the paper, we will treat polynomials in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ and vectors in $\mathbb{Z}^n$ as the same data type. So when, for example, we talk of multiplying two vectors, we actually mean converting the vectors to polynomials and then multiplying the polynomials in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$. Similarly, the norm[1] of a polynomial is just the norm of the corresponding vector. It's not hard to see that for polynomials $\mathbf{v}, \mathbf{w} \in \mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$, the following relation holds:

$$\|\mathbf{v}\mathbf{w}\|_\infty \leq \|\mathbf{v}\|_\infty \|\mathbf{w}\|_1 \leq n\|\mathbf{v}\|_\infty \|\mathbf{w}\|_\infty$$

$(\mathbf{x}^n+1)$-cyclic lattices are a particular class of lattices that received attention because one can construct efficient and provably secure cryptographic primitives based on the hardness of finding approximate short vectors in these lattices [18, 29, 19, 20]. The main reason for this efficiency is that the multiplication of two polynomials in $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ can be done in time $\tilde{O}(n)$ using the Fast Fourier Transform. While the results in this paper can be applied to lattices that correspond to ideals in other rings, it would only unnecessarily complicate matters because the ring $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n+1\rangle$ seems to be the most useful theoretically and in practice.

While a lot is known about the complexity of SVP in general lattices, very little is known about this problem when restricted to ideal lattices. Nevertheless, the problem is related to some problems in algebraic number theory (see [18,

[1] This is a slight abuse of the word *norm*. Because of the reduction modulo p, it's not true that for any integer α we have $\|\alpha\mathbf{a}\|_\infty = |\alpha|\|\mathbf{a}\|_\infty$, but it still holds true that $\|\mathbf{a} + \mathbf{b}\|_\infty \leq \|\mathbf{a}\|_\infty + \|\mathbf{b}\|_\infty$ and $\|\alpha\mathbf{a}\|_\infty \leq |\alpha|\|\mathbf{a}\|_\infty$

30]) that do not have any efficient solution. And it seems that the currently best lattice algorithms are unable to take advantage of the extra structure provided by ideal lattices. Therefore, it still seems that solving SVP_γ takes time $2^{O(n)}$ when $\gamma = n^{O(1)}$ [16, 4].

2.3 Lattice-Based Collision-Resistant Hash Function

Let R be the ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n+1\rangle$. We define the following family of hash functions:

Definition 1. *For any integer m and $D \subseteq R$, the function family $\mathcal{H}(R, D, m)$ mapping D^m to R is defined as*

$$\mathcal{H}(R, D, m) = \{h_{\widehat{\mathbf{a}}} : \widehat{\mathbf{a}} \in R^m\}, \textit{where for any } \widehat{\mathbf{z}} \in D^m,\ h_{\widehat{\mathbf{a}}}(\widehat{\mathbf{z}}) = \widehat{\mathbf{a}} \cdot \widehat{\mathbf{z}}$$

That is, if $\widehat{\mathbf{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ and $\widehat{\mathbf{z}} = (\mathbf{z}_1, \ldots, \mathbf{z}_m)$, then $h_{\widehat{\mathbf{a}}}(\widehat{\mathbf{z}}) = \mathbf{a}_1\mathbf{z}_1 + \ldots + \mathbf{a}_m\mathbf{z}_m$ where all the operations are performed in the ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$. It's not hard to see that the hash functions in $\mathcal{H}(R, D, m)$ satisfy the following two properties for any $\widehat{\mathbf{y}}, \widehat{\mathbf{z}} \in R^m$ and $\mathbf{c} \in R$:

$$h(\widehat{\mathbf{y}} + \widehat{\mathbf{z}}) = h(\widehat{\mathbf{y}}) + h(\widehat{\mathbf{z}}) \tag{1}$$

$$h(\widehat{\mathbf{y}}\mathbf{c}) = h(\widehat{\mathbf{y}})\mathbf{c} \tag{2}$$

The collision problem $\mathrm{Col}(h, D)$ is defined as follows:

Definition 2. *Given an element $h \in \mathcal{H}(R, D, m)$, the collision problem Col(h, D), where $D \subseteq R$, asks to find two distinct elements $\widehat{\mathbf{z}}, \widehat{\mathbf{z}}' \in D$ such that $h(\widehat{\mathbf{z}}) = h(\widehat{\mathbf{z}}')$.*

In [18], it was shown that when D is some restricted domain, solving the $\mathrm{Col}(h, D)$ problem for random $h \in \mathcal{H}(R, D, m)$ is as hard as solving SVP_γ for *any* $(\mathbf{x}^n+1)$-cyclic lattice.

Theorem 1. *Let $R = \mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$ be a ring where n is any power of 2, and define $D = \{\mathbf{y} \in R : \|\mathbf{y}\|_\infty \leq d\}$ for some integer d. Let $\mathcal{H}(R, D, m)$ be a hash function family as in Definition 1 such that $m > \frac{\log p}{\log 2d}$ and $p \geq 4dmn^{1.5} \log n$. If there is a polynomial-time algorithm that solves the Col(h, D) problem for a random $h \in \mathcal{H}(R, D, m)$ with some non-negligible probability, then there is a polynomial-time algorithm that can solve $\mathrm{SVP}_\gamma(\mathit{\Lambda})$ for every $(\mathbf{x}^n + 1)$-cyclic lattice $\mathit{\Lambda}$, where $\gamma = 16dmn \log^2 n$.*

2.4 Cryptographic Definitions

Digital Signatures. We recall the definitions of signature schemes and what it means for a signature scheme to be secure.

Definition 3. *A* signature scheme *consists of a triplet of polynomial-time (possibly probabilistic) algorithms (G, S, V) such that for every pair of outputs (s, v) of $G(1^n)$ and any n-bit message m,*

$$Pr[V(v, m, S(s, m)) = 1] = 1$$

where the probability is taken over the randomness of algorithms S and V.

In the above definition, G is called the key-generation algorithm, S is the signing algorithm, V is the verification algorithm, and s and v are, respectively, the signing and verification keys.

A signature scheme is said to be secure if there is only a negligible probability that any forger, after seeing signatures of messages of his choosing, can sign a message whose signature he has not already seen [12].

Definition 4. *A signature scheme (G, S, V) is said to be secure if for every polynomial-time (possibly randomized) forger $\mathcal{F}$, the probability that after seeing the public key and $\{(\mu_1, S(s, \mu_1)), \ldots, (\mu_q, S(s, \mu_q))\}$ for any q messages μ_i of its choosing (where q is polynomial in n), $\mathcal{F}$ can produce $(\mu \neq \mu_i, \sigma)$ such that $V(v, \mu, \sigma) = 1$, is negligibly small. The probability is taken over the randomness of G, S, V, and $\mathcal{F}$.*

In the standard security definition of a signature scheme, the forger should not be able to produce a signature of a new message. A stronger notion of security, called *strong unforgeability* requires that in addition to the above, a forger shouldn't even be able to come up with a different signature for a message whose signature he has already seen. The schemes presented in this paper satisfy this stronger notion of unforgeability.

Identification Schemes. An identification scheme consists of a key-generation algorithm and a description of an interactive protocol between a prover, possessing the secret key, and verifier possessing the corresponding public key. In general, it is required that the verifier accepts the interaction with a prover who behaves honestly with probability one, but this definition can be relaxed so that sometimes an honest prover is not accepted with some small probability.

The standard active attack model against identification schemes proceeds in two phases [5]. In the first phase, the adversary interacts with the prover in an effort to obtain some information. In the second stage, the adversary plays the role of the prover and tries to make a verifier accept the interaction. We remark that in the second stage, the adversary no longer has access to the honest prover. The adversary succeeds if he is able to make an honest verifier accept with some non-negligible probability.

Witness-Indistinguishability. We will only define the concept of witness-indistinguishability in a way that pertains to our application and we refer the reader to [6] for the more general definition. For convenience, we will use the notation from the identification protocol in Figure 1. An identification scheme is said to be perfectly witness-indistinguishable if for any public key S, and any two valid secret keys s, s' (i.e. $s, s' \in D_s$ and $g^s \bmod N = g^{s'} \bmod N = S$), the view of any (possibly malicious) verifier in the interaction where the prover uses s has the exact same distribution as the view where the prover uses s'. In other words, it is impossible for the verifier to figure out which of the valid secret keys the prover is using to authenticate himself.

Parameter	Definition	Sample Instantiations			
n	integer that is a power of 2	512	512	512	1024
m	any integer	4	5	8	8
σ	any integer	127	2047	2047	2047
κ	integer s.t. $2^{\kappa}\binom{n}{\kappa} \geq 2^{160}$	24	24	24	21
p	integer $\approx (2\sigma+1)^m \cdot 2^{-\frac{128}{n}}$	$2^{31.7}$	$2^{59.8}$	$2^{95.8}$	$2^{95.9}$
R	ring $\mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$				
D	$\{\mathbf{g} \in R : \lVert\mathbf{g}\rVert_\infty \leq mn\sigma\kappa\}$				
D_s	$\{\mathbf{g} \in R : \lVert\mathbf{g}\rVert_\infty \leq \sigma\}$				
D_c	$\{\mathbf{g} \in R : \lVert\mathbf{g}\rVert_1 \leq \kappa\}$				
D_y	$\{\mathbf{g} \in R : \lVert\mathbf{g}\rVert_\infty \leq mn\sigma\kappa\}$				
G	$\{\mathbf{g} \in R : \lVert\mathbf{g}\rVert_\infty \leq mn\sigma\kappa - \sigma\kappa\}$				
Signature Size	$\approx mn \log(2mn\sigma\kappa)$ bits	49000	72000	119000	246000
Public Key Size	$\approx n \log p$ bits	16000	31000	49000	98000
Secret Key Size	$\approx mn \log(2\sigma+1)$ bits	16000	31000	49000	98000
Hash Function Size	$\approx mn \log p$ bits	65000	153000	392000	786000
Length of vector needed to break signature		$2^{23.5}$	$2^{27.9}$	$2^{28.6}$	$2^{29.4}$
Length of shortest vector that can be found		$2^{25.5}$	$2^{36.7}$	$2^{47.6}$	$2^{69.4}$

Fig. 2. Lattice-Based Schemes' Parameter Definitions and Sample Instantiations

3 Lattice-Based Constructions

In this section, we present our lattice-based identification (Figure 3) and signature (Figure 4) schemes. In Figure 2 we define all the parameters that will appear in this section as well as give some concrete instantiations. The parameter κ controls the size of the domain from which the challenges/signatures come from. In order to have soundness error of at most 2^{-80}, this parameter must be set such that the size of this domain is 2^{160}. The parameter p is chosen such that every public key has a very high probability of having multiple corresponding secret keys associated with it. The free parameters n, m, and σ need to be set in a way so that it is computationally infeasible find collisions in the underlying hash function family $\mathcal{H}(R, D, m)$.

The last two lines of the above table deal with the practical cryptanalysis of our signature scheme. The last line of the table specifies the length of the shortest vector in a certain lattice defined by our signature scheme that can be found in practice, while the line above that specifies the length of the vector that needs to be found in order to forge a signature. See Section 3.3 for more details.

3.1 Identification Scheme

The secret key of the prover, denoted $\widehat{\mathbf{s}}$, consists of a set of m polynomials from the set D_s which are picked uniformly at random. The public key of the prover consists of a hash function h which is picked randomly from the family $\mathcal{H}(R, D, m)$, and the polynomial $\mathbf{S} = h(\widehat{\mathbf{s}})$. We point out that it is not necessary for every prover to have a distinct h. If trusted randomness is available, then everyone can share the same random h which considerably lowers the public key size because the hash function h can be hard-coded into the signing and verification algorithms.

In the first step of the protocol, the prover picks a random $\widehat{\mathbf{y}} \in D_y^m$, and "commits" to it by sending $\mathbf{Y} = h(\widehat{\mathbf{y}})$ to the verifier. The verifier then picks a random challenge $\mathbf{c}$ from D_c and sends it to the prover. The prover then computes $\widehat{\mathbf{z}} = \widehat{\mathbf{s}}\mathbf{c} + \widehat{\mathbf{y}}$. If this result falls into the range G^m, the prover sends it to the verifier. Otherwise, he aborts the protocol. Upon receiving $\widehat{\mathbf{z}}$, the verifier accepts the interaction if $\widehat{\mathbf{z}} \in \mathrm{G}^m$ and $h(\widehat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$. Using the homomorphic properties of h (see (1) and (2)), we see that $h(\widehat{\mathbf{s}}\mathbf{c} + \widehat{\mathbf{y}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$, and so an honest prover who does not abort will always be accepted.

Proving the soundness and completeness of the protocol is done using the following series of steps:

1. Show that an honest prover is accepted with probability $1/e$.
2. Show that the ID scheme is perfectly witness-indistinguishable.
3. Show that with probability $1 - 2^{-128}$, for a randomly-picked $\widehat{\mathbf{s}} \in D_s^m$, there is another $\widehat{\mathbf{s}}' \in D_s^m$ such that $h(\widehat{\mathbf{s}}) = h(\widehat{\mathbf{s}}')$.
4. Show how to extract a collision in h from an adversary who succeeds in breaking the protocol

Step 1 shows that the completeness of the protocol is $1/e$. We will explain how to increase this number later. Step 2 is essentially the main part of the proof, which shows that for every pair of possible secret keys $\widehat{\mathbf{s}}, \widehat{\mathbf{s}}'$ such that $\mathbf{S} = h(\widehat{\mathbf{s}}) = h(\widehat{\mathbf{s}}')$, no adversarial verifier can determine which secret key is being used by the prover. The reason for this is that we have set up the parameters so that for every secret key $\widehat{\mathbf{s}} \in D_s^m$, every challenge $\mathbf{c} \in D_c$, and every response $\widehat{\mathbf{z}} \in \mathrm{G}^m$, the value of $\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}$ is in D_y. This implies that having seen the history $(\mathbf{Y}, \mathbf{c}, \widehat{\mathbf{z}})$, it is impossible to tell whether the secret key was $\widehat{\mathbf{s}}$ and we picked masking parameter $\widehat{\mathbf{y}}$, or the secret key was $\widehat{\mathbf{s}}'$ and we picked the masking parameter $\widehat{\mathbf{y}}' = \widehat{\mathbf{z}} - \widehat{\mathbf{s}}'\mathbf{c} = \widehat{\mathbf{y}} + \widehat{\mathbf{s}}\mathbf{c} - \widehat{\mathbf{s}}'\mathbf{c} = \widehat{\mathbf{y}} + (\widehat{\mathbf{s}} - \widehat{\mathbf{s}}')\mathbf{c}$ because $h(\widehat{\mathbf{s}}) = h(\widehat{\mathbf{s}}') = \mathbf{S}$ and $h(\widehat{\mathbf{y}}) = h(\widehat{\mathbf{y}}') = \mathbf{Y}$.

To make the claim in step 2 non-vacuous, we need to show that for a randomly picked secret key, there is indeed a high probability that another secret key exists which produces the same public key. This is done in step 3.

In step 4, we show how to use a successful adversary to solve the $\mathrm{Col}(h, D)$ problem for a random $h \in \mathcal{H}(R, D, m)$. Given a random $h \in \mathcal{H}(R, D, m)$, we pick a random secret key $\widehat{\mathbf{s}}$ and publish the public keys h and $\mathbf{S} = h(\widehat{\mathbf{s}})$. In the first stage of the attack, the adversary plays the role of the verifier, and we are able to perfectly play the part of the prover because we know the secret key. In the second stage when the adversary attempts to impersonate the prover, we receive his commitment, and send a random challenge $\mathbf{c} \in D_c$. After he responds with $\widehat{\mathbf{z}}$, we rewind and pick another random challenge $\mathbf{c}' \in D_c$, to which the adversary will respond with $\widehat{\mathbf{z}}'$. The responses of the adversary and our knowledge of the secret key allow us to obtain the equation $h(\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}) = h(\widehat{\mathbf{z}}' - \widehat{\mathbf{s}}\mathbf{c}')$. By our choice

Private key: $\widehat{\mathbf{s}} \xleftarrow{\$} D_s^m$
Public key: $h \xleftarrow{\$} \mathcal{H}(R, D, m), \mathbf{S} \leftarrow h(\widehat{\mathbf{s}})$

Prover		Verifier
$\widehat{\mathbf{y}} \xleftarrow{\$} D_y^m,\ \mathbf{Y} \leftarrow h(\widehat{\mathbf{y}})$	$\xrightarrow{\mathbf{Y}}$	
	$\xleftarrow{\mathbf{c}}$	$\mathbf{c} \xleftarrow{\$} D_c$
$\widehat{\mathbf{z}} \leftarrow \widehat{\mathbf{s}}\mathbf{c} + \widehat{\mathbf{y}}$		
if $\widehat{\mathbf{z}} \notin \mathrm{G}^m$ then $\widehat{\mathbf{z}} \leftarrow \perp$	$\xrightarrow{\widehat{\mathbf{z}}}$	
		Accept iff $\widehat{\mathbf{z}} \in \mathrm{G}^m$ and $h(\widehat{\mathbf{z}}) = \mathbf{S}\mathbf{c} + \mathbf{Y}$

Fig. 3. Lattice-Based Identification Scheme.

of parameters, both $\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}$ and $\widehat{\mathbf{z}}' - \widehat{\mathbf{s}}\mathbf{c}'$ are in D, and because of the witness-indistinguishability of the protocol, the adversary cannot know our exact secret key. Therefore with probability at least $1/2$, $\widehat{\mathbf{z}} - \widehat{\mathbf{s}}\mathbf{c}$ and $\widehat{\mathbf{z}}' - \widehat{\mathbf{s}}\mathbf{c}'$ will be distinct and we have a collision for h. Thus an adversary who can break the ID scheme can be used to solve $\mathrm{Col}(h, D)$ for random $h \in \mathcal{H}(R, D, m)$, and by Theorem 1, this implies finding the approximate short vector in all $(\mathbf{x}^n + 1)$-cyclic lattices.

Theorem 2. *If the identification scheme in Figure 3 is insecure against active attacks for the parameters in Table 2, then there is polynomial-time algorithm that can solve* $\mathrm{SVP}_\gamma(\Lambda)$ *for* $\gamma = \tilde{O}(n^2)$ *for every lattice* Λ *corresponding to an ideal in the ring* $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$*.*

Notice that the ID scheme is not quite satisfactory because a valid prover is only accepted with probability $1/e$. This means that the scheme may have to be repeated several times until the prover succeeds. Because we showed that the scheme is witness-indistinguishable, the repetitions can be performed in parallel, and the witness-indistinguishability property will still be preserved [6]. So the straight-forward way to modify the ID scheme would be, for example, to pick 30 different $\widehat{\mathbf{y}}_i$'s and send the $\mathbf{Y}_i = h(\widehat{\mathbf{y}}_i)$ to the verifier. Then the verifier will send 30 challenges, and the prover replies to the first one of these challenges that he can. This would result in a protocol where the honest prover is accepted with probability about $1 - 2^{-20}$.

But there are some significant improvements that can be made. First of all, the verifier needs to send only one challenge, rather than one challenge for every commitment (this is because we show that for every challenge $\mathbf{c}$, the probability of aborting is equal over the random choice of $\widehat{\mathbf{y}}$). And secondly, we can use a standard trick to shorten the length of every $\mathbf{Y}_i$, which will result in large savings in our protocol because the length of each $\mathbf{Y}$ is approximately $n \log p$ bits, which could be as large as 100,000 bits! Instead of sending $\mathbf{Y}$, we can send $H(\mathbf{Y})$ where H is any collision resistant hash function. Unlike with h, we will not need H to have any algebraic properties like (1) and (2), so H could be a cryptographic hash function such as SHA or an efficient lattice-based hash function from [20] whose output is about 512 bits. So sending 30 $H(\mathbf{Y})$'s will only require about $15,000$ bits in total. In this modified protocol, the verifier's challenge and the prover's reply remain the same as in the old protocol. But to authenticate the

Signing Key: $\widehat{\mathbf{s}} \xleftarrow{\$} D_s^m$
Verification Key: $h \xleftarrow{\$} \mathcal{H}(R, D, m), \mathbf{S} \leftarrow h(\widehat{\mathbf{s}})$
Random Oracle: $\mathrm{H} : \{0,1\}^* \rightarrow D_c$

```
Sign($\mu, h, \widehat{\mathbf{s}}$)
  1: $\widehat{\mathbf{y}} \xleftarrow{\$} D_y^m$
  2: $\mathbf{e} \leftarrow \mathrm{H}(h(\widehat{\mathbf{y}}), \mu)$
  3: $\widehat{\mathbf{z}} \leftarrow \widehat{\mathbf{s}}\mathbf{e} + \widehat{\mathbf{y}}$
  4: if $\widehat{\mathbf{z}} \notin \mathrm{G}^m$, then goto step 1
  5: output $(\widehat{\mathbf{z}}, \mathbf{e})$
```

```
Verify($\mu, \widehat{\mathbf{z}}, \mathbf{e}, h, \mathbf{S}$)
  1: Accept iff
       $\widehat{\mathbf{z}} \in \mathrm{G}^m$ and $\mathbf{e} = \mathrm{H}(h(\widehat{\mathbf{z}}) - \mathbf{S}\mathbf{e}, \mu)$
```

Fig. 4. Lattice-Based Signature Scheme

prover, the verifier checks whether $\widehat{\mathbf{z}} \in \mathrm{G}^m$ and that $H(h(\widehat{\mathbf{z}}) - \mathbf{Sc})$ is equal to some $H(\mathbf{Y})$ sent by the prover in the first step [2]. It can be shown that an adversary who breaks this protocol can be used to find a collision either in H or in h. We will give more details in the full version of the paper.

3.2 Signature Scheme

Our signature scheme is presented in Figure 4. The public and secret keys are just like in the ID scheme. To sign a message μ, we pick a random $\widehat{\mathbf{y}}$ and compute $\mathbf{e} = \mathrm{H}(h(\widehat{\mathbf{y}}), \mu)$ and send $(\widehat{\mathbf{z}}, \mathbf{e})$ as the signature only if $\widehat{\mathbf{z}}$ is in the set G^m. Otherwise we repeat the procedure until $\widehat{\mathbf{z}}$ ends up in G^m. The probability that we succeed in getting $\widehat{\mathbf{z}}$ to be in G^m on any particular try is the same as the probability that the ID scheme in Figure 3 doesn't send $\perp$, which is $1/e$. So we expect to repeat the signing procedure less than 3 times to get a signature.

The witness-indistinguishability of the signature scheme follows directly from the witness indistinguishability of the ID scheme because the challenge is now simply generated by a random oracle rather than by the verifier. The proof of security of the signature scheme uses the forking lemma [32] to obtain two signatures from a forger that use the same random oracle query. Then using the same ideas as in the security proof of the ID scheme, it can be shown how to use these signatures to obtain a solution to the $\mathrm{Col}(h, D)$ problem for a random $h \in \mathcal{H}(R, D, m)$.

Theorem 3. *If the signature scheme in Figure 4 for the parameters in Table 2 is not strongly unforgeable, then there is a polynomial-time algorithm that can solve* $\mathrm{SVP}_\gamma(\Lambda)$ *for* $\gamma = \tilde{O}(n^2)$ *for every lattice* Λ *corresponding to an ideal in the ring* $\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$.

3.3 Concrete Parameters

The security of our ID (and signature) scheme depends on its soundness and the hardness of finding collisions in hash functions from a certain family. As

[2] One could lower the communication complexity even further by combining the 30 hashes into a hash tree.

mentioned earlier, we set the parameters κ and p such that the soundness error is at most 2^{-80}. We now discuss how to set the remaining parameters so that finding collisions in the resulting hash function is infeasible with the techniques known today. For this, we will use the work of [8], who showed that, given a reasonable amount of time, algorithms for finding short vectors in random lattices produce a vector that is no smaller than 1.01^n times the shortest vector of the lattice.

We showed that an adversary who succeeds in forging a signature can be used to find a collision in a hash function chosen randomly from $\mathcal{H}(R, D, m)$. This is equivalent to finding "short" vectors a certain lattice which we will now define. For a polynomial $\mathbf{a} \in \mathbb{Z}_p[\mathbf{x}]/\langle \mathbf{x}^n + 1\rangle$, let $Rot(\mathbf{a})$ be the $n \times n$ matrix whose i^{th} column is the polynomial $\mathbf{a}\mathbf{x}^i$, and let $\mathbf{A}$ be the $n \times nm$ matrix $\mathbf{A} = [Rot(\mathbf{a}_1)||Rot(\mathbf{a}_2)||\ldots||Rot(\mathbf{a}_m)]$ where $\mathbf{a}_i$ are the polynomials which define the hash function h. If we define the lattice $\Lambda_p^{\perp}(\mathbf{A}) = \{\mathbf{u} \in \mathbb{Z}^{mn} : \mathbf{A}\mathbf{u} = 0(\bmod p)\}$, then finding a vector $\mathbf{u} \in \Lambda_p^{\perp}(\mathbf{A})$ whose ℓ_∞ norm is at most $2mn\sigma\kappa$ is equivalent to finding a collision in $h \in \mathcal{H}(R, D, m)$.

The random lattices on which the experiments of [8] were run differ from $\Lambda_p^{\perp}(\mathbf{A})$, but in [25], experiments were run on lattices that are very similar [3] to $\Lambda_p^{\perp}(\mathbf{A})$ which obtained the same results as [8]. Furthermore, it was shown in [25] that it is inefficient to try to find a short vector in $\Lambda_p^{\perp}(\mathbf{A})$ by using all its mn dimensions. Rather, one should only use the first $\sqrt{n \log p / \log 1.01}$ dimensions and zero out the others. This results in a vector whose ℓ_2 length is $\min\{p, 2^{2\sqrt{n \log p \log 1.01}}\}$, and whose ℓ_∞ norm is at least

$$\min\{p, 2^{2\sqrt{n \log p \log 1.01}} \cdot (n \log p / \log 1.01)^{-1/4}\} \tag{3}$$

Since solving the $\mathrm{Col}(h, D)$ problem is equivalent to finding an element $\widehat{\mathbf{y}}$ such that $h(\widehat{\mathbf{y}}) = 0$ and $\|\widehat{\mathbf{y}}\|_\infty \le 2mn\sigma\kappa$, we want to make sure that when we set our parameters, the value of $2mn\sigma\kappa$ is smaller than the value in (3). In the instantiation of the scheme that produces a signature of length approximately 49000 bits, the value of $2mn\sigma\kappa$ is around $2^{23.5}$, while the value of the shortest vector (in the ℓ_∞ norm) that can be found according to (3) is around $2^{25.5}$ (see the last two lines of the table in Figure 2).

We hope that our work provides further motivation for studying lattice-reduction algorithms for lattices of the form $\Lambda_p^{\perp}(\mathbf{A})$, which also happen to be central to the cryptanalysis of other lattice-based schemes such as [20, 19, 15].

[3] The lattices in [25] were just like $\Lambda_p^{\perp}(\mathbf{A})$, except each entry of $\mathbf{A}$ was chosen uniformly at random modulo p. Since the currently best lattice-reduction algorithms don't "see" the algebraic structure of the lattice, it is very reasonable to assume that their performance will be the same on our lattices and the lattices in [25]. Of course it's possible that a different algorithm that has yet to be discovered will be able to use the algebraic structure of $\mathbf{A}$ to achieve better results.

```
Secret Key: $s \xleftarrow{\$} D_s$
Public Key: $N$, $g$, and $S \leftarrow g^s \bmod N$
Random Oracle: $\mathrm{H} : \{0,1\}^* \to D_c$

Sign($\mu, N, g, s$)
  1: $y \xleftarrow{\$} D_y$
  2: $e \leftarrow \mathrm{H}(g^y \bmod N, \mu)$
  3: $z \leftarrow se + y$
  4: [if $z \notin \mathrm{G}$, then goto step 1]
  5: output (z,e)

Verify($\mu, z, e, N, g, S$)
  1: Accept iff $e = \mathrm{H}(g^z S^{-e} \bmod N, \mu)$
```

Fig. 5. Factoring-Based Signature Schemes. Line 4 is only executed in the aborting scheme.

	Without Aborting	With Aborting
k	128	
N	1024-bit product of two 2^k-strong primes	
g	asymmetric basis in $\mathbb{Z}_N^*$ such that $ord(g)$ has 160 bits	
σ	2^{168}	
D_c	$\{0, \ldots, 2^k\}$	
D_s	$\{0, \ldots, \sigma\}$	
k'	64	-
D_y	$\{0, \ldots, 2^{k+k'} \cdot \sigma\}$	$\{0, \ldots, 2^{k+1} \cdot \sigma\}$
G	-	$\{2^k \cdot \sigma, \ldots, 2^{k+1} \cdot \sigma\}$
Signature Size (bits)	488	425

Fig. 6. Factoring-Based Scheme's Variable Definitions.

4 Factoring-Based Constructions

We now present a modification of a signature scheme presented in [31] whose security is based on the hardness of factoring . We will need the following two definitions from [31].

Definition 5. *A prime p is said to be α-***strong** *if $p = 2r + 1$ where r is an integer whose prime factors are all greater than α.*

Definition 6. *Let $N = pq$, where p and q are primes. Then an element $g \in \mathbb{Z}_N^*$ is said to be an* **asymmetric basis** *if the parity of $ord(g)$ in $\mathbb{Z}_p^*$ differs from the parity of $ord(g)$ in $\mathbb{Z}_q^*$.*

Both schemes are presented in Figure 5 (our scheme only differs from that in [31] by the addition of line 4), and the parameters in [31] as well as our modified parameters are presented in Figure 6. We point out that the scheme of [31] is a variant of Girault's scheme [10], and our technique of shortening the signature length would apply equally well to all its variants [10, 31, 11] as well as to the blind signature constructed in [31].

The signature of a message μ consists of the pair (z, e). The length of z in the non-aborting version of the protocol has length $k + k' + \log \sigma = 360$, while in our protocol the length is $k + 1 + \log \sigma = 297$. The savings are essentially due to the fact that we can pick y in a much smaller range, and the fact that we are allowed to abort keeps the scheme secure.

If in step 4, z is not in G, then the signing procedure has to be repeated. It can be shown that this happens with probability 1/2. So we expect to run the signing protocol twice for every signature. But if we assume that off-line computations (i.e. computations before receiving the message) are free, then we can change the protocol so that we expect to compute just one extra random oracle query over the non-aborting signature scheme. The way to do this is to always keep several y_i and $g^{y_i} \bmod N$ stored along with the ranges that e would have to fall into so that $se + y_i \in \mathrm{G}$ (the range is just $(\mathrm{G} - y_i)/s$). Then when we are asked to sign a message μ, we compute $e = \mathrm{H}(g^{y_1} \bmod N, \mu)$ and then check whether it's in the valid range of y_1. If it is, then we compute $sc + y_1$ and output it. If it's not, then we recompute e using y_2, and so on. The important thing to note is that we only compute $sc + y_i$ once, and we still expect to succeed after two tries. As an added bonus, we only use up one y_i per message, since the y_i that "didn't work" can be safely tried for the next message.

Theorem 4. *An adversary who breaks the aborting signature scheme in T steps can be used to factor N in poly(T) steps.*

Acknowledgements. I am very grateful to Daniele Micciancio for suggesting many substantial improvements and simplifications to an earlier version of this work. I would also like to thank the anonymous referees for their useful suggestions.

References

1. M. Abdalla, J.H. An, M. Bellare, and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT*, pages 418–433, 2002.
2. M. Ajtai. Generating hard instances of lattice problems. In *STOC*, 1996.
3. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, 1997. An improved version is described in ECCC 2007.
4. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.
5. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1(2):77–94, 1988.
6. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.
7. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.
8. N. Gama and P. Q. Nguyen. Predicting lattice reduction. In *EUROCRYPT*, 2008.
9. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices, and new cryptographic constructions. In *STOC*, 2008.
10. M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. In *EUROCRYPT*, pages 481–486, 1990.
11. M. Girault, G. Poupard, and J. Stern. On the fly authentication and signature schemes based on groups of unknown order. *J. Cryptology*, 19(4):463–487, 2006.

12. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
13. L. Guillou and J.J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *CRYPTO*, pages 216–231, 1988.
14. J. Hoffstein, J. Pipher, and J. H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.
15. A. Kawachi, K. Tanaka, and K. Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, 2008.
16. A. K. Lenstra, H. W. Lenstra Jr., and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, (261):513–534, 1982.
17. V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *Public Key Cryptography*, pages 162–179, 2008.
18. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155, 2006.
19. V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures. In *TCC*, pages 37–54, 2008.
20. V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen. SWIFFT: a modest proposal for FFT hashing. In *FSE*, 2008.
21. R. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378, 1987.
22. R. Merkle. A certified digital signature. In *CRYPTO*, pages 218–238, 1989.
23. D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007.
24. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. on Computing*, 37(1):267–302, 2007.
25. D. Micciancio and O. Regev. Lattice-based cryptography. In D. Bernstein and J. Buchmann, editors, *Chapter in Post-quantum Cryptography*. Springer, 2009.
26. D. Micciancio and S. Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In *CRYPTO*, pages 282–298, 2003.
27. T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO*, pages 31–53, 1992.
28. C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.
29. C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, 2006.
30. C. Peikert and A. Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In *STOC*, 2007.
31. D. Pointcheval. The composite discrete logarithm and secure authentication. In *Public Key Cryptography*, pages 113–128, 2000.
32. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.
33. O. Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.
34. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.
35. C.P. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.
36. P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997.
37. D. Stehle, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public-key encryption based on ideal lattices. In *ASIACRYPT*, 2009.