Formalizing the Hidden Number Problem in Isabelle/HOL

Sage Binder

University of Iowa

2025-09-29

Joint work with Eric Ren and Katherine Kosaian

The Hidden Number Problem

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
    - Originally for DH bit-security.

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
  - Originally for DH bit-security.
  - Now an important paradigm in cryptography.

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
  - Originally for DH bit-security.
  - Now an important paradigm in cryptography.
  - Many variants for different protocols.

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
  - Originally for DH bit-security.
  - Now an important paradigm in cryptography.
  - Many variants for different protocols.

Broad idea

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
  - Originally for DH bit-security.
  - Now an important paradigm in cryptography.
  - Many variants for different protocols.

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
  - Originally for DH bit-security.
  - Now an important paradigm in cryptography.
  - Many variants for different protocols.

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?

The Hidden Number Problem

- Introduced by Boneh and Venkatesan in 1996.
  - Originally for DH bit-security.
  - Now an important paradigm in cryptography.
  - Many variants for different protocols.

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Talk outline

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Talk outline

- Brief DH review

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Talk outline

- Brief DH review
- Intuition DH to HNP reduction

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Talk outline

- Brief DH review
- Intuition DH to HNP reduction
- HNP formal statement

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Talk outline

- Brief DH review
- Intuition DH to HNP reduction
- HNP formal statement
- High-level proof idea

Broad idea

- Suppose $\mathcal{O}(t)$ yields some information about $\alpha t \bmod p$.
- Can we recover α given access to $\mathcal{O}$?
- Yes, with high probability.

Talk outline

- Brief DH review
- Intuition DH to HNP reduction
- HNP formal statement
- High-level proof idea
- Formalization details

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^{\times}$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^{a} and g^{b}

Alice	Adversary	Bob
g	g	g

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a		

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a		b

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a		b
g^a		

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a		b
g^a		g^b

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a	g^a	b
g^b	g^b	g^a

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a	g^a	b
g^b	g^b	g^a
$(g^b)^a = g^{ab}$		$(g^a)^b = g^{ab}$

Diffie-Hellman Key Exchange

- Let g be a (public) generator of a group (say $(\mathbb{Z}/p\mathbb{Z})^\times$).
- Goal: establish shared secret between Alice and Bob.
- Idea: g^{ab} hard to compute from g^a and g^b

Alice	Adversary	Bob
g	g	g
a	g^a	b
g^b	g^b	g^a
$(g^b)^a = g^{ab}$	$\cancel{g^{ab}}$	$(g^a)^b = g^{ab}$

Diffie-Hellman Key Exchange

- An adversary observing communication must solve the discrete log problem to recover g^{ab} from intercepted information.

Alice	Adversary	Bob
g	g	g
a	g^a	b
g^b	g^b	g^a
$(g^b)^a = g^{ab}$	$\cancel{g^{ab}}$	$(g^a)^b = g^{ab}$

Diffie-Hellman Key Exchange

- An adversary observing communication must solve the discrete log problem to recover g^{ab} from intercepted information.
- We assume discrete log is hard...

Diffie-Hellman Key Exchange

- An adversary observing communication must solve the discrete log problem to recover g^{ab} from intercepted information.
- We assume discrete log is hard...
- Under this assumption: is it hard to gain even partial information?

Diffie-Hellman Key Exchange

- An adversary observing communication must solve the discrete log problem to recover g^{ab} from intercepted information.
- We assume discrete log is hard...
- Under this assumption: is it hard to gain even partial information?
- Yes!

Diffie-Hellman Key Exchange

- An adversary observing communication must solve the discrete log problem to recover g^{ab} from intercepted information.
- We assume discrete log is hard...
- Under this assumption: is it hard to gain even partial information?
- Yes! Idea: compute g^{ab} given a bit-leaking oracle.

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.
- $A(g^{a+x}, g^b) = \mathrm{MSB}_k(\mathrm{DH}(g^{a+x}, g^b))$

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.
- $A(g^{a+x}, g^b) = \mathrm{MSB}_k(\mathrm{DH}(g^{a+x}, g^b))$
  $= \mathrm{MSB}_k((g^{ab}) \cdot (g^b)^x)$

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.
- $A(g^{a+x}, g^b) = \mathrm{MSB}_k(\mathrm{DH}(g^{a+x}, g^b))$
  $= \mathrm{MSB}_k((g^{ab}) \cdot (g^b)^x)$
  $= \mathrm{MSB}_k(\alpha h^x)$

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.
- $A(g^{a+x}, g^b) = \mathrm{MSB}_k(\mathrm{DH}(g^{a+x}, g^b))$

  $= \mathrm{MSB}_k((g^{ab}) \cdot (g^b)^x)$

  $= \mathrm{MSB}_k(\alpha h^x)$

  $= \mathrm{MSB}_k(\alpha t)$

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.
- $A(g^{a+x}, g^b) = \mathrm{MSB}_k(\mathrm{DH}(g^{a+x}, g^b))$

$$= \mathrm{MSB}_k((g^{ab}) \cdot (g^b)^x)$$

$$= \mathrm{MSB}_k(\alpha h^x)$$

$$= \mathrm{MSB}_k(\alpha t)$$

$$=: \mathcal{O}(t)$$

The Hidden Number Problem

- $\mathrm{DH}(g^x, g^y) = g^{xy}$
- Suppose $A(g^x, g^y) = \mathrm{MSB}_k(\mathrm{DH}(g^x, g^y))$.
  - Roughly, MSB_k gives k most significant bits.
- $A(g^{a+x}, g^b) = \mathrm{MSB}_k(\mathrm{DH}(g^{a+x}, g^b))$
  $= \mathrm{MSB}_k((g^{ab}) \cdot (g^b)^x)$
  $= \mathrm{MSB}_k(\alpha h^x)$
  $= \mathrm{MSB}_k(\alpha t)$
  $=: \mathcal{O}(t)$
- Main theorem: we can recover α given $\mathcal{O}$.

Theorem 1

Theorem 1

- Let α be hidden.

Theorem 1

- Let α be hidden. Let $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.

Theorem 1

- Let α be hidden. Let $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- Let $t_1, \ldots, t_d$ be sampled uniformly from $(\mathbb{Z}/p\mathbb{Z})^\times$.

Theorem 1

- Let α be hidden. Let $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- Let $t_1, \ldots, t_d$ be sampled uniformly from $(\mathbb{Z}/p\mathbb{Z})^\times$.

There exists an adversary $\mathcal{A}$ such that

Theorem 1

- Let α be hidden. Let $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- Let $t_1, \ldots, t_d$ be sampled uniformly from $(\mathbb{Z}/p\mathbb{Z})^\times$.

There exists an adversary $\mathcal{A}$ such that

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

Theorem 1

- Let α be hidden. Let $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- Let $t_1, \ldots, t_d$ be sampled uniformly from $(\mathbb{Z}/p\mathbb{Z})^\times$.

There exists an adversary $\mathcal{A}$ such that

$$\Pr\left[\mathcal{A}\Big((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d))\Big) = \alpha\right] \geq 1/2.$$

Can recover α given access to random oracle $\mathcal{O}$ in poly time.

Theorem 1

- Let α be hidden. Let $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- Let $t_1, \ldots, t_d$ be sampled uniformly from $(\mathbb{Z}/p\mathbb{Z})^\times$.

There exists an adversary $\mathcal{A}$ such that

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

Can recover α given access to random oracle $\mathcal{O}$ in poly time.

- $961 < n := \lceil \log(p) \rceil$
- $d := 2 \cdot \lceil \sqrt{n} \rceil$
- $k := \lceil \sqrt{n} \rceil + \lceil \log(n) \rceil$

Proof idea

- $u := (\mathcal{O}(t_1), \ldots, \mathcal{O}(t_d), 0)$

Proof idea

- $u := (\mathcal{O}(t_1), \ldots, \mathcal{O}(t_d), 0)$
- Recall $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.

Proof idea

- $u := (\mathcal{O}(t_1), \ldots, \mathcal{O}(t_d), 0)$
- Recall $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- $\mathcal{O}(t_i) \approx \alpha t_i - mp$ for some m.

Proof idea

- $u := (\mathcal{O}(t_1), \ldots, \mathcal{O}(t_d), 0)$
- Recall $\mathcal{O}(t) = \mathrm{MSB}_k(\alpha t \bmod p)$.
- $\mathcal{O}(t_i) \approx \alpha t_i - mp$ for some m.
- Idea: Approximate α using lattice methods.

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

- Build lattice L from basis:

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

- Build lattice L from basis:
  - $x_i = (\underbrace{0, \ldots, p, \ldots, 0}_{i\text{th component} = p}, 0)$ for $i \leq d$;

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

- Build lattice L from basis:
  - $x_i = (\underbrace{0, \ldots, p, \ldots, 0}_{i\text{th component} = p}, 0)$ for $i \leq d$;
  - $x_{d+1} = (t_1, \ldots, t_d, 1/p)$.

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta/p)$$

- Build lattice L from basis:
  - $x_i = (\underbrace{0, \ldots, p, \ldots, 0}_{i\text{th component} = p}, 0)$ for $i \leq d$;
  - $x_{d+1} = (t_1, \ldots, t_d, 1/p)$.

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta/p)$$

- Build lattice L from basis:
  - $x_i = (\underbrace{0, \ldots, p, \ldots, 0}_{i\text{th component} = p}, 0)$ for $i \leq d$;
  - $x_{d+1} = (t_1, \ldots, t_d, 1/p)$.

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod{p}$ with high probability

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$
$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta / p)$$

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod p$ with high probability

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta / p)$$

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod{p}$ with high probability

- Therefore, find v minimizing $|v - u|$.

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$

$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta / p)$$

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod{p}$ with high probability

- Therefore, find v minimizing $|v - u|$.
- This is the closest vector problem which is NP-hard...

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$
$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta / p)$$

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod{p}$ with high probability

- Therefore, find v minimizing $|v - u|$.
- This is the closest vector problem which is NP-hard...
- But we can approximate it!

Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$
$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta/p)$$

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod p$ with high probability

- Therefore, find v minimizing $|v - u|$.
- This is the closest vector problem which is NP-hard...
- But we can approximate it!
  - With Babai's nearest plane algorithm.

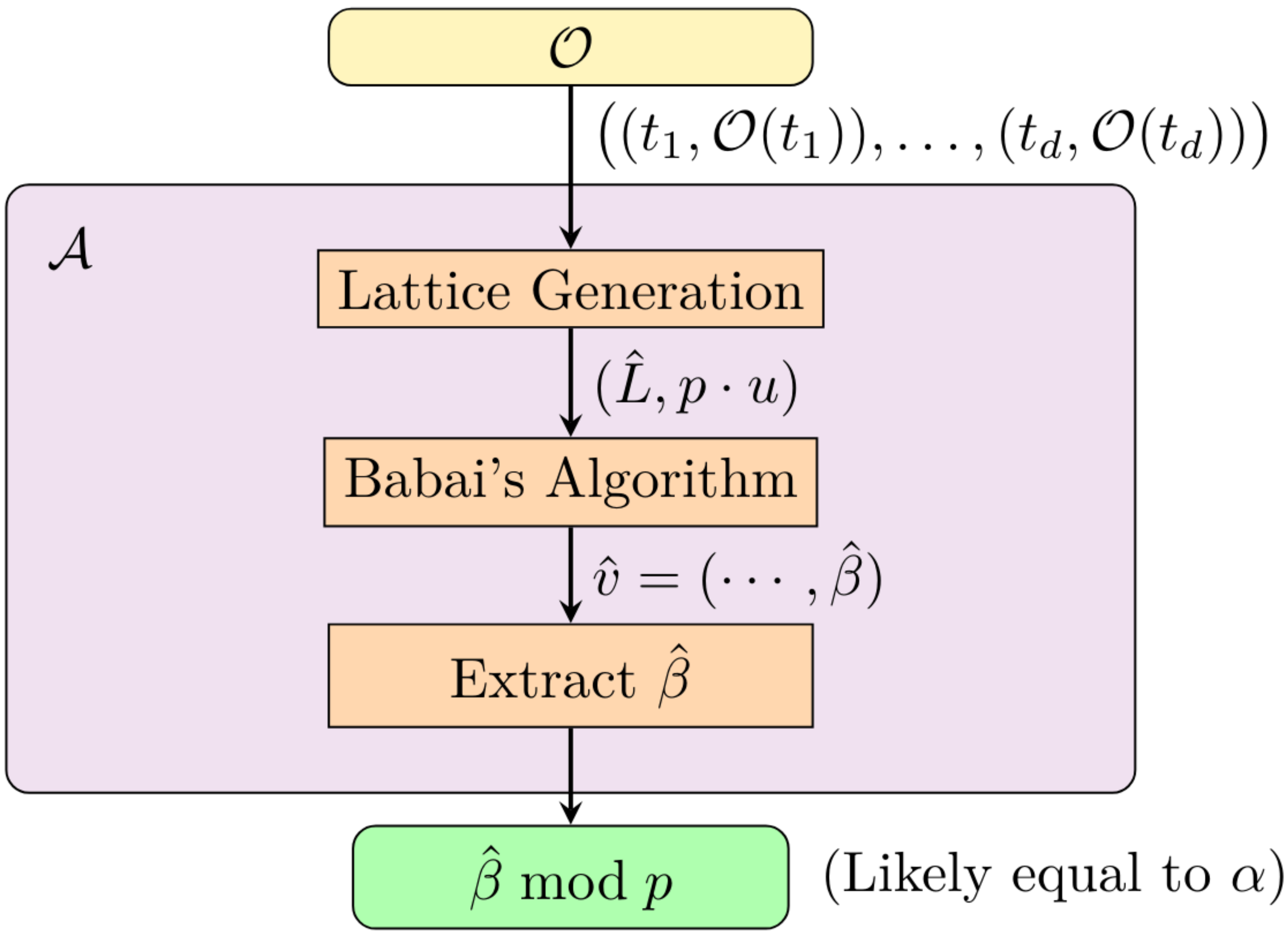
Proof idea

$$u \approx (\alpha t_1 - m_1 p, \ldots, \alpha t_d - m_d p, 0)$$
$$L \ni v = (\beta t_1 - b_1 p, \ldots, \beta t_d - b_d p, \beta / p)$$

If $|v - u|$ is small, then $\beta \equiv \alpha \pmod{p}$ with high probability

- Therefore, find v minimizing $|v - u|$.
- This is the closest vector problem which is NP-hard...
- But we can approximate it!
  - With Babai's nearest plane algorithm.
  - Finds $v \in L$ with $|u - v|$ close to optimal.

Adversary



$\mathcal{O}$
$((t_1, \mathcal{O}(t_1)), \ldots, (t_d, \mathcal{O}(t_d)))$
$\mathcal{A}$
Lattice Generation
$(\hat{L}, p \cdot u)$
Babai's Algorithm
$\hat{v} = (\cdots, \hat{\beta})$
Extract $\hat{\beta}$
$\hat{\beta} \bmod p$
(Likely equal to α)

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod p$ are far from u.

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod{p}$ are far from u.

- Need to formalize Babai's algorithm.

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod p$ are far from u.

- Need to formalize Babai's algorithm.
- Need formal probabilistic reasoning.

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod p$ are far from u.

- Need to formalize Babai's algorithm.
- Need formal probabilistic reasoning.
- Need to formalize bounds (e.g. "for large enough n") and make arithmetic precise.

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod p$ are far from u.

- Need to formalize Babai's algorithm.
  - We build on existing LLL formalization (many authors including René Thiemann).
- Need formal probabilistic reasoning.

- Need to formalize bounds (e.g. "for large enough n") and make arithmetic precise.

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod p$ are far from u.

- Need to formalize Babai's algorithm.
  - We build on existing LLL formalization (many authors including René Thiemann).
- Need formal probabilistic reasoning.
  - We build on existing PMF library.
- Need to formalize bounds (e.g. "for large enough n") and make arithmetic precise.

Formalization needs

- Main goal:
  With probability at least $1/2$, all $v = (\ldots, \beta/p) \in L$ with $\beta \not\equiv \alpha \pmod p$ are far from u.

- Need to formalize Babai's algorithm.
  - We build on existing LLL formalization (many authors including René Thiemann).
- Need formal probabilistic reasoning.
  - We build on existing PMF library.
- Need to formalize bounds (e.g. "for large enough n") and make arithmetic precise.
  - Isabelle/HOL automation very valuable.

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

- We "algebraicize" a geometry proof by Stephens-Davidowitz.

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

- We "algebraicize" a geometry proof by Stephens-Davidowitz.
- Let $D := \min\{|v - u| : v \in L\}$.

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

- We "algebraicize" a geometry proof by Stephens-Davidowitz.
- Let $D := \min\{|v - u| : v \in L\}$.
- Can prove Babai output v satisfies $|v - u| \leq 2^{\dim(L)/4.6} D$.

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

- We "algebraicize" a geometry proof by Stephens-Davidowitz.
- Let $D := \min\{|v - u| : v \in L\}$.
- Can prove Babai output v satisfies $|v - u| \leq 2^{\dim(L)/4.6} D$.
- Literature commonly gives $2^{\dim(L)/2} D$.

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

- We "algebraicize" a geometry proof by Stephens-Davidowitz.
- Let $D := \min\{|v - u| : v \in L\}$.
- Can prove Babai output v satisfies $|v - u| \le 2^{\dim(L)/4.6} D$.
- Literature commonly gives $2^{\dim(L)/2} D$.
  - Not strong enough for final HNP result.

Formalizing Babai's nearest plane algorithm

<u>Babai's algorithm bounds:</u>

- We "algebraicize" a geometry proof by Stephens-Davidowitz.
- Let $D := \min\{|v - u| : v \in L\}$.
- Can prove Babai output v satisfies $|v - u| \leq 2^{\dim(L)/4.6} D$.
- Literature commonly gives $2^{\dim(L)/2} D$.
  - Not strong enough for final HNP result.
- Boneh and Venkatesan use $2^{\dim(L)/4} D$.

Formalizing Babai's nearest plane algorithm

Babai's algorithm bounds:

- We "algebraicize" a geometry proof by Stephens-Davidowitz.
- Let $D := \min\{|v - u| : v \in L\}$.
- Can prove Babai output v satisfies $|v - u| \leq 2^{\dim(L)/4.6} D$.
- Literature commonly gives $2^{\dim(L)/2} D$.
  - Not strong enough for final HNP result.
- Boneh and Venkatesan use $2^{\dim(L)/4} D$.
- We formalize $\sqrt{\dim(L)}(4/3)^{\dim(L)/2} D$, which suffices.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL **`min`** only for finite sets.
- We use **`inf`** instead: $D := \inf\{|v - u| : v \in L\}$...

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.
- We use `inf` instead: $D := \inf\{|v - u| : v \in L\}$...
- But still need $v_0 \in L$ witnessing $|v_0 - u| = D$.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.
- We use `inf` instead: $D := \inf\{|v - u| : v \in L\}$...
- But still need $v_0 \in L$ witnessing $|v_0 - u| = D$.
- Intuitively true (lattice is discrete); annoying to formalize

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.
- We use `inf` instead: $D := \inf\{|v - u| : v \in L\}$...
- But still need $v_0 \in L$ witnessing $|v_0 - u| = D$.
- Intuitively true (lattice is discrete); annoying to formalize
- Instead, we obtain v_ϵ where $|v_\epsilon - u| \leq (1 + \epsilon)D$.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.
- We use **`inf`** instead: $D := \inf\{|v - u| : v \in L\}$...
- But still need $v_0 \in L$ witnessing $|v_0 - u| = D$.
- Intuitively true (lattice is discrete); annoying to formalize
- Instead, we obtain v_ϵ where $|v_\epsilon - u| \leq (1 + \epsilon)D$.
  - For arbitrary $\epsilon > 0$.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.
- We use `inf` instead: $D := \inf\{|v - u| : v \in L\}$...
- But still need $v_0 \in L$ witnessing $|v_0 - u| = D$.
- Intuitively true (lattice is discrete); annoying to formalize
- Instead, we obtain v_ϵ where $|v_\epsilon - u| \leq (1 + \epsilon)D$.
  - For arbitrary $\epsilon > 0$.
- $(1 + \epsilon)$ trickles down to final bound.

Formalizing Babai's nearest plane algorithm

- Informally: $D := \min\{|v - u| : v \in L\}$ is clearly w.d.
- Formally: Isabelle/HOL `min` only for finite sets.
- We use `inf` instead: $D := \inf\{|v - u| : v \in L\}$...
- But still need $v_0 \in L$ witnessing $|v_0 - u| = D$.
- Intuitively true (lattice is discrete); annoying to formalize
- Instead, we obtain v_ϵ where $|v_\epsilon - u| \leq (1 + \epsilon)D$.
  - For arbitrary $\epsilon > 0$.
- $(1 + \epsilon)$ trickles down to final bound.
  - Can be removed since $\epsilon > 0$ is arbitrary.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.
  - Can scale $L \subseteq \mathbb{Q}^n$ to $L' \in \mathbb{Z}^n$.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.
  - Can scale $L \subseteq \mathbb{Q}^n$ to $L' \in \mathbb{Z}^n$.
  - Target vector u can be in $\mathbb{Q}^n$.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.
  - Can scale $L \subseteq \mathbb{Q}^n$ to $L' \in \mathbb{Z}^n$.
  - Target vector u can be in $\mathbb{Q}^n$.
- In literature, can have $d \leq n$.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.
  - Can scale $L \subseteq \mathbb{Q}^n$ to $L' \in \mathbb{Z}^n$.
  - Target vector u can be in $\mathbb{Q}^n$.
- In literature, can have $d \leq n$.
  - Our proof requires an invertible (thus square) change-of-basis matrix.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.
  - Can scale $L \subseteq \mathbb{Q}^n$ to $L' \in \mathbb{Z}^n$.
  - Target vector u can be in $\mathbb{Q}^n$.
- In literature, can have $d \leq n$.
  - Our proof requires an invertible (thus square) change-of-basis matrix.
  - Thus, we restrict to $d = n$.

Formalizing Babai's nearest plane algorithm

- In literature, Babai's algorithm applies to $L \subseteq \mathbb{R}^n$
  - LLL only formalized for $L \subseteq \mathbb{Z}^n$.
  - Our formal Babai's algorithm is also restricted to $\mathbb{Z}^n$.
  - Can scale $L \subseteq \mathbb{Q}^n$ to $L' \in \mathbb{Z}^n$.
  - Target vector u can be in $\mathbb{Q}^n$.
- In literature, can have $d \leq n$.
  - Our proof requires an invertible (thus square) change-of-basis matrix.
  - Thus, we restrict to $d = n$.
- Like LLL, our formal Babai's algorithm is executable.

Formalizing the HNP

Formalizing the HNP

Proof. Let β, γ be two integers. Define the *modular distance* between β and γ as

$$\mathrm{dist}_p(\beta,\gamma) = \min_{b\in\mathbb{Z}} |\beta - \gamma - bp|$$

For example, $\mathrm{dist}_p(1,p) = 1$. Suppose $\beta \not\equiv \gamma \pmod p$ and they are both integers in the range $[1, p-1]$. Define

$$A = \Pr_t\left[\mathrm{dist}_p(\beta t, \gamma t) > 2p/2^\mu\right]$$

where t is an integer chosen uniformly at random in $[1, p-1]$. Then

$$A = \Pr_t\left[\frac{2p}{2^\mu} < (\beta-\gamma)t \bmod p < p - \frac{2p}{2^\mu}\right] = \frac{\lfloor p - \frac{2p}{2^\mu}\rfloor - \lceil \frac{2p}{2^\mu}\rceil}{p-1} \geq 1 - \frac{5}{2^\mu}$$

This follows since for every $x \in [\frac{2p}{2^\mu}, p - \frac{2p}{2^\mu}]$ there exists a t such that $(\beta-\gamma)t = x \pmod p$. In general, a lattice point v has the form

$$v = (\beta t_1 - b_1 p,\ \beta t_2 - b_2 p, \ldots, \beta t_d - b_d p,\ \beta/p)$$

for some integers $\beta, b_1, \ldots, b_d$. Suppose $\| v - u \| < p/2^\mu$. We show that with probability at least $\frac{1}{2}$ the vector v satisfies $\beta \equiv \alpha \pmod p$ and $\beta t_i - b_i p \in [0, p]$ for all i. Observe that if $\beta = \alpha \pmod p$, then $\beta t_i - b_i p \in [0,p]$ for all i. Otherwise at least one of the components of $v - u$ is bigger in absolute value than $p/2^\mu$.

Now, suppose $\beta \not\equiv \alpha \pmod p$. Then

$$\Pr[\| v - u \| > p/2^\mu] \geq \Pr[\exists i \ :\ \mathrm{dist}_p(t_i\beta, a_i) > p/2^\mu] \geq$$
$$\Pr[\exists i \ :\ \mathrm{dist}_p(t_i\beta, t_i\alpha) > 2p/2^\mu] = 1 - (1-A)^d \geq 1 - \left(\frac{5}{2^\mu}\right)^d$$

Since $\beta \not\equiv \alpha \pmod p$ there are exactly $p-1$ values of $\beta \bmod p$ to consider. Hence, the probability there exists a lattice point contradicting the statement of the theorem is at most

$$(p-1)\cdot\left(\frac{5}{2^\mu}\right)^d < \frac{1}{2}$$

The last inequality follows from the fact that $d(\mu - \log_2 5) > \log p + 1$. This completes the proof of the theorem. ∎

Formalizing the HNP

Informal proof: one-page

Formalizing the HNP

Informal proof: one-page

Formal proof: 4000 LoC

Formalizing the HNP

<u>Informal</u> proof: one-page

<u>Formal</u> proof: 4000 LoC

We clarify:

Formalizing the HNP

Informal proof: one-page

Formal proof: 4000 LoC

We clarify:

- proof steps,

Formalizing the HNP

Informal proof: one-page

Formal proof: 4000 LoC

We clarify:

- proof steps,
- definitions,

Formalizing the HNP

Informal proof: one-page

Formal proof: 4000 LoC

We clarify:

- proof steps,
- definitions,
- bounds and arithmetic.

Formalizing the HNP

Counting is hard

Formalizing the HNP

Counting is hard

- Original paper:

Formalizing the HNP

Counting is hard

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

Formalizing the HNP

Counting is hard

- Original paper:
  - $\left|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \leq \frac{2p}{2^\mu}\}\right|$
    $= \left\lfloor p - \frac{2p}{2^\mu} \right\rfloor - \left\lceil \frac{2p}{2^\mu} \right\rceil$
    $\geq (p-1)(1 - \frac{5}{2^\mu})$.
  - Edge cases and ceil/floor arithmetic difficult to formalize.

Formalizing the HNP

Counting is hard

- Original paper:
  - $\left|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \le \frac{2p}{2^\mu}\}\right|$
    $= \left\lfloor p - \frac{2p}{2^\mu} \right\rfloor - \left\lceil \frac{2p}{2^\mu} \right\rceil$
    $\ge (p-1)(1 - \frac{5}{2^\mu})$.
  - Edge cases and ceil/floor arithmetic difficult to formalize.
- Our approach:

Formalizing the HNP

Counting is hard

- Original paper:
  - $\left|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \leq \frac{2p}{2^\mu}\}\right|$
    $= \left\lfloor p - \frac{2p}{2^\mu} \right\rfloor - \left\lceil \frac{2p}{2^\mu} \right\rceil$
    $\geq (p-1)(1 - \frac{5}{2^\mu})$.
  - Edge cases and ceil/floor arithmetic difficult to formalize.
- Our approach:
  - $\left|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \leq B\}\right| \leq 2B$.

Formalizing the HNP

Counting is hard

- Original paper:
  - $$\left|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \leq \tfrac{2p}{2^\mu}\}\right|$$
    $$= \left\lfloor p - \frac{2p}{2^\mu} \right\rfloor - \left\lceil \frac{2p}{2^\mu} \right\rceil$$
    $$\geq (p-1)(1 - \tfrac{5}{2^\mu}).$$
  - Edge cases and ceil/floor arithmetic difficult to formalize.
- Our approach:
  - $\left|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \leq B\}\right| \leq 2B.$
  - Weaker, but sufficient.

Formalizing the HNP

Counting is hard

- Original paper:
  - $|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \le \frac{2p}{2^\mu}\}|$
    $= \left\lfloor p - \frac{2p}{2^\mu} \right\rfloor - \left\lceil \frac{2p}{2^\mu} \right\rceil$
    $\ge (p-1)(1 - \frac{5}{2^\mu})$.
  - Edge cases and ceil/floor arithmetic difficult to formalize.
- Our approach:
  - $|\{t \in (\mathbb{Z}/p\mathbb{Z})^\times : \mathrm{dist}_p(\beta t, \alpha t) \le B\}| \le 2B$.
  - Weaker, but sufficient.
  - Simple argument, simple to formalize

Formalizing the HNP

Clarifying MSB_k definition

Clarifying MSB_k definition

- Original paper:

Formalizing the HNP

Clarifying MSB_k definition

- Original paper:
  - First, $\mathrm{MSB}_k(x)$ is the unique $t \in \mathbb{Z}$ such that $(t-1) \cdot \frac{p}{2^k} \leq x < t \cdot \frac{p}{2^k}$.

Formalizing the HNP

Clarifying MSB_k definition

- Original paper:
  - First, $\mathrm{MSB}_k(x)$ is the unique $t \in \mathbb{Z}$ such that $(t-1) \cdot \frac{p}{2^k} \leq x < t \cdot \frac{p}{2^k}$.
    - Like like a right-shift (by $n - k$ bits).

Formalizing the HNP

Clarifying MSB_k definition

- Original paper:
  - First, $\mathrm{MSB}_k(x)$ is the unique $t \in \mathbb{Z}$ such that $(t-1) \cdot \frac{p}{2^k} \le x < t \cdot \frac{p}{2^k}$.
    - Like like a right-shift (by $n - k$ bits).
  - Actually, assume $\mathrm{MSB}_k(x)$ satisfies $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^{k+1}}$.

Formalizing the HNP

Clarifying MSB_k definition

- Original paper:
  - First, $\mathrm{MSB}_k(x)$ is the unique $t \in \mathbb{Z}$ such that $(t-1) \cdot \frac{p}{2^k} \leq x < t \cdot \frac{p}{2^k}$.
    - Like like a right-shift (by $n - k$ bits).
  - Actually, assume $\mathrm{MSB}_k(x)$ satisfies $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^{k+1}}$.
    - Like like a right-then-left-shift.

Formalizing the HNP

Clarifying MSB_k definition

- Original paper:
  - First, $\mathrm{MSB}_k(x)$ is the unique $t \in \mathbb{Z}$ such that $(t-1) \cdot \frac{p}{2^k} \leq x < t \cdot \frac{p}{2^k}$.
    - Like like a right-shift (by $n - k$ bits).
  - Actually, assume $\mathrm{MSB}_k(x)$ satisfies $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^{k+1}}$.
    - Like like a right-then-left-shift.
  - In fact, only need $\frac{p}{2^k}$.

Formalizing the HNP

Clarifying MSB_k definition

- Original paper:
  - First, $\mathrm{MSB}_k(x)$ is the unique $t \in \mathbb{Z}$ such that $(t-1) \cdot \frac{p}{2^k} \leq x < t \cdot \frac{p}{2^k}$.
  - Actually, assume $\mathrm{MSB}_k(x)$ satisfies $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^{k+1}}$.
  - In fact, only need $\frac{p}{2^k}$.
- Our approach:
  - Work in Isabelle locale fixing MSB_k operator and assuming $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^k}$.

Formalizing the HNP

Clarifying MSB_k definition

- Our approach:
  - Work in Isabelle locale fixing MSB_k operator and assuming $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^k}$.

Formalizing the HNP

Clarifying MSB_k definition

- Our approach:
  - Work in Isabelle locale fixing MSB_k operator and assuming $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^k}$.
  - Instantiate locale with original MSB_k definition,

Formalizing the HNP

Clarifying MSB_k definition

- Our approach:
  - Work in Isabelle locale fixing MSB_k operator and assuming $|x - \mathrm{MSB}_k(x)| < \frac{p}{2^k}$.
  - Instantiate locale with original MSB_k definition,
  - as well as simple "right-then-left-shift" definition.

Formalizing the HNP

Probability helper lemmas

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "`do`" notation.

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "`do`" notation.
- We formulate cryptographic "game" as probabilistic algorithm.

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "`do`" notation.
- We formulate cryptographic "game" as probabilistic algorithm.

```
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))
  }"
```

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "**do**" notation.
- We often use pattern:

```
do {x ← p; return_pmf (P x)}
```

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "`do`" notation.
- We often use pattern:

  `do {x ← p; return_pmf (P x)}`
- We prove lemmas to reason about these expressions.

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "**`do`**" notation.
- We often use pattern:
  `do {x ← p; return_pmf (P x)}`
- We prove lemmas to reason about these expressions.
  - We lift existing measure-theoretic lemmas to this level of abstraction.

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "**`do`**" notation.
- We often use pattern:

  `do {x ← p; return_pmf (P x)}`
- We prove lemmas to reason about these expressions.
  - We lift existing measure-theoretic lemmas to this level of abstraction.
- Helper lemmas aid expressivity

Formalizing the HNP

Probability helper lemmas

- In Isabelle/HOL lib: PMF monad with "**`do`**" notation.
- We often use pattern:

  `do {x ← p; return_pmf (P x)}`
- We prove lemmas to reason about these expressions.
  - We lift existing measure-theoretic lemmas to this level of abstraction.
- Helper lemmas aid expressivity and improve Sledgehammer performance.

Formalizing the HNP

Hiding α in locale

Formalizing the HNP

Hiding α in locale

- α fixed in locale.

Formalizing the HNP

Hiding α in locale

- α fixed in locale.
- Adversary defined as function in locale.

Formalizing the HNP

Hiding α in locale

- α fixed in locale.
- Adversary defined as function in locale.
- How to ensure α is "hidden" from adversary?

Formalizing the HNP

Hiding α in locale

- α fixed in locale.
- Adversary defined as function in locale.
- How to ensure α is "hidden" from adversary?
- Can manually inspect that adversary does not use α.

Formalizing the HNP

Hiding α in locale

- α fixed in locale.
- Adversary defined as function in locale.
- How to ensure α is "hidden" from adversary?
- Can manually inspect that adversary does not use α.
  - Not satisfying; not in spirit of formal verification.

Formalizing the HNP

Hiding α in locale

- α fixed in locale.
- Adversary defined as function in locale.
- How to ensure α is "hidden" from adversary?
- Can manually inspect that adversary does not use α.
  - Not satisfying; not in spirit of formal verification.
- Instead, we use locale hierarchy; define adversary before α is fixed.

Formalizing the HNP

Hiding α in locale

- α fixed in locale.
- Adversary defined as function in locale.
- How to ensure α is "hidden" from adversary?
- Can manually inspect that adversary does not use α.
  - Not satisfying; not in spirit of formal verification.
- Instead, we use locale hierarchy; define adversary before α is fixed.
  - Simple, but streamlines manual verification.

Future work

Future work

- Explore further library and automation support for game-based and probabilistic reasoning.

Future work

- Explore further library and automation support for game-based and probabilistic reasoning.
- Other hidden number problems:

Future work

- Explore further library and automation support for game-based and probabilistic reasoning.
- Other hidden number problems:
  - Elliptic curve HNP
  - Extended HNP
  - Modular inverse HNP
  - ... many more!

Future work

- Explore further library and automation support for game-based and probabilistic reasoning.
- Other hidden number problems:
  - Elliptic curve HNP
  - Extended HNP
  - Modular inverse HNP
  - ... many more!
- Formalize time complexity of adversary

Future work

- Explore further library and automation support for game-based and probabilistic reasoning.
- Other hidden number problems:
  - Elliptic curve HNP
  - Extended HNP
  - Modular inverse HNP
  - ... many more!
- Formalize time complexity of adversary
  - Need to formalize complexity of Babai.

Future work

- Explore further library and automation support for game-based and probabilistic reasoning.
- Other hidden number problems:
  - Elliptic curve HNP
  - Extended HNP
  - Modular inverse HNP
  - ... many more!
- Formalize time complexity of adversary
  - Need to formalize complexity of Babai.
  - Luckily, LLL complexity is formalized.