Formalizing the Hidden Number Problem in Isabelle/HOL

Sage Binder[1]
University of Iowa, Iowa City, IA, USA

Eric Ren
Independent, Atlanta, GA, USA

Katherine Kosaian
University of Iowa, Iowa City, IA, USA

Abstract

We formalize the hidden number problem (HNP), as introduced in a seminal work by Boneh and Venkatesan in 1996, in Isabelle/HOL. Intuitively, the HNP involves demonstrating the existence of an algorithm (the "adversary") which can compute (with high probability) a hidden number α given access to a bit-leaking oracle. Originally developed to establish the security of Diffie–Hellman key exchange, the HNP has since been used not only for protocol security but also in cryptographic attacks, including notable ones on DSA and ECDSA. Further, as the HNP establishes an expressive paradigm for reasoning about security in the context of information leakage, many HNP variants for other specialized cryptographic applications have since been developed. A main contribution of our work is explicating and clarifying the HNP proof blueprint from the original source material; naturally, formalization forces us to make all assumptions and proof steps precise and transparent. For example, the source material did not explicitly define the adversary and only abstractly defined what information is being leaked; our formalization concretizes both definitions. Additionally, the HNP makes use of an instance of Babai's nearest plane algorithm, which solves the approximate closest vector problem; we formalize this as a result of independent interest. Our formalizations of Babai's algorithm and the HNP adversary are executable, setting up potential future work, e.g. in developing formally verified instances of cryptographic attacks.

2012 ACM Subject Classification Security and privacy → Logic and verification

Keywords and phrases hidden number problem, Babai's nearest plane algorithm, cryptography, interactive theorem proving, Isabelle/HOL

Digital Object Identifier 10.4230/LIPIcs.ITP.2025.23

Acknowledgements Thank you to Yong Kiam Tan for useful comments on the paper, and thank you to the anonymous ITP reviewers for their valuable feedback.

1 Introduction

The *hidden number problem (HNP)* [11] and its variants have widespread applications in cryptography. At a high level, the goal of the HNP is to demonstrate the existence of an algorithm (the "adversary") which can compute (with high probability) a hidden number α given access to an oracle which leaks some information about α. In particular, the oracle is specifically chosen to model the kind of information that may allow an attacker to gain partial knowledge of encrypted messages. Thus, the existence of an adversary demonstrates that gaining partial knowledge of encrypted messages is as difficult as gaining full knowledge. Initially, the HNP was used to analyze the security of the Diffie–Hellman key exchange protocol. The HNP and its variants have also been used in many cryptographic attacks (both

[1] Corresponding author

more theoretical analyses and real-world instances), including ones on DSA [36, 37], ECDSA [38, 13, 41, 35, 19, 34], the Learning With Errors (LWE) problem [27], the MEGA cloud storage provider [21], and more [33]. Our interest in formalizing the HNP originated in a desire to formalize the mathematics underlying common cryptographic attacks. Specifically, the HNP paradigm is very useful for proving that a cryptographic attack is likely to succeed, since it provides a mathematical framework which represents the attacker's task of computing secret information (the hidden α) given some public/leaked information (modeled by the oracle).

We contribute (to our knowledge) the first formalization of the HNP, following Boneh and Venkatesan's original work [11]. Our work totals ∼7000 LoC in Isabelle/HOL, ∼4000 of which are specific to the HNP proof. Additionally, the HNP relies on a result of independent interest – *Babai's nearest plane algorithm* [1], which takes a target vector and a lattice as input and returns a lattice vector that is close to the target vector. Babai's algorithm itself relies heavily on the famous LLL basis reduction algorithm, which takes a lattice basis and produces a "nearly orthogonal" basis for the same lattice [28]. The LLL algorithm has already been formalized in Isabelle/HOL [12, 43, 18], and we build on this development[2] to formalize the most relevant instance of Babai's algorithm (which can be stated at various levels of generality), balancing executability, limitations due to the existing LLL formalization, and ease of proof. Our formalization of Babai's algorithm is ∼2450 LoC. Our formalizations of Babai's algorithm and the HNP adversary are executable via Isabelle/HOL's code generator. Our work focuses on correctness and defers a formalized complexity analysis to future work.

Throughout this paper, we highlight places where we differ from the source material. Though Boneh and Venkatesan's original work [11] presents deep mathematical ideas, it also leaves much to be clarified. For instance, the definition of the oracle is not fully specified, and the main theorem is stated for unspecified "large enough n". In Section 2, we clarify the statement of the main theorem and also present the specific instance of Babai's algorithm we formalize. Further, while the source material successfully communicates the general proof techniques employed, our formalization requires a very precise reorganization of the original proof. For example, Boneh and Venkatesan leave the definition of the adversary implicit in the proof, omit many arithmetic details, and assert without explanation some steps involving key properties of definitions. We discuss our proof reorganization and clarifications, including an explicit presentation of the adversary, in Section 3. During formalization, low-level technical challenges naturally arise; we discuss some technical details of both the Babai's algorithm and HNP formalizations in Section 4. Following this, we discuss related work in Section 5 and conclude with a discussion of potential future work in Section 6. Our formalization is available on the Isabelle Archive of Formal Proofs (AFP) [8].

2 Preliminaries

We now turn to precise mathematical descriptions of Babai's algorithm and the HNP.

2.1 Babai's nearest plane algorithm

The *closest vector problem* is the problem of finding a vector $v_{\min}$ in a given lattice L that is closest to a target vector u – that is, finding $v_{\min}$ witnessing $D := \min\{\|v - u\| : v \in L\}$. Although the task of finding $v_{\min}$ is NP-complete [24], it is possible to find a vector that is

[2] To our knowledge, LLL has not yet been verified in other theorem provers, making Isabelle/HOL the natural choice for our work.

not too much further from u than $v_{\min}$ in polynomial time. In particular, Babai's nearest plane algorithm finds a vector $\hat{v} \in L \subseteq \mathbb{R}^n$ where $\|\hat{v} - u\| \leq \sqrt{n}(\sqrt{\delta})^d D$, where $\delta \geq 4/3$ is an adjustable approximation factor and d is the dimension of the lattice L.

The most general form of Babai's algorithm allows a target vector $u \in \mathbb{R}^n$ and a lattice $L \subseteq \mathbb{R}^n$ spanned by d vectors (with $d \leq n$). However, the algorithm requires obtaining a "nearly orthogonal" basis of L by applying the LLL basis reduction algorithm [28], which in Isabelle/HOL has been formalized for lattices $L \subseteq \mathbb{Z}^n$ [12, 43, 18]. Since we build on this existing LLL formalization, our formalization of Babai's algorithm also only accepts lattices in $\mathbb{Z}^n$. We can apply our algorithm to a lattice $\mathbb{Q}^n$ by first scaling it to a lattice in $\mathbb{Z}^n$ (a technique we apply later on), but this workaround is not available for lattices in $\mathbb{R}^n$. The target vector u may be drawn from $\mathbb{Q}^n$. We also focus on the $n = d$ case, so our formal algorithm only accepts lattices where $n = d$ (which is sufficient for our purposes in the hidden number problem).

We follow a set of lecture notes by Stephens-Davidowitz [42] which addresses the relevant $n = d$ case. The informal proof, worded geometrically, references the hyperplanes defined by integer multiples of the lattice basis vectors. Reformulated algebraically, this is an argument about the coefficients of lattice vectors as linear combinations of the basis vectors. In the formal proof, we obtain these coefficients by inverting the matrix corresponding to the lattice basis vectors, which is only possible if said matrix is square, i.e., when $n = d$. While we explored ways of lifting the $n = d$ special case to a more general result for $n \leq d$, we did not succeed;[3] as the matrix and its inverse are involved in many of the proof details, we expect that generalizing the result may require changes to large portions of the proof. Hence, specializing to the $n = d$ case appears to present a non-trivial limitation. We discuss the details of our proof strategy and the formalized result in Section 4.1.

2.2 The Hidden Number Problem

Let p be a large prime. Intuitively, the HNP, introduced by Boneh and Venkatesan [11], is concerned with designing an adversary capable of finding a fixed "hidden number" $\alpha \in (\mathbb{Z}/p\mathbb{Z})^\times$ with high probability (where $(\mathbb{Z}/p\mathbb{Z})^\times$ is the set $\{1, \ldots, p-1\}$ viewed as a group under multiplication modulo p), given access to an oracle which (roughly speaking) leaks the first k bits of $(\alpha \cdot t) \bmod p$ for randomly sampled values of t (and a fixed choice of k).

To make this precise, Boneh and Venkatesan introduce the MSB_k operator, where they initially present $\mathrm{MSB}_k(x)$ as "the integer t such that $(t-1) \cdot p/2^k \leq x < t \cdot p/2^k$" [11]. This can be thought of as giving approximately k bits of information about x. However, this is not quite the MSB_k function they ultimately use – they assume that MSB_k is slightly modified from the previous description to satisfy $|x - \mathrm{MSB}_k(x)| < p/2^{k+1}$ (though they do not concretely describe the modification). The rest of the theorem statement and proof relies only on this bound and not on the precise definition of MSB_k. In fact, the proof turns out to only require a bound of $p/2^k$. In our formalization, we let the MSB_k function be arbitrary, subject to this assumption, but we do subsequently define two concrete MSB_k functions (a non-computational definition modeled on the original (incomplete) description given by Boneh and Venkatesan, and a more intuitive computational definition) and prove that they satisfy this assumption (see Section 4.2).

[3] We considered embedding a lattice with $d < n$ into $\mathbb{R}^d$, or padding out such a lattice with additional but superfluous basis vectors, but neither workaround succeeded.

With MSB_k fixed, the adversary receives a sequence of tuples $\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big)$, where $\mathcal{O}$ is an oracle defined by $\mathcal{O}(x) = \mathrm{MSB}_k((x \cdot \alpha) \bmod p)$ and the t_i's are uniformly and independently sampled from $(\mathbb{Z}/p\mathbb{Z})^\times$. Given this input, the adversary then tries to produce the original hidden number α. A successful adversary implies that the problem of finding α can be reduced to the problem of computing the oracle. In the context of Diffie–Hellman, a successful adversary means that if the secret key is difficult to recover, then so is partial information about the secret key.

▶ **Theorem 1.** *Fix $\alpha \in (\mathbb{Z}/p\mathbb{Z})^\times$ (the "hidden number"). Let $\mathcal{O}\colon \mathbb{N} \to \mathbb{N}$ be given by $\mathcal{O}(t) = \mathrm{MSB}_k((\alpha \cdot t) \bmod p)$ where $k := \lceil \sqrt{n} \rceil + \lceil \log(n) \rceil$. Then there exists a deterministic polynomial-time algorithm $\mathcal{A}$ (the "adversary") such that*

$$\Pr_{t_1,\ldots,t_d}[\mathcal{A}\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big) = \alpha] \geq \frac{1}{2},$$

where $d := 2\sqrt{n}$ and the t_i are sampled uniformly and independently from $(\mathbb{Z}/p\mathbb{Z})^\times$.

The original proof by Boneh and Venkatesan additionally assumes that n (and hence p) is "sufficiently large" – for our formalization, we explicitly assume $n > 961$, which we derive from both our particular bound for Babai's algorithm (discussed later in Section 4.1) and the inequalities used in the proof of Theorem 1. As n (abstractly) represents the number of bits of a secret key, this lower bound does not introduce any practical restrictions – modern encryption schemes use secrets much longer than 961 bits.

Note that while Theorem 1 establishes that the adversary has polynomial time complexity, we focus on the correctness of the adversary and do not address complexity. As the existing LLL formalization includes a formal complexity analysis [12, 43, 18], we believe that extending their approach to a formal complexity analysis of Babai's algorithm and the HNP adversary is very doable. This is because Babai's algorithm is a rather direct application of LLL, and the HNP adversary we will present (Section 3.1) is a direct application of Babai's algorithm. Nevertheless, we see verifying complexity as interesting future work, as formal verification has historically revealed problems in informal complexity analysis [20].

For the convenience of the reader, we collect some of our frequently-used notation and abbreviations in Table 1 (though this notation will be explained as it is introduced throughout Section 3). We attempt to make our notation consistent with the original paper, though we introduce floor and ceiling operators where appropriate.[4]

3 Formalizing the Hidden Number Problem

We found it useful to develop an informal blueprint to guide our formalization of the HNP. Our blueprint explicates omitted proof details and under-specified definitions, and clarifies the overall organization of the proof. As Theorem 1 is an existence theorem, its proof naturally involves first defining the witness (the adversary), then proving that the witness satisfies the theorem statement. We will define the adversary and present our proof blueprint.

Note that in our discussion (both preceding and forthcoming), we implicitly identify $(\mathbb{Z}/p\mathbb{Z})^\times$ with $\{1, \ldots, p-1\} \subseteq \mathbb{N}$ in the natural way, explicitly writing "mod p" where appropriate. For instance, $\mathcal{O}$ takes a natural number as input, but we consider $\mathcal{O}(t_i)$ where $t_i \in (\mathbb{Z}/p\mathbb{Z})^\times$; we view t_i as a natural number since the "mod p" operation is explicit in the

[4] The original paper sometimes omits details like floors and ceilings; these details are naturally explicated in the course of formalization, where all type information is explicit.

Table 1 Notation used throughout. Note, t and t_i often appear as bound variables, but we remain consistent in their usage even when bound.

Symbol	Value	Meaning
p	Fixed	A fixed, large prime.
α	Fixed	A fixed "hidden number" in $(\mathbb{Z}/p\mathbb{Z})^\times$.
$\mathcal{A}$	See Section 3.1	The adversary which tries to produce α.
t, t_i	Random variables	Uniformly, independently sampled values from $(\mathbb{Z}/p\mathbb{Z})^\times$.
$\mathcal{O}(x)$	$\text{MSB}_k((x \cdot \alpha) \bmod p)$	Oracle which leaks information about α to the adversary.
n	$\lceil \log(p) \rceil$	We view α as an n-bit integer; we assume $n > 961$.
d	$2 \cdot \lceil \sqrt{n} \rceil$	The number of samples $(t, \mathcal{O}(t))$ given to the adversary.
k	$\lceil \sqrt{n} \rceil + \lceil \log(n) \rceil$	The number of bits that the oracle $\mathcal{O}$ leaks.
μ	$\lceil \frac{1}{2}\sqrt{n} \rceil + 3$	A fixed value required for a key lemma in the proof.

definition of $\mathcal{O}$. This is reflective of our formalization, where we work with $\{1, \ldots, p-1\}$ (as a `nat set`) and explicitly apply the `mod` operator where appropriate (see the formal definitions of $\mathcal{O}$ and `game` introduced in Section 3.3). We use the "$(\mathbb{Z}/p\mathbb{Z})^\times$" notation in our presentation to emphasize its nature as a group.

3.1 Defining The Adversary

Boneh and Venkatesan do not explicitly define the adversary, but its definition is inherent in their proof [11]. Our formalization blueprint begins by defining an explicit adversary, $\mathcal{A}$. Suppose $t_1, \ldots, t_d \in (\mathbb{Z}/p\mathbb{Z})^\times$ are sampled uniformly and independently. The adversary $\mathcal{A}$ receives the list of tuples $\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big)$, defines the vector $u \coloneqq (\mathcal{O}(t_1), \ldots, \mathcal{O}(t_d), 0)$, and applies Babai's nearest plane algorithm to u and the $(d+1)$-dimensional lattice L generated by the rows of the matrix

$$\begin{pmatrix} p & 0 & \cdots & 0 & 0 \\ 0 & p & & \vdots & \vdots \\ \vdots & & \ddots & 0 & \\ 0 & \cdots & 0 & p & 0 \\ t_1 & t_2 & \cdots & t_d & 1/p \end{pmatrix}.$$

Because the formalized LLL algorithm (and hence our formalization of Babai's algorithm) operates on lattices of integral vectors, we cannot work directly with the lattice as described above. Instead, we scale this lattice up by p, resulting in a lattice $\hat{L}$ which is a subset of $\mathbb{Z}^{d+1}$. This introduces some minor casting nuisances (between `rat` and `int`), and also introduces a factor of p in the final distance bound provided by our formal Babai's algorithm, but does not substantially change the proof approach. In fact, in the proof, the scaling is almost an afterthought – almost the entire proof works with the original lattice L, and the scaling only arises in the formal "glue" between the adversary implementation and the final proof steps.

So, from Babai's algorithm, $\mathcal{A}$ now possesses a vector $\hat{v} \in \hat{L}$ which is close[5] to $p \cdot u$ (which is u scaled by p). Let $\hat{\beta} \in \mathbb{Z}$ be the last coordinate of $\hat{v}$. Now, if $\hat{\beta} \equiv \alpha \pmod p$ (the "good" case), then $\mathcal{A}$ can extract α from $\hat{v}$ by computing $\alpha = (\hat{\beta} \bmod p)$. Foreshadowing, we note that the proof of Theorem 1 ultimately relies on showing that $\Pr_{t_i}[\hat{\beta} \equiv \alpha \pmod p] \geq 1/2$, i.e. the good case occurs with probability at least $1/2$.

[5] We discuss exact bounds in Section 4.1.

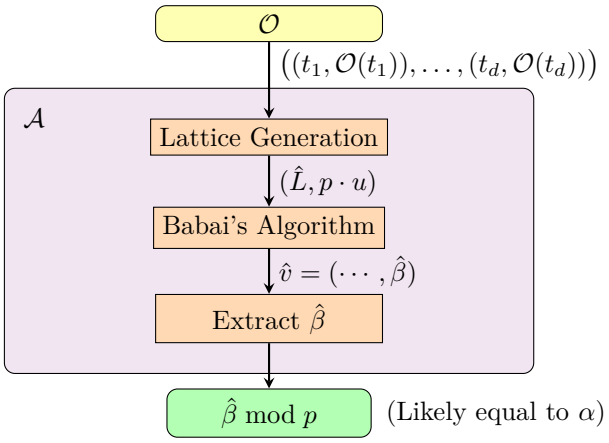

Figure 1 Visual representation of the adversary.

We use Isabelle's locale system [2] to fix ambient variables throughout our formalization (so they do not have to be redefined in every lemma statement). To start, we define our adversary within a locale that fixes the ambient variables p and d.[6]

```
locale hnp_adversary =
  fixes d p :: nat
```

This locale contains only what is necessary for the definition of the adversary; in particular, it does not contain α. Within this locale, our adversary $\mathcal{A}$ is defined as follows:

```
fun A_vec :: "(nat × nat) list ⇒ int vec" where
  "A_vec pairs =
    (let basis = int_gen_basis (map fst pairs);
      u = (map_vec rat_of_int (scaled_uvec_from_pairs pairs));
      c = 4/3
    in full_babai basis u c)"
```

```
fun A :: "(nat × nat) list ⇒ nat" where
  "A pairs =  nat (((A_vec pairs) $ d) mod p)"
```

When the adversary is used in the main theorem, the input variable `pairs` is assigned to the list of tuples $\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big)$. With this instantiation of `pairs`, `A_vec` first calls `map fst pairs` to extract the list $(t_1, \ldots, t_d)$. This list is passed to the function `int_gen_basis`, which constructs a basis for the lattice $\hat{L}$. Then, `A_vec` computes the vector $p \cdot u$, using the helper function `scaled_uvec_from_pairs` (note, `map_vec rat_of_int` is applied merely to convert types from `int vec` to `rat vec` by element-wise conversion). Next, `A_vec` applies Babai's algorithm (with an approximation factor of $c = 4/3$) to yield a vector $\hat{v}$ that is close to $p \cdot u$. Finally, the adversary $\mathcal{A}$ returns the last component of $\hat{v}$ modulo p, which is likely α. A visualization of the adversary is given in Figure 1.

[6] It is convenient to use a locale to fix these variables, but since we place no assumptions on p and d within this locale (nor do we prove any lemmas; the assumptions and lemmas appear later in a different locale for the main theorem), we could equivalently pass p and d as arguments into our definition of $\mathcal{A}$, and thread them through all helper functions.

3.2 Formalization Blueprint

Having defined the adversary $\mathcal{A}$, we must now show that it satisfies the statement of Theorem 1. We discuss the proof here in backwards fashion (starting from the main theorem and subsequently presenting helper lemmas). Let $t_1, \ldots, t_d$ be sampled uniformly and independently from $(\mathbb{Z}/p\mathbb{Z})^\times$, and let L and u be the associated lattice and vector as in the definition of $\mathcal{A}$. As noted in Section 3.1, it suffices to show that $\Pr_{t_i}[\hat{\beta} \equiv \alpha \pmod p] \geq 1/2$, where $\hat{\beta}$ is as in the definition of $\mathcal{A}$. To show this, Boneh and Venkatesan prove a so-called "uniqueness theorem", which intuitively says that with probability at least $1/2$, *every* vector with shape $(\ldots, \frac{\beta}{p}) \in L$ (with $\beta \in \mathbb{Z}$) that is close to u satisfies $\beta \equiv \alpha \pmod p$. This lemma may seem slightly surprising, since it is technically stronger than what we need, but its more general statement is a hammer that allows us to reason only about the shape of $\hat{v}$ and its closeness to u, and not about the details of Babai's algorithm itself. The uniqueness theorem is stated precisely as follows [11].

▶ **Theorem 2** (Uniqueness Theorem). *Define* $\mu \coloneqq \lceil \frac{1}{2}\sqrt{n} \rceil + 3$. *With probability at least* $1/2$, *all* $v \in L$ *such that* $|v - u| < \frac{p}{2^\mu}$ *have last coordinate* β/p *for some* $\beta \in \mathbb{Z}$ *with* $\beta \equiv \alpha \pmod p$.

Note that the probability in this theorem is considered over the uniformly, independently sampled t_i's (since u and L depend on the t_i's). Also note that it follows directly from the definition of L that every $v \in L$ has last coordinate of the form β/p for some $\beta \in \mathbb{Z}$; the non-trivial portion of Theorem 2 is the $\beta \equiv \alpha \pmod p$ claim.

Theorem 2 thus reduces the proof of Theorem 1 to simply showing that the $\hat{v}$ computed by Babai's algorithm in $\mathcal{A}$ satisfies $|\hat{v} - p \cdot u| < p \cdot \frac{p}{2^\mu}$ (note the scaling by p), a task which amounts to arithmetic manipulation of the bound given by Babai's algorithm. The bulk of our work is in formalizing Theorem 2, so we now turn to its informal proof blueprint.

We found that the original proof of Theorem 2 given by Boneh and Venkatesan was not very well-suited to formalization. For example, they do not provide many details in the steps involving probabilistic approximations, and they elide many arithmetic details; notably, they do not provide the explicit lower bound on n that is required for the proof (their result is stated for "large enough n"; formalization requires an explicit lower bound), and they omit the detailed definition unfolding and manipulation required to derive certain probabilistic inequalities. In light of this, we reformulated Boneh and Venkatesan's presentation to be more amenable to formalization. We now give the high-level structure of our proof approach, invoking the yet-to-be-stated Lemma 4, a key helper lemma which we subsequently discuss. For some abstraction, we introduce the following definition.

▶ **Definition 3.** *Call a vector* $v \in L$ *"good" if either* $|v - u| \geq \frac{p}{2^\mu}$ *or the last coordinate of* v *is equal to* β/p *for some* $\beta \in \mathbb{Z}$ *such that* $\beta \equiv \alpha \pmod p$. *If* $v \in L$ *is not good, call it "bad".*

Theorem 2 is thus saying that, with probability at least $1/2$, every $v \in L$ is good. In our development, the lemma `sampled_lattice_likely_good` is the formalization of Theorem 2.

Proof of Theorem 2. We show that the probability that some $v \in L$ is bad is less than $1/2$. First, note that $L = \bigcup_{\beta=0}^{p-1} L_\beta$ where

$$L_\beta = \{(\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta'/p) : b_1, \ldots, b_d \in \mathbb{Z},\ \beta' \equiv \alpha \pmod p\}.^7$$

Now, no vector in L_α is bad. For each $\beta \not\equiv \alpha \pmod p$, the probability that some $v \in L_\beta$ is bad is less than $\left(\frac{5}{2^\mu}\right)^d$ (this will follow from Lemma 4). Multiplying this by $p - 1$ (as there are $p - 1$ subsets L_β where $\beta \not\equiv \alpha \pmod p$) yields that the probability that some $v \in L$ is bad is less than $(p-1) \cdot \left(\frac{5}{2^\mu}\right)^d < \frac{1}{2}$. ◀

[7] In fact, $\{L_\beta : \beta \in \{0, \ldots, p-1\}\}$ is a partition of L, but this fact is not needed for the proof.

The key step in the preceding proof is the claim that, for a fixed β, if $\beta \not\equiv \alpha \pmod{p}$, then L_β is unlikely to have any vectors close to u (and thus unlikely to have any bad vectors). This is precisely stated as the following lemma, which we call `fixed_beta_bad_prob` in our development.

▶ **Lemma 4.** *Let* $\beta \in \{0, \ldots, p-1\}$ *with* $\beta \not\equiv \alpha \pmod{p}$. *Then* $\Pr_{t_i}[\exists v \in L_\beta : |v - u| < \frac{p}{2^\mu}] < \left(\frac{5}{2^\mu}\right)^d$.

Note that the probability is taken over the t_i since, for a fixed β, L_β and u are determined exactly by the t_i's. The proof of Lemma 4 uses the following definition [11].

▶ **Definition 5.** *Let* $\mathrm{dist_p} \colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{N}$ *be given by* $\mathrm{dist_p}(x, y) = \min\{|x - y - bp| : b \in \mathbb{Z}\}$.

The equivalent definition $\mathrm{dist_p}(x, y) = \min\{(x - y) \bmod p, (y - x) \bmod p\}$ is occasionally easier to work with, so we formalize this equivalence.[8]

Note that the proof of Lemma 4 relies on Lemmas 6 and 8 (which we have yet to state).

Proof of Lemma 4. Recall that $u \coloneqq (\mathcal{O}(t_1), \ldots, \mathcal{O}(t_d), 0)$. Now, suppose there exists $v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta'/p) \in L_\beta$ such that $|v - u| < \frac{p}{2^\mu}$. Then certainly each coordinate of $v - u$ has magnitude less than $\frac{p}{2^\mu}$. That is, $|\beta t_i - b_i p - \mathcal{O}(t_i)| < \frac{p}{2^\mu}$ for $i = 1, \ldots, d$. Then Lemma 6 will imply that $\mathrm{dist_p}(\beta t_i, \alpha t_i) \le \frac{2p}{2^\mu}$ for $i = 1, \ldots, d$. So,

$$\Pr_{t_i}[\exists v \in L_\beta : |v - u| < \frac{p}{2^\mu}] \le \Pr_{t_i}[\forall i \in \{1, \ldots, d\}.\ \mathrm{dist_p}(\beta t_i, \alpha t_i) \le \frac{2p}{2^\mu}].$$

Since the t_i's are independently sampled, and we are checking the probability of the same event for each t_i, we can instead compute the probability of the event for a single $t \in (\mathbb{Z}/p\mathbb{Z})^\times$ and raise the result to the dth power. Formally, defining

$$A \coloneqq \Pr_t[\mathrm{dist_p}(\beta t, \alpha t) > \frac{2p}{2^\mu}],$$

which is the probability of the *inverse* of the condition we wish to check, we have

$$\Pr_{t_i}[\forall i \in \{1, \ldots, d\}.\ \mathrm{dist_p}(\beta t_i, \alpha t_i) \le \frac{2p}{2^\mu}] = (1 - A)^d.$$

Finally, Lemma 8 will give $1 - A \le \frac{5}{2^\mu}$, which yields the desired result. ◀

Lemma 6, which we call `dist_p_helper` in our development, relies on the minimality of $\mathrm{dist_p}$ and the bound $|\mathrm{MSB}_k(x) - x| < \frac{p}{2^k}$. Additionally, our proof uses the general inequality $\mathrm{dist_p}(x, y) - \mathrm{dist_p}(x, y') \le |y - y'|$, as well as the fact that $\mathrm{dist_p}(x, y \bmod p) = \mathrm{dist_p}(x, y)$. The proof (which is entirely omitted in the source material [11]) is as follows.

▶ **Lemma 6.** *Let* $t \in (\mathbb{Z}/p\mathbb{Z})^\times$, $b \in \mathbb{Z}$, *and* $\beta \in \{0, \ldots, p-1\}$, *and suppose* $|\beta t - bp - \mathcal{O}(t)| < \frac{p}{2^\mu}$. *Then* $\mathrm{dist_p}(\beta t, \alpha t) \le \frac{2p}{2^\mu}$.

Proof. First, note that by the minimality of $\mathrm{dist_p}$, we have

$$\mathrm{dist_p}(\beta t, \mathcal{O}(t)) \le |\beta t - bp - \mathcal{O}(t)| < \frac{p}{2^\mu}. \tag{1}$$

[8] The equivalence can be seen by noting that $(x - y) \bmod p = \min(\{x - y - bp : b \in \mathbb{Z}\} \cap \mathbb{N})$ and $(y - x) \bmod p = \min(\{y - x - bp : b \in \mathbb{Z}\} \cap \mathbb{N})$.

Moreover, we have $|(\alpha t \bmod p) - \mathcal{O}(t)| \leq \frac{p}{2^k} \leq \frac{p}{2^\mu}$, which implies

$$\begin{aligned}\mathrm{dist_p}(\beta t, \alpha t) - \mathrm{dist_p}(\beta t, \mathcal{O}(t)) &= \mathrm{dist_p}(\beta t, \alpha t \bmod p) - \mathrm{dist_p}(\beta t, \mathcal{O}(t)) \\ &\leq |(\alpha t \bmod p) - \mathcal{O}(t)| \leq \frac{p}{2^\mu}.\end{aligned}$$

Finally, applying Equation (1) yields $\mathrm{dist_p}(\beta t, \alpha t) \leq \frac{p}{2^\mu} + \frac{p}{2^\mu} = \frac{2p}{2^\mu}$, as desired. ◀

Additionally, the proof of Lemma 4 also invokes Lemma 8, which bounds the quantity $1 - A$. In our formalization, we introduce the following helper lemma, which we call `dist_p_instances` in our development, to prove Lemma 8.

▶ **Lemma 7.** *Fix $B \in \mathbb{Q}_{>0}$ and $\beta \in \{0, \ldots, p-1\}$, and suppose $\beta - \alpha$ is coprime to p. Then*

$$|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist_p}(\beta t, \alpha t) \leq B\}| \leq 2B.$$

Proof. It suffices to show that, for each $x \in \{0, \ldots, \lfloor B \rfloor\}$,

$$c_x \coloneqq |\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist_p}(\beta t, \alpha t) = x\}| \leq 2.$$

Note that $c_0 = 0$ (because $\beta \neq \alpha$), so fix $x \in \{1, \ldots, \lfloor B \rfloor\}$. Now, since $\beta - \alpha$ is coprime to p, there exist unique $t_1, t_2 \in (\mathbb{Z}/p\mathbb{Z})^\times$ such that $(\beta - \alpha)t_1 \equiv x \pmod p$ and $(\alpha - \beta)t_2 \equiv x \pmod p$. Then $\mathrm{dist_p}(\beta t, \alpha t) = x$ implies $t = t_1$ or $t = t_2$ (in other words, t_1 and t_2 are the only candidate values for t), so $c_x \leq 2$. ◀

Note that Boneh and Venkatesan's original argument for bounding $1 - A$ does not use Lemma 7, and instead involves showing

$$\left|\left\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist_p}(\beta t, \alpha t) \leq \frac{2p}{2^\mu}\right\}\right| = \left\lfloor p - \frac{2p}{2^\mu} \right\rfloor - \left\lceil \frac{2p}{2^\mu} \right\rceil \geq (p-1) \cdot \left(1 - \frac{5}{2^\mu}\right). \tag{2}$$

The equality in Equation (2) is a stronger statement than Lemma 7, but it is both harder to prove and harder to work with than the result given by Lemma 7. Specifically, dealing with the ± 1 terms that float around in ceiling/floor calculations is often unwieldy in a formal setting, especially when proving a result where these terms cannot be rounded away. Lemma 7 allows us to avoid precise ceiling/floor arithmetic while still giving a bound that is strong enough to prove Lemma 8 (`prob_A_helper` in our development) as follows.

▶ **Lemma 8.** *Let $\beta \in \{0, \ldots, p-1\}$, and suppose $\beta \neq \alpha$. Let $A \coloneqq \Pr_t[\mathrm{dist_p}(\beta t, \alpha t) > \frac{2p}{2^\mu}]$. Then $1 - A \leq \frac{5}{2^\mu}$.*

Proof of Lemma 8. Since t is sampled uniformly from $(\mathbb{Z}/p\mathbb{Z})^\times$, we have

$$A = \frac{|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist_p}(\beta t, \alpha t) > \frac{2p}{2^\mu}\}|}{|(\mathbb{Z}/p\mathbb{Z})^\times|} = 1 - \frac{|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist_p}(\beta t, \alpha t) \leq \frac{2p}{2^\mu}\}|}{p-1}.$$

Since $\beta \neq \alpha$, it follows that $\beta - \alpha$ is coprime to p (since $\beta < p$ and $\alpha < p$), so applying Lemma 7 with $B = \frac{2p}{2^\mu}$ gives

$$1 - A = \frac{|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist_p}(\beta t, \alpha t) \leq \frac{2p}{2^\mu}\}|}{p-1} \leq \frac{2 \cdot \frac{2p}{2^\mu}}{p-1} \leq \frac{5}{2^\mu},$$

where the last inequality follows from arithmetic (and our lower bound on p; see Section 2.2). ◀

We illustrate the lemma dependency tree and the high-level structure of the proof in Figure 2. We now turn to the formal statement of the HNP in Isabelle/HOL.

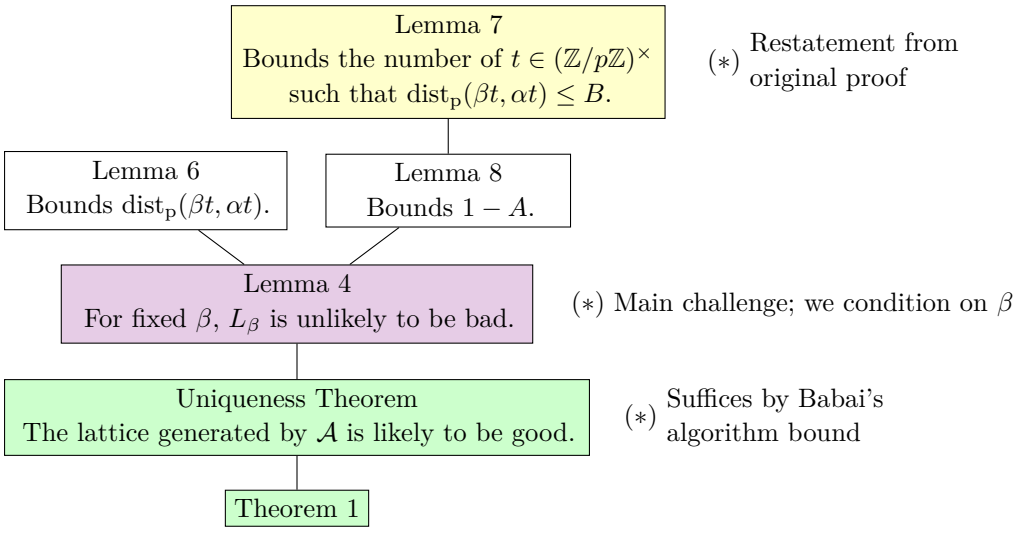

Figure 2 The lemma dependency tree. Lemma 4 (shaded purple) contains the essence of the argument and poses the primary challenge; Lemma 7 (shaded yellow) differs from the source material. The uniqueness theorem and Theorem 1 (shaded green) represent the top-level result (Theorem 1 follows directly from the uniqueness theorem and our bound in Babai's algorithm).

3.3 Top-level Statement

We state our top-level theorem within a locale. We begin with our *hnp_arith* locale, which fixes the ambient variables *p*, α, *n*, *k*, and *d* along with the necessary assumptions on each. This locale contains basic technical lemmas about the algebraic relationships between the ambient variables required throughout the rest of the proof.[9]

```
locale hnp_arith =
  fixes n α d p k :: nat
  assumes p: "prime p"
  assumes α: "α ∈ {1..<p}"
  assumes d: "d =  2 * ceiling (sqrt n)"
  assumes n: "n = ceiling (log 2 p)"
  assumes k: "k = ceiling (sqrt n) + ceiling (log 2 n)"
  assumes n_big: "961 < n"
```

We then introduce the *hnp* locale, which inherits from *hnp_arith* and *hnp_adversary*, and fixes the *msb_k* function along with its critical assumption.

```
locale hnp = hnp_arith n α d p k + hnp_adversary d p for n α d p k +
  fixes msb_k :: "nat ⇒ nat"
  assumes msb_k_dist: "⋀x. |int x - int (msb_k x)| < p / (2^k)"
```

Within this locale, we define $\mathcal{O}$ in terms of *msb_k*, as discussed in Section 2.2.

```
definition 𝒪 :: "nat ⇒ nat" where
  "𝒪 t = msb_k ((α * t) mod p)"
```

[9] The lemmas include results like $k + 1 < \log(p)$ and $\mu + 1 \leq k$.

We can now define *game*, which uses familiar *do* notation provided by the Isabelle/HOL standard libraries to formally capture the cryptographic "game" that our adversary must win with probability at least 1/2.

```
definition game :: "((nat × nat) list ⇒ nat) ⇒ bool pmf" where
  "game 𝒜' = do {
    ts ← replicate_pmf d (pmf_of_set {1..<p});
    return_pmf (α = 𝒜' (map (λt. (t, 𝒪 t)) ts))
  }"
```

Intuitively, one can think of *game* as a probabilistic procedure: using the probability mass function (*pmf*) monad,[10] *game* takes in an adversary $\mathcal{A}'$, uses the *replicate_pmf* function[11] to uniformly sample a list $(t_1, \ldots, t_d)$ from $\{1, \ldots, p-1\}^d$, passes $\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big)$ into the adversary $\mathcal{A}'$, and finally returns *True* if $\mathcal{A}'$ outputs α, and *False* otherwise. More formally, one can understand *game* as taking in an adversary $\mathcal{A}'$ and returning a probability mass function on the set $\{\textit{True}, \textit{False}\}$, where the probability given to *True* is the probability that $\mathcal{A}'\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big) = \alpha$. Thus, our top-level theorem is

```
theorem hnp_adversary_exists: "pmf (game 𝒜) True ≥ 1/2"
```

where $\mathcal{A}$ is the adversary defined in Section 3.1. This asserts that our explicit adversary $\mathcal{A}$ wins *game* with probability at least 1/2 (that is, $\mathcal{A}\big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\big) = \alpha$ for at least half of all lists $(t_1, \ldots, t_d) \in \{1, \ldots, p-1\}^d$).

4 Formalization Details

In this section, we consider the lessons learned from our formalization and emphasize some technical points of our formalization and proof strategies. We start by detailing our proof approach for Babai's nearest plane algorithm, focusing on where we chose to depart from the source material to aid in formalization (Section 4.1). Next, we clarify the definition of the MSB_k operator and present two concrete formal instantiations (Section 4.2). We then turn to some high-level discussion of our locale structure (Section 4.3) and finally discuss some low-level contributions to Isabelle's formal libraries (Section 4.4). Throughout our formalization, we enjoy and benefit from our choice of Isabelle/HOL [39], both for the strong automation afforded by Sledgehammer [40] and the very mature existing library of mathematics (including both the Isabelle/HOL standard libraries and the AFP).

4.1 Babai's Algorithm Details

We first give some details of our proof approach to formalizing Babai's nearest plane algorithm, followed by some discussion of the bounds we obtain.

We start with some intuition for how our proof of Babai's nearest plane algorithm proceeds. Let $\mathcal{B} = \{b_1, ..., b_d\}$ be a basis of the lattice L that is LLL-reduced with parameter $\delta \geq 4/3$, let $\tilde{\mathcal{B}} = \{\tilde{b}_1, ..., \tilde{b}_d\}$ be the orthogonalization of $\mathcal{B}$ computed by the Gram–Schmidt algorithm, and let $\hat{v}$ be the output of Babai's Algorithm. As $\tilde{\mathcal{B}}$ is orthogonal, $\|\hat{v} - u\|^2 = \sum_{i=1}^{d} \frac{\langle (\hat{v}-u), \tilde{b}_i \rangle^2}{\|\tilde{b}_i\|^2}$. Therefore, we may optimize $\|\hat{v} - u\|^2$ by optimizing each of the terms $\frac{\langle (\hat{v}-u), \tilde{b}_i \rangle^2}{\|\tilde{b}_i\|^2}$.

[10] Often called the Giry monad.

[11] All of *pmf*, *replicate_pmf*, and *return_pmf* are defined in the Isabelle/HOL standard libraries; see Section 4.4 for more discussion on *pmf* and *replicate_pmf*.

Babai's algorithm begins with $\hat{v}_0 \coloneqq \mathbf{0}$ and, at step i, defines $\hat{v}_i \coloneqq \hat{v}_{i-1} + z_i b_{n-i+1}$, where $z_i \in \mathbb{Z}$ minimizes $\langle (\hat{v}_i - u), \tilde{b}_{n-i+1} \rangle$. In particular, we can show that at step i, $\langle (\hat{v}_i - u), \tilde{b}_{n-i+1} \rangle$ can be brought below $\frac{1}{2}\|\tilde{b}_{n-i+1}\|^2$ in general and becomes 0 if $2\delta^{d/2} D < \|\tilde{b}_{n-i+1}\|$ (as long as $\mathcal{B}$ is LLL-reduced with parameter δ). Moreover, by nature of the Gram–Schmidt orthogonalization, the change-of-basis matrix from $\mathcal{B}$ to $\tilde{\mathcal{B}}$ is upper-triangular, so $\langle (\hat{v}_i - u), \tilde{b}_{n-i+1} \rangle = \langle (\hat{v} - u), \tilde{b}_{n-i+1} \rangle$, so the bound on each $\langle (\hat{v}_i - u), \tilde{b}_{n-i+1} \rangle$ carries through to the final output. Thus, we obtain

$$\|\hat{v} - u\|^2 = \sum_{i=1}^{d} \frac{\langle (\hat{v} - u), \tilde{b}_i \rangle^2}{\|\tilde{b}_i\|^2} \leq d \frac{1}{4} 4\delta^d D^2,$$

which yields $\|\hat{v} - u\| \leq \sqrt{d}\delta^{d/2} D$ as claimed. Additionally, because the algorithm begins with the lattice vector $\hat{v}_0 = \mathbf{0} \in L$, and in each step only adds integer multiples of vectors from $\mathcal{B}$ (the original basis of L), the final output vector $\hat{v}$ must be a lattice vector, as needed.

This is an algebraic analog of a geometric proof in lecture notes by Stephens-Davidowitz [42]. In our formalization, we make one modification involving the definition of $D \coloneqq \min\{\|v - u\| : v \in L\}$. It is intuitively clear that this D is well-defined (in particular, since $L \subseteq \mathbb{Z}^d$). One may try to formalize the well-definedness of D by picking a ball B centered at u such that $L \cap B \neq \varnothing$, arguing that $\min\{\|v - u\| : v \in L\} = \min\{\|v - u\| : v \in L \cap B\}$ if the RHS exists, and finally showing that the RHS exists by showing that $L \cap B$ is finite. However, this approach involves some geometric argumentation, which tends to be cumbersome to formalize. We instead sought a more algebraic path, starting with the alternative definition $D \coloneqq \inf\{\|v - u\| : v \in L\}$, making well-definedness of D immediate. With this new definition of D, we cannot obtain a vector $v_{\min}$ witnessing $\|v_{\min} - u\| = D$ (which is done in the standard proof approach), but we *can* obtain a vector v_ϵ such that $\|v_\epsilon - u\| \leq (1 + \epsilon)D$, for arbitrary $\epsilon > 0$. This $(1 + \epsilon)$ factor carries through the remainder of the proof, yielding $\|\hat{v} - u\| \leq (1 + \epsilon)\sqrt{n}(\sqrt{\delta})^d D$ for all $\epsilon > 0$, which implies $\|\hat{v} - u\| \leq \sqrt{n}(\sqrt{\delta})^d D$ as needed.

So, a key proof challenge is to obtain the v_ϵ for arbitrary $\epsilon > 0$. If $D \neq 0$, then $D < (1 + \epsilon)D$, meaning v_ϵ is yielded immediately by the definition of the infimum. However, if $D = 0$, then we must obtain v_ϵ where $\|v_\epsilon - u\| = 0$ (that is, $v_\epsilon = u$); this is a special case of the fact we had hoped to avoid formalizing, but the condition $D = 0$ enables the following argument that is much simpler than in the general case: If $D = 0$ but v_ϵ does not exist, then we may obtain distinct vectors in L that are arbitrarily close to u. This means that they are very close to each other, and hence their difference is a very short nonzero vector in L. Applying the lower bound on the sizes of short nonzero vectors in L from the formalized LLL algorithm [18] yields a contradiction, so v_ϵ must exist.

Intuitively, the $D = 0$ condition implies that if two vectors are both approximately D away from u, then those vectors are themselves close together (by the triangle inequality). In the $D > 0$ case, one can space out many far-apart vectors approximately on a sphere of radius D centered at u; bounding how many such far-apart vectors one can fit is tricky.

As Boneh and Venkatesan note [11], Babai's algorithm is commonly presented with a final bound of $\|\hat{v} - u\| \leq 2^{d/2} D$. They further note that with proper adjustment of constants in the LLL algorithm, the bound can be reduced to $2^{d/4.6} D$. Boneh and Venkatesan use a bound of $2^{d/4} D$ in their proof; notably, $2^{d/2} D$ is not strong enough for their proof. Our final bound lies between these two bounds: we prove that $\|\hat{v} - u\| \leq \sqrt{d}(4/3)^{d/2} D$. This is not as strong as the bound of $2^{d/4} D$, but is stronger than $2^{d/2} D$, and (in particular) is strong enough for the proof of Theorem 1.

4.2 Concrete MSB Functions

As our top-level result is stated for an arbitrary MSB_k function satisfying $|x - \mathrm{MSB}_k(x)| < p/2^k$, we formalize two concrete MSB_k functions.

We first formalize the MSB_k function given by Boneh and Venkatesan, namely that $\mathrm{MSB}_k(x)$ is the unique t such that $(t-1) \cdot p/2^k \leq x < t \cdot p/2^k$. However, this definition is inconsistent with the desired property $|x - \mathrm{MSB}_k(x)| < p/2^k$, and the details of reconciling this are omitted in the source material. To attain this, we set $\mathrm{MSB}_k(x) = \lfloor t \cdot p/2^k \rfloor - 1$, where t is as before. Note, this modification still does not achieve the $|x - \mathrm{MSB}_k(x)| < p/2^{k+1}$ bound as presented in the source material [11], but the $p/2^k$ bound suffices for the proof. As this definition is not stated in a computable way (though one could formulate an equivalent computable definition), a literal translation requires Isabelle/HOL's `THE` operator, which intuitively represents the unique object satisfying a particular property.[12] We provide the following literal translation, named `msb_p` to emphasize its direct dependence on `p`.

```
definition msb_p :: "nat ⇒ nat ⇒ nat ⇒ nat" where
  "msb_p p k x =
    (let t = THE t. (t - 1) * (p / 2^k) ≤ x ∧ x < t * p / 2^k
    in nat (floor (t * p / 2^k)) - 1)"
```

We prove that `msb_p` is well-defined by showing that there is indeed a unique t satisfying $(t-1) \cdot p/2^k \leq x < t \cdot p/2^k$.

As Boneh and Venkatesan's definition of MSB_k is non-computational and perhaps slightly esoteric, we formalize the following straightforward (computational) definition, `msb_kp1`, which literally computes the $k+1$ most significant bits of its input, using the `div` operation for integer division (note that the $+1$ is required to achieve the desired bound).

```
definition msb_kp1 :: "nat ⇒ nat ⇒ nat ⇒ nat" where
  "msb_kp1 k n x = (x div 2^(n - (k + 1))) * 2^(n - (k + 1))"
```

This is essentially an arithmetic description of a right-shift of x followed by a left-shift of x, both by $n - (k+1)$ bits, where we assume x is expressed in n bits. While the information we desire is captured just by the right shift, the left-shift is necessary to satisfy the desired bound (but adds no information to the oracle).

These two MSB_k functions differ slightly in value, though they express the same intention. The `msb_p` instantiation can be understood as partitioning $\mathbb{N}$ into intervals of length approximately $p/2^k$, and returning a value in the same interval as x. The `msb_kp1` instantiation can be understood as instead partitioning $\mathbb{N}$ into intervals of length $2^{n-(k+1)}$, and returning a value in the same interval as x. Noting that $p \approx 2^n$, one sees $p/2^k \approx 2^{n-k}$. Hence, `msb_kp1` gives around one additional bit of information about x.[13] This means instantiating the `hnp` locale with $\mathrm{MSB}_k =$ `msb_p` yields a slightly stronger result, since the associated oracle gives less information. Instantiating $\mathrm{MSB}_k =$ `msb_kp1`, on the other hand, gives a computational interpretation.

[12] If `t` ≡ `(THE x. P x)`, then we can prove `P t` by showing that there exists a unique `x` satisfying `P x`. If no such `x` exists, then the value of `t` is unspecified, meaning one can only prove things about `t` that hold for all values of the same type.

[13] The additional bit is required because $2^{n-1} < p < 2^n$, so taking intervals of length exactly 2^{n-k} is not quite good enough to satisfy the `msb_p_dist` assumption, but taking intervals of length $2^{n-(k+1)}$ is.

We verify that both of these MSB_k functions satisfy the *msb_k_dist* assumption in the *hnp* locale, which means that a different instantiation of the top-level *hnp_adversary_exists* theorem can be derived for each choice of MSB_k (though an instantiation using *msb_p* will not yield an executable oracle).

The computational instantiation MSB_k = *msb_kp1* is useful for demonstrating an execution of the adversary on a realistic example, since example oracle queries can be directly computed from the definition of *msb_kp1* via Isabelle/HOL's code generator. We include a short example of this in our AFP entry.

4.3 Locale Usage

In both our formalization of Babai's algorithm and the HNP, we use Isabelle locales [2] to fix ambient variables. For the HNP, we define the *hnp_adversary* locale, which fixes p and d, and is used to define the adversary $\mathcal{A}$ (see Section 3.1). Our main locale, *hnp*, inherits from this locale, and contains the rest of the proof. Note that if we defined $\mathcal{A}$ directly in the *hnp* locale, we could do the following:

```
fun cheating_𝒜 :: "(nat × nat) list ⇒ nat" where
  "cheating_𝒜 _ = α"
```

While the "cheating" in this example is obvious, we want to ensure a similar "leak" cannot happen in a subtler way. Our *hnp_adversary* locale avoids the need to manually inspect the definition of $\mathcal{A}$ for such a leak because α is not present in the context in which we define $\mathcal{A}$.

Still, as always with formal developments, the truly scrupulous reader must perform some manual inspection of the definitions in the top-level statement to be assured of its correctness. In our case, a skeptic must verify that we do not pass α to the *hnp_adversary* locale through some other variable name (either the p or d), which is easily checked by examining the locale inheritance in the *hnp* locale (discussed previously in Section 3.3). Moreover, it must be verified that in our definition of *game*, we do not pass more information about α to $\mathcal{A}$ than what is expressed in the statement of Theorem 1, which requires verifying our definition of $\mathcal{O}$. However, what is successfully avoided by our locale setup is the need to check our definition of $\mathcal{A}$ itself; since the main theorem is an existence proof, the scrupulous reader need only verify that the main theorem is correctly stated, and that α is not passed to $\mathcal{A}$ in the definition of *game* – it is not necessary to manually verify any aspect of the definition of the witness $\mathcal{A}$ itself.

4.4 PMF Helper Lemmas

The probability mass function (*pmf*) monad, defined in the Isabelle/HOL standard libraries, allows a user to express procedures involving probabilistic sampling using familiar *do* notation. For instance, a procedure which samples a value *x* from probability distribution *p* and checks if *P x* holds can be expressed as *check_event* ≡ *do {x ← p; return_pmf (P x)}*, and the probability that *check_event* returns *True* (i.e., the probability that *P x* holds) as *pmf check_event True*.

As we modeled our formalization on the game-playing style presented in the CryptHOL tutorial [30], we use the language of cryptographic games throughout our formalization (both in proofs and in our top-level statement) and thus frequently use *do* notation to express the probability of an event. Retrospectively, perhaps an explicitly measure-theoretic style would have been more standard – for instance, *measure_pmf.prob p {x. P x}* also expresses the probability of *P x* when *x* is sampled from *p*.

Indeed, we found that the existing lemma support for probabilistic reasoning is primarily focused on supporting such measure-theoretic reasoning, and is not as well-developed for working directly with the `pmf` monad. Consequently, our use of the `pmf` monad (specifically for expressing games with `do` notation) made it challenging to find relevant helper lemmas – for instance, we could not find a lemma directly stating that if `P x ⟶ Q x`, then

```
pmf (do {x ← p; return_pmf P x}) True ≤ pmf (do {x ← p; return_pmf Q x}) True
```

Yet, there is a lemma for the measure-theoretic analog, i.e. `{x. P x} ⊆ {x. Q x}` implies

```
measure_pmf.prob p {x. P x} ≤ measure_pmf.prob p {x. Q x}
```

If we chose to restrict our statements and proofs to such measure-theoretic expressions, or alternatively translated back and forth between measure-theoretic expressions and monadic `pmf` expressions, then the existing measure-theoretic lemmas could have been sufficient for our purposes. However, we believe that the `pmf` monad provides a valuable probabilistic language – in general, cryptographic games can be quite complex, which makes the intuitive expressivity of the `do` notation very valuable – that deserves a suite of specialized helper lemmas analogous to the existing measure-theoretic ones.

We establish helper lemmas which lift some basic `measure_pmf.prob` lemmas to apply directly to monadic `pmf` constructs. While most of these helper lemmas are not too challenging to prove (they typically involve translating a monadic `pmf` expression into an explicitly measure-theoretic form, applying the appropriate lemmas, then translating the result back to the `pmf` form), they support reasoning directly about cryptographic games expressed with `do` notation, which is convenient in manual proofs and also supports Sledgehammer's automation. We believe our lemmas may be of use to future Isabelle/HOL users who wish to work extensively with probabilistic expressions involving `do` notation and the `pmf` monad.

We discuss one particularly notable `pmf` helper lemma here. Throughout our informal proof, we have assumed a sequence of t_i's that are uniformly, independently sampled from a finite set; we formally achieve this using the `replicate_pmf` function, defined in Isabelle/HOL's probability library (`HOL-Probability`) as follows:

```
primrec replicate_pmf :: "nat ⇒ 'a pmf ⇒ 'a list pmf" where
  "replicate_pmf 0 _ = return_pmf []"
| "replicate_pmf (Suc n) p =
    do {x ← p; xs ← replicate_pmf n p; return_pmf (x#xs)}"
```

Intuitively, if `p` is a probability distribution over elements of type `'a`, then `replicate_pmf n p` gives a probability distribution over elements of type `list 'a` of length `n`. We formally prove that `replicate_pmf` matches our intuitive notion of repeated independent samples as follows:

```
lemma replicate_pmf_events:
  fixes p :: "'a pmf"
  fixes n :: nat
  fixes E :: "nat ⇒ 'a ⇒ bool"
  fixes A :: "nat ⇒ real"
  assumes "∀ i < n. pmf (do {x ← p; return_pmf (E i x)}) True = A i"
  defines "events_pmf ≡
    (do {xs ← replicate_pmf n p; return_pmf (∀ i < n. E i (xs!i))})"
  shows "pmf events_pmf True = (∏ i<n. A i)"
```

Intuitively, this lemma states that the probability of a sequence of events occuring is simply the product of the probabilities of those events. In our formalization, we use this lemma to go from probability $1 - A$ (for one event) to probability $(1 - A)^d$ (for d events) in the proof of Lemma 4. Furthermore, we use this lemma to connect `replicate_pmf` to the existing `indep_events` definition provided by the Isabelle/HOL probability library:

```
lemma replicate_pmf_indep:
  fixes p :: "'a pmf"
  fixes n :: nat
  fixes E :: "nat ⇒ 'a ⇒ bool"
  defines "rp ≡ replicate_pmf n p"
  shows "prob_space.indep_events rp (λi. {xs ∈ rp. E i (xs!i)}) {..<n}"
```

5 Related Work

There is growing interest in formalizing cryptographic constructs in interactive theorem provers. In Isabelle/HOL, the framework CryptHOL [6, 30] was developed to assist engineers in formalizing *security properties* of cryptographic constructions; in particular, the framework supports formalizing cryptographic *games*, which allow one to model adversaries and their interactions with encryption algorithms. CryptHOL has already enjoyed wide usage: existing Isabelle/HOL AFP developments which build on CryptHOL include a formal implementation and verification of the post-quantum Kyber algorithm [25, 23], a formalization of commitment schemes and Sigma protocols [14, 17, 16], and a formalization of multi-party communication protocols [15]. Also in the AFP is an extension of CryptHOL [5, 29] that implements the constructive cryptography paradigm [31, 32], allowing for abstract, composable security statements that enable more reusable proofs.

We adopt the style of cryptographic games as described in the CryptHOL tutorial [30], but we do not require the more sophisticated machinery that CryptHOL provides (e.g. random oracles with state, failure and failure handling, etc.). As our cryptographic `game` can be stated entirely using definitions from the standard Isabelle/HOL probability library, we do not depend on CryptHOL.

Other related work is cryptography-adjacent. For example, the closest vector problem (CVP) has been formalized and proved to be NP-hard [26]. Our work on formalizing Babai's nearest plane algorithm, which gives a polynomial-time approximation algorithm for the CVP, is related, although we do not explicitly connect our work to this existing development.

For our formalization of Babai's algorithm, we build directly on the LLL basis reduction algorithm, which reduces a given lattice basis to a short, "nearly orthogonal" basis [28] and has been formalized in Isabelle/HOL [12, 43, 18].

While Isabelle/HOL is the natural choice for our work (since, to our knowledge, the LLL algorithm has not been formalized in any other theorem provers), there are many other tools which provide support for formally verifying game-based security proofs, including EasyCrypt [3] and the CertiCrypt Rocq framework [4]. Also, the CryptAttackTester framework specializes in formally quantifying the effectiveness of cryptographic attacks [7]. Much cryptographic formalization work has been done in these (and other) tools.

6 Conclusion and Future Work

We formalize Boneh and Venkatesan's seminal development of the hidden number problem [11] in Isabelle/HOL. To our knowledge, this is the first formalization of the HNP, and involved developing a clearly organized proof blueprint which rigorizes the definitions of the adversary

and the oracle and fills in previously omitted proof details. We also formalize the relevant instance of Babai's nearest plane algorithm [1, 42], and contribute some lemmas which support directly reasoning about formal game-based constructs. The HNP is used both in security proofs and in cryptographic attacks for well-known protocols, so our formalization lays the foundation for future formal cryptographic developments.

Specifically, as we view formalizing cryptographic attacks to be a necessary part of moving cryptography into the formal sphere, it would be interesting to formalize attacks on DSA and ECDSA. Our formalizations of the HNP's adversary and of Babai's nearest plane algorithm are executable, which may be relevant for future formalizations of instances of cryptographic attacks, e.g. [41]. Further, it would be interesting to formalize some variants of the HNP, such as the Extended HNP [22], the Modular Inverse HNP [9], and the Elliptic Curve HNP [10] (which was developed to analyze the security of Elliptic Curve Diffie–Hellman).

Additionally, while our development focuses on establishing the *correctness* of the adversary, formalizing a time-complexity analysis of the adversary's computations to establish that the adversary runs in polynomial time would be interesting future work. This would largely require formalizing that Babai's nearest plane algorithm runs in polynomial time. This may be tractable in the near future, as the formal development of the LLL algorithm already includes a formalized complexity analysis [43].

References

1 László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Comb.*, 6(1):1–13, 1986. doi:10.1007/BF02579403.

2 Clemens Ballarin. Locales: A module system for mathematical theories. *J. Autom. Reason.*, 52(2):123–153, 2014. doi:10.1007/s10817-013-9284-7.

3 Cécile Baritel-Ruet. *Formal Security Proofs of Cryptographic Standards : A necessity achieved using EasyCrypt*. Theses, Université Côte d'Azur, October 2020. URL: https://theses.hal.science/tel-03150443.

4 Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. Formal certification of code-based cryptographic proofs. *SIGPLAN Not.*, 44(1):90–101, January 2009. doi:10.1145/1594834.1480894.

5 David Basin, Andreas Lochbihler, Ueli Maurer, and S Reza Sefidgar. Abstract modeling of system communication in constructive cryptography using CryptHOL. In *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*, pages 1–16. IEEE, 2021. doi:10.1109/CSF51468.2021.00047.

6 David A. Basin, Andreas Lochbihler, and S. Reza Sefidgar. CryptHOL: Game-based proofs in higher-order logic. *J. Cryptol.*, 33(2):494–566, 2020. doi:10.1007/S00145-019-09341-Z.

7 Daniel J. Bernstein and Tung Chou. CryptAttackTester: high-assurance attack analysis. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO*, volume 14925 of *LNCS*, pages 141–182. Springer, 2024. doi:10.1007/978-3-031-68391-6_5.

8 Sage Binder, Eric Ren, and Katherine Kosaian. The hidden number problem. *Archive of Formal Proofs*, June 2025. , Formal proof development. URL: https://isa-afp.org/entries/Hidden_Number_Problem.html.

9 Dan Boneh, Shai Halevi, and Nick Howgrave-Graham. The modular inversion hidden number problem. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *LNCS*, pages 36–51. Springer, 2001. doi:10.1007/3-540-45682-1_3.

10 Dan Boneh and Igor E. Shparlinski. On the unpredictability of bits of the elliptic curve Diffie–Hellman scheme. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 201–212. Springer, 2001. doi:10.1007/3-540-44647-8_12.

11 Dan Boneh and Ramarathnam Venkatesan. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 129–142. Springer, 1996. doi:10.1007/3-540-68697-5_11.

12 Ralph Bottesch, Max W. Haslbeck, and René Thiemann. A verified efficient implementation of the LLL basis reduction algorithm. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR*, volume 57 of *EPiC Series in Computing*, pages 164–180. EasyChair, 2018. doi:10.29007/XWWH.

13 Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In Ian Goldberg and Tyler Moore, editors, *FC*, volume 11598 of *LNCS*, pages 3–20. Springer, 2019. doi:10.1007/978-3-030-32101-7_1.

14 David Butler. *Formalising cryptography using CryptHOL*. PhD thesis, University of Edinburgh, UK, 2020. doi:10.7488/ERA/510.

15 David Butler, David Aspinall, and Adrià Gascón. Formalising oblivious transfer in the semi-honest and malicious model in CryptHOL. In Jasmin Blanchette and Catalin Hritcu, editors, *CPP*, pages 229–243. ACM, 2020. doi:10.1145/3372885.3373815.

16 David Butler and Andreas Lochbihler. Sigma protocols and commitment schemes. *Archive of Formal Proofs*, October 2019. , Formal proof development. URL: https://isa-afp.org/entries/Sigma_Commit_Crypto.html.

17 David Butler, Andreas Lochbihler, David Aspinall, and Adrià Gascón. Formalising $\sum$-protocols and commitment schemes using CryptHOL. *J. Autom. Reason.*, 65(4):521–567, 2021. doi:10.1007/S10817-020-09581-W.

18 Jose Divasón, Sebastiaan J. C. Joosten, René Thiemann, and Akihisa Yamada. A Verified Factorization Algorithm for Integer Polynomials with Polynomial Complexity. *Archive of Formal Proofs*, February 2018. , Formal proof development. URL: https://isa-afp.org/entries/LLL_Factorization.html.

19 Shuqin Fan, Wenbo Wang, and Qingfeng Cheng. Attacking OpenSSL implementation of ECDSA with a few signatures. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *SIGSAC*, pages 1505–1515. ACM, 2016. doi:10.1145/2976749.2978400.

20 Hugo Férée, Samuel Hym, Micaela Mayero, Jean-Yves Moyen, and David Nowak. Formal proof of polynomial-time complexity with quasi-interpretations. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2018, pages 146–157, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3167097.

21 Nadia Heninger and Keegan Ryan. The hidden number problem with small unknown multipliers: Cryptanalyzing MEGA in six queries and other applications. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC*, volume 13940 of *LNCS*, pages 147–176. Springer, 2023. doi:10.1007/978-3-031-31368-4_6.

22 Martin Hlavác and Tomás Rosa. Extended hidden number problem and its cryptanalytic applications. In Eli Biham and Amr M. Youssef, editors, *SAC*, volume 4356 of *LNCS*, pages 114–133. Springer, 2006. doi:10.1007/978-3-540-74462-7_9.

23 Katharina Kreuzer. CRYSTALS-Kyber_Security. *Archive of Formal Proofs*, December 2023. , Formal proof development. URL: https://isa-afp.org/entries/CRYSTALS-Kyber_Security.html.

24 Katharina Kreuzer. Hardness of Lattice Problems. *Archive of Formal Proofs*, february 2023. , Formal proof development. URL: https://isa-afp.org/entries/CVP_Hardness.html.

25 Katharina Kreuzer. Verification of correctness and security properties for CRYSTALS-KYBER. *IACR Cryptol. ePrint Arch.*, page 87, 2023. URL: https://eprint.iacr.org/2023/087.

26 Katharina Kreuzer and Tobias Nipkow. Verification of NP-hardness reduction functions for exact lattice problems. In Brigitte Pientka and Cesare Tinelli, editors, *CADE*, volume 14132 of *LNCS*, pages 365–381. Springer, 2023. doi:10.1007/978-3-031-38499-8_21.

27 Kim Laine and Kristin E. Lauter. Key recovery for LWE in polynomial time. *IACR Cryptol. ePrint Arch.*, page 176, 2015. URL: http://eprint.iacr.org/2015/176.

28 Arjen Lenstra, Hendrik Lenstra, and Lovász László. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261, December 1982. doi:10.1007/BF01457454.

29 Andreas Lochbihler and S. Reza Sefidgar. Constructive cryptography in HOL. *Archive of Formal Proofs*, December 2018. , Formal proof development. URL: https://isa-afp.org/entries/Constructive_Cryptography.html.

30 Andreas Lochbihler and S. Reza Sefidgar. A tutorial introduction to CryptHOL. *IACR Cryptol. ePrint Arch.*, page 941, 2018. URL: https://eprint.iacr.org/2018/941.

31 Ueli Maurer. Constructive cryptography–a new paradigm for security definitions and proofs. In *Joint Workshop on Theory of Security and Applications*, pages 33–56. Springer, 2011. doi:10.1007/978-3-642-27375-9_3.

32 Ueli Maurer and Renato Renner. From indifferentiability to constructive cryptography (and back). In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part I 14*, pages 3–24. Springer, 2016. doi:10.1007/978-3-662-53641-4_1.

33 Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. Raccoon attack: Finding and exploiting most-significant-bit-oracles in TLS-DH(E). In Michael D. Bailey and Rachel Greenstadt, editors, *USENIX*, pages 213–230. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/merget.

34 Gabrielle De Micheli, Rémi Piau, and Cécile Pierrot. A tale of three signatures: Practical attack of ECDSA with wNAF. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT*, volume 12174 of *LNCS*, pages 361–381. Springer, 2020. doi:10.1007/978-3-030-51938-4_18.

35 Elke De Mulder, Michael Hutter, Mark E. Marson, and Peter Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. *J. Cryptogr. Eng.*, 4(1):33–45, 2014. doi:10.1007/S13389-014-0072-Z.

36 Phong Q Nguyen. The dark side of the hidden number problem: Lattice attacks on DSA. In *Cryptography and Computational Number Theory*, pages 321–330. Springer, 2001.

37 Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the Digital Signature Algorithm with partially known nonces. *J. Cryptol.*, 15(3):151–176, 2002. doi:10.1007/S00145-002-0021-3.

38 Phong Q. Nguyen and Igor E. Shparlinski. The insecurity of the Elliptic Curve Digital Signature Algorithm with partially known nonces. *Des. Codes Cryptogr.*, 30(2):201–217, 2003. doi:10.1023/A:1025436905711.

39 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. doi:10.1007/3-540-45949-9.

40 Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In Geoff Sutcliffe, Stephan Schulz, and Eugenia Ternovska, editors, *IWIL*, volume 2 of *EPiC Series in Computing*, pages 1–11. EasyChair, 2010. doi:10.29007/36DT.

41 Keegan Ryan. Return of the hidden number problem. A widespread and novel key extraction attack on ECDSA and DSA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):146–168, 2019. doi:10.13154/TCHES.V2019.I1.146-168.

42 Noah Stephens-Davidowitz. Lecture 5: CVP and Babai's Algorithm, august 2016. New York University, Lattices Mini Course. URL: https://www.noahsd.com/mini_lattices/05__babai.pdf.

43 René Thiemann, Ralph Bottesch, Jose Divasón, Max W. Haslbeck, Sebastiaan J. C. Joosten, and Akihisa Yamada. Formalizing the LLL basis reduction algorithm and the LLL factorization algorithm in Isabelle/HOL. *J. Autom. Reason.*, 64(5):827–856, 2020. doi:10.1007/S10817-020-09552-1.