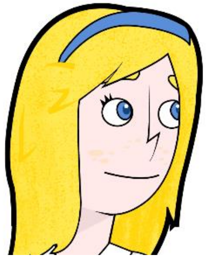
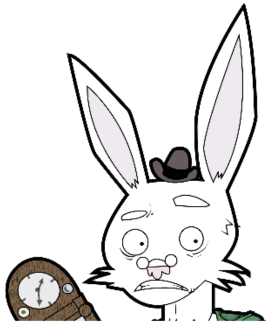
Lattice-based Cryptography



Cryptography Course

Prof. Daniele Venturi
Dipartimento di Informatica



Sapienza Università di Roma

Academic Year 2024/2025

The Quantum Threat

- An algorithm by Shor [Sho94] solves the factoring and discrete logarithm problems in **polynomial-time** on a **quantum** machine
  - The algorithm requires an **ideal** quantum Turing machine
  - Factoring a 1024-bit integer requires **2050** logical **qubits** and a quantum circuit with **billions** of quantum gates
  - Despite recent progress on quantum computation, current implementations can only factor **tiny numbers** (e.g., 15 and 21)
- Nevertheless, the NIST started in 2017 a process to solicit, evaluate, and standardize **quantum-resistant** cryptography
  - The selected algorithms were announced in 2022
  - Most of these algorithms are based on **lattices**

What's the Rush?

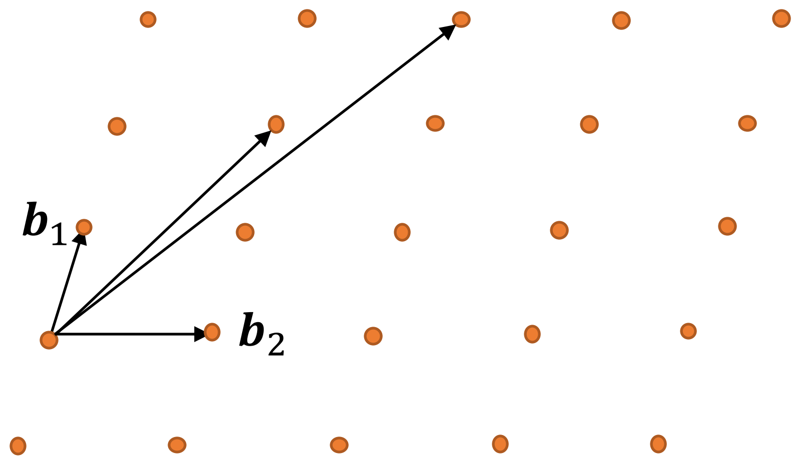
- Big quantum computers won't be available for **many years**
  - If **ever**…
  - Can't we just wait?
- Better safe than sorry
  - **Harvesting attacks:** Store today's keys/ciphertexts to break later
  - **Rewrite history:** Forge signatures for old keys
  - Deploying new cryptography **at scale** requires 10+ years

Lattices

What is a Lattice?

- Simply, a set of points in a **high-dimensional** space
  - Arranged **periodically**
- Formally, take n **linearly independent** vectors $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ in $\mathbb{R}^n$ and consider all **integer** combinations

$$\mathcal{L} = \{a_1\boldsymbol{b}_1 + \cdots + a_n\boldsymbol{b}_n : a_1, \dots, a_n \in \mathbb{Z}\}$$



- We call $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ a **basis**
- The **same lattice** may have **different** equivalent **basis**
  - Even if base vectors are **long**, there are **short vectors** in the lattice

History

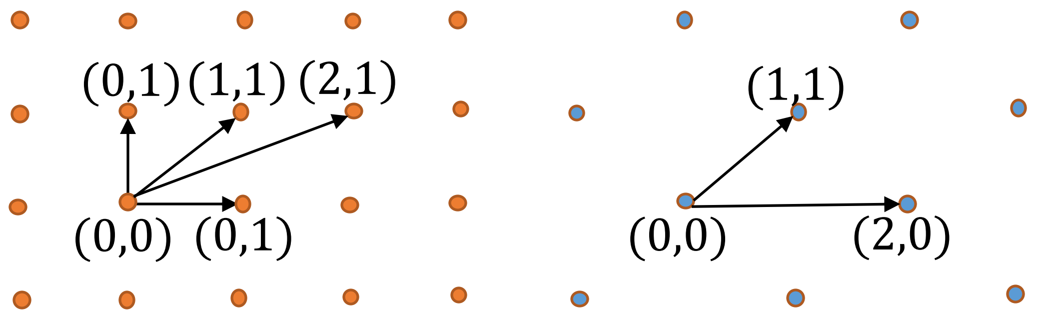
- **Geometric** objects with rich mathematical structure
- Considerable **mathematical interest** starting from Gauss (1801), Hermite (1850), and Minkowski (1896)



- Recently, many **interesting applications** (cryptanalysis, factoring rational polynomials, finding integer relations, …)

Equivalent Bases

- Sometimes, we write $\mathcal{L}(\boldsymbol{B})$ where $\boldsymbol{B}$ is the matrix whose columns are $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$
  - One can also define a lattice as a **discrete additive subgroup** of $\mathbb{R}^n$



- **Equivalent** bases:
  - Permute vectors (i.e., $\boldsymbol{b}_i \leftrightarrow \boldsymbol{b}_j$)
  - Negate vectors (i.e., $\boldsymbol{b}_i \leftarrow (-\boldsymbol{b}_i)$)
  - Add integer multiple of another vector (i.e., $\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i + k \cdot \boldsymbol{b}_j, k \in \mathbb{Z}$)

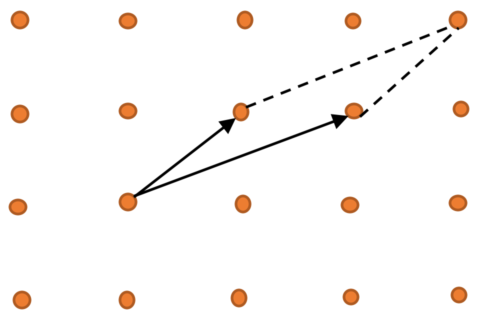
- **Theorem:** Two bases $\boldsymbol{B}_1, \boldsymbol{B}_2$ are **equivalent** iff $\boldsymbol{B}_1 = \boldsymbol{B}_2 \cdot \boldsymbol{U}$
  - $\boldsymbol{U}$ **unimodular** (i.e., integer matrix with $\det(\boldsymbol{U}) = \pm 1$)

Equivalent Bases

- Let $\boldsymbol{B}_1 = \boldsymbol{B}_2 \cdot \boldsymbol{U}$
  - If $\boldsymbol{U}$ is **unimodular**, so is $\boldsymbol{U}^{-1}$ and $\boldsymbol{B}_2 = \boldsymbol{B}_1 \cdot \boldsymbol{U}^{-1}$
  - Hence, $\mathcal{L}(\boldsymbol{B}_1) \subseteq \mathcal{L}(\boldsymbol{B}_2)$ and $\mathcal{L}(\boldsymbol{B}_2) \subseteq \mathcal{L}(\boldsymbol{B}_1)$ or $\mathcal{L}(\boldsymbol{B}_1) = \mathcal{L}(\boldsymbol{B}_2)$
- Let $\boldsymbol{B}_1 = \boldsymbol{B}_2 \cdot \boldsymbol{W}$ and $\boldsymbol{B}_2 = \boldsymbol{B}_1 \cdot \boldsymbol{V}$ for **integer matrices** $\boldsymbol{V}, \boldsymbol{W}$
  - Hence, $\boldsymbol{B}_1 = \boldsymbol{B}_1 \cdot \boldsymbol{V} \cdot \boldsymbol{W}$ or $\boldsymbol{B}_1 \cdot (\boldsymbol{I} - \boldsymbol{V} \cdot \boldsymbol{W}) = \mathbf{0}$
  - Since the vectors in $\boldsymbol{B}_1$ are **linearly independent**, $\boldsymbol{I} - \boldsymbol{V} \cdot \boldsymbol{W} = \mathbf{0}$
  - Thus, $\boldsymbol{V} \cdot \boldsymbol{W} = \boldsymbol{I}$ and $\det(\boldsymbol{V}) \cdot \det(\boldsymbol{W}) = \det(\boldsymbol{V} \cdot \boldsymbol{W}) = 1$
  - Since $\boldsymbol{V}, \boldsymbol{W}$ are **integer matrices** $\det(\boldsymbol{V}), \det(\boldsymbol{W}) \in \mathbb{Z}$ and $\det(\boldsymbol{V}) = \det(\boldsymbol{W}) = \pm 1$

The Fundamental Region

- The **fundamental region** of a lattice corresponds to a **periodic tiling** of $\mathbb{R}^n$ by copies of some body
  - For instance, $[0,1)$ is a fundamental region of the **integer lattice** $\mathbb{Z}$, as every $x \in \mathbb{R}$ is in the **unique translate** $\lfloor x \rfloor + [0,1)$



- A lattice base yields a fundamental region called the **fundamental parallelepiped**

$$\mathcal{P}(\boldsymbol{B}) = \boldsymbol{B} \cdot [0,1)^n = \left\{ \sum_{i=1}^{n} c_i \cdot \boldsymbol{b}_i : c_i \in [0,1) \right\}$$

- Useful for measuring **arbitrary** points **relative to a lattice**
  - $\mathcal{P}(\boldsymbol{B})$ is **half-open** and $\boldsymbol{v} + \mathcal{P}(\boldsymbol{B})$ for $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ forms a **tiling** of $\mathbb{R}^n$
  - For **every** $\boldsymbol{x} \in \mathbb{R}^n$, there is a **unique** $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ s.t. $\boldsymbol{x} \in (\boldsymbol{v} + \mathcal{P}(\boldsymbol{B}))$

Determinant

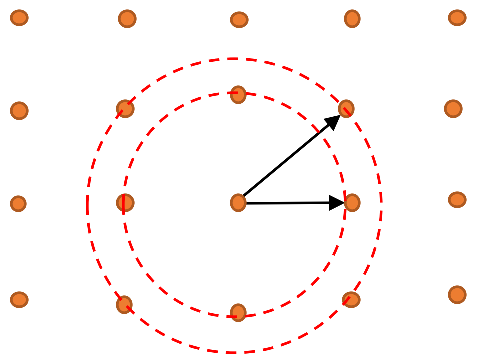
- The **determinant** of a lattice $\mathcal{L}(\boldsymbol{B})$ is $\det(\mathcal{L}) = |\det(\boldsymbol{B})|$
- Note that this is well defined, as for every **unilateral** $\boldsymbol{U}$

$$|\det(\boldsymbol{B} \cdot \boldsymbol{U})| = |\det(\boldsymbol{B}) \cdot \det(\boldsymbol{U})| = |\det(\boldsymbol{B})|$$

- The determinant corresponds to the **volume** of the **fundamental parallelepiped**
  - The determinant is the **reciprocal** of the **density** (i.e., **big** determinant means **sparse** lattice)
  - Moreover, the volume is the **same** for **every** fundamental region

Successive Minima

- Let $\lambda_1(\mathcal{L})$ be the length of the **shortest non-zero** vector in a lattice $\mathcal{L}$
  - Usually, in terms of the **Euclidean** norm
  - The shortest vector is **never unique**, as for every $\boldsymbol{v} \in \mathcal{L}$ also $-\boldsymbol{v} \in \mathcal{L}$
- More generally, $\lambda_k(\mathcal{L})$ denotes the **radius** of the **ball** containing k **linearly independent** vectors
  - For $k = n$ the ball contains a basis of the entire space

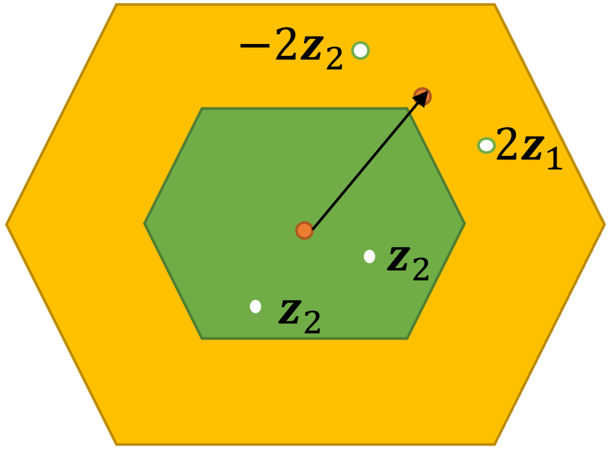


Minkowski's Theorem

- **Lemma (Blichfeld):** For any lattice $\mathcal{L}$ and set $\mathcal{S}$ with $\mathrm{vol}(\mathcal{S}) > \det(\mathcal{L})$, $\exists$ **distinct** $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{S}$ s.t. $\boldsymbol{z}_1 - \boldsymbol{z}_2 \in \mathcal{L}$
- Consider $\mathcal{S}_{\boldsymbol{x}} = \mathcal{S} \cap (\boldsymbol{x} + \mathcal{P}(\boldsymbol{B}))$ with $\boldsymbol{x} \in \mathcal{L}(\boldsymbol{B})$
  - So, $\mathcal{S} = \bigcup_{\boldsymbol{x}\in\mathcal{L}(\boldsymbol{B})} \mathcal{S}_{\boldsymbol{x}}$ and $\mathrm{vol}(\mathcal{S}) = \sum_{\boldsymbol{x}\in\mathcal{L}(\boldsymbol{B})} \mathrm{vol}(\mathcal{S}_{\boldsymbol{x}})$
  - For **each** $\boldsymbol{x} \in \mathcal{L}(B)$, $\mathcal{S}_{\boldsymbol{x}} - \boldsymbol{x} = (\mathcal{S} - \boldsymbol{x}) \cap \mathcal{P}(\boldsymbol{B}) \subseteq \mathcal{P}(\boldsymbol{B})$
  - Then, $\mathrm{vol}\big(\mathcal{P}(\boldsymbol{B})\big) < \mathrm{vol}(\mathcal{S}) = \sum_{\boldsymbol{x}\in\mathcal{L}(\boldsymbol{B})} \mathrm{vol}(\mathcal{S}_{\boldsymbol{x}}) = \sum_{\boldsymbol{x}\in\mathcal{L}(\boldsymbol{B})} \mathrm{vol}(\mathcal{S}_{\boldsymbol{x}} - \boldsymbol{x})$
- There are **distinct** $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{L}(\boldsymbol{B})$ s.t. $(\mathcal{S}_{\boldsymbol{x}} - \boldsymbol{x}) \cap (\mathcal{S}_{\boldsymbol{y}} - \boldsymbol{y}) \neq \emptyset$
  - Take $\boldsymbol{z} \in (\mathcal{S}_{\boldsymbol{x}} - \boldsymbol{x}) \cap (\mathcal{S}_{\boldsymbol{y}} - \boldsymbol{y})$, so that $\boldsymbol{z}_1 = \boldsymbol{z} + \boldsymbol{x} \in \mathcal{S}_{\boldsymbol{x}} \subseteq \mathcal{S}$ and $\boldsymbol{z}_2 = \boldsymbol{z} + \boldsymbol{y} \in \mathcal{S}_{\boldsymbol{y}} \subseteq \mathcal{S}$
  - Hence, $\boldsymbol{z}_1 - \boldsymbol{z}_2 = \boldsymbol{x} - \boldsymbol{y} \in \mathcal{L}(\boldsymbol{B})$

Minkowski's Theorem

- **Theorem (Minkowski):** For any lattice $\mathcal{L}$ and **convex**, **zero-symmetric**, set $\mathcal{S}$ with $\mathrm{vol}(\mathcal{S}) > 2^n \det(\mathcal{L})$, there exists a **non-zero** lattice point in $\mathcal{S}$



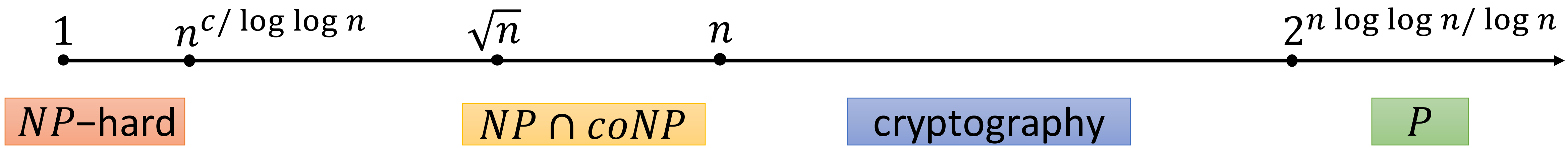
  - Let $\mathcal{S}/2 = \{\boldsymbol{x}: 2\boldsymbol{x} \in \mathcal{S}\}$ with $\mathrm{vol}(\mathcal{S}/2) = 2^{-n} \cdot \mathrm{vol}(\mathcal{S}) > \det(\mathcal{L})$
  - Take $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{S}/2$; by **Blichfeld** $\boldsymbol{z}_1 - \boldsymbol{z}_2 \in \mathcal{L}$
  - Now, $2\boldsymbol{z}_1, -2\boldsymbol{z}_2 \in \mathcal{S}$ and $\boldsymbol{z}_1 - \boldsymbol{z}_2 = \frac{2\boldsymbol{z}_1 - 2\boldsymbol{z}_2}{2} \in \mathcal{S}$

- **Corollary:** For every $\mathcal{L}$, we have that $\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$
  - Let $\ell = \min_{\boldsymbol{x} \in \mathcal{L} \setminus \boldsymbol{0}} \|\boldsymbol{x}\|_\infty$ and assume $\ell > \det(\mathcal{L})^{1/n}$
  - The hypercube $\mathcal{C} = \{\boldsymbol{x}: \|\boldsymbol{x}\|_\infty < \ell\}$ is **convex**, **symmetric** and has volume $\mathrm{vol}(\mathcal{C}) = (2\ell)^n > 2^n \det(\mathcal{L})$

Hard Problems

- $\mathbf{SVP}_\gamma$: Given $\boldsymbol{B}$, find vector in $\mathcal{L}(\boldsymbol{B})$ with length $\leq \gamma \cdot \lambda_1(\mathcal{L}(\boldsymbol{B}))$
- $\mathbf{GapSVP}_\gamma$: Given $\boldsymbol{B}$, **decide** if $\lambda_1(\mathcal{L}(\boldsymbol{B}))$ is ≤ 1 or $\geq \gamma$
- $\mathbf{SIVP}_\gamma$: Given $\boldsymbol{B}$, find n **linearly independent** vectors in $\mathcal{L}(\boldsymbol{B})$ with length $\leq \gamma \cdot \lambda_n(\mathcal{L}(\boldsymbol{B}))$
- $\mathbf{CVP}_\gamma$: Given $\boldsymbol{B}$ and $\boldsymbol{v}$, find a lattice point that is at most γ times **farther** than the **closest** lattice point
  - It is known that $\mathbf{SVP}_\gamma \leq \mathbf{CVP}_\gamma$
- $\mathbf{BDD}$: Find **closest** lattice point, given that $\boldsymbol{v}$ is **already close**

General Hardness Results



- Exact algorithms take time 2^n
- **Polynomial-time** algorithm for gap $\gamma = 2^{n\log\log n/\log n}$
- No better **quantum** algorithm known
- NP **hardness** for gap $\gamma = n^{c/\log\log n}$
  - For cryptographic applications, we need $\gamma = \Omega(n)$
  - Not believed to be NP-hard for $\gamma = \sqrt{n}$

SAPIENZA UNIVERSITÀ DI ROMA

Small Integer Solution Problem

- Fix **dimension** n, and **modulus** q (e.g., $q \approx n^2$)
- Given **random** vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_m \in \mathbb{Z}_q^n$, find **non-zero small** $z_1, \dots, z_m \in \mathbb{Z}$ such that

$$z_1 \cdot \boldsymbol{a}_1 + z_2 \cdot \boldsymbol{a}_2 + \cdots + z_m \cdot \boldsymbol{a}_m = \mathbf{0} \quad \text{in } \mathbb{Z}_q^n$$

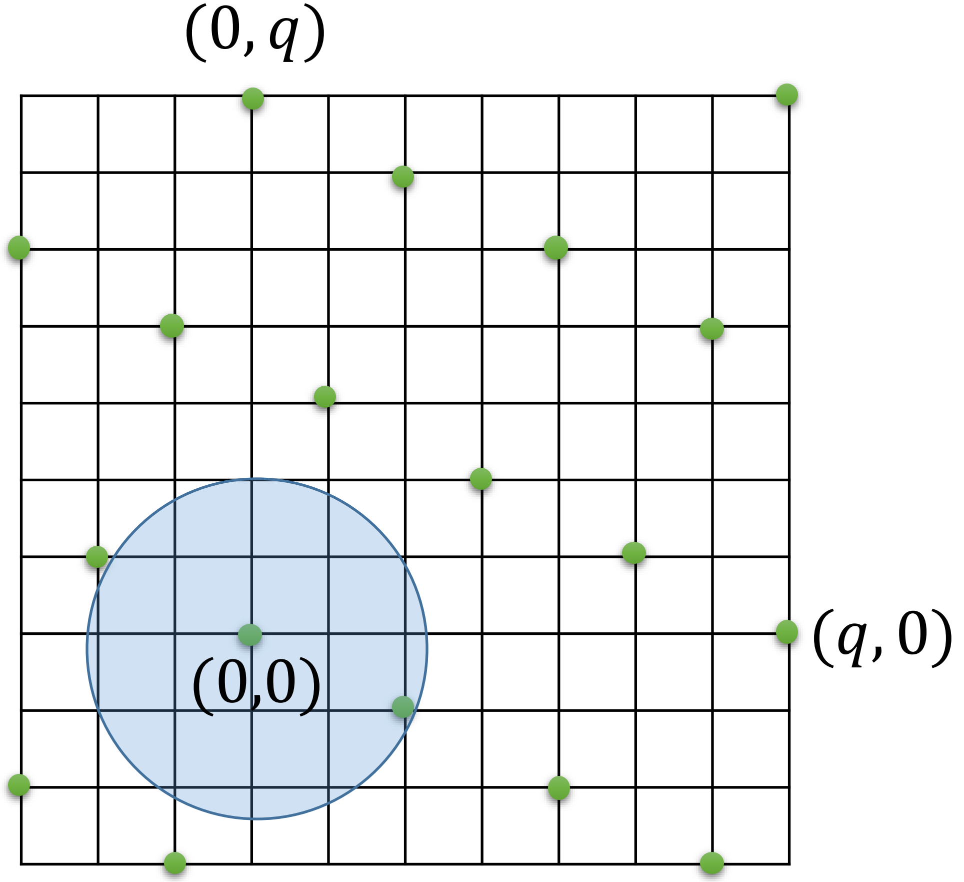
- Observations:
  - Trivial if the size of the z_i's is **not restricted** (Gaussian elimination)
  - Equivalently, find **non-zero short** $\boldsymbol{z} \in \mathbb{Z}^m$ s.t. $\boldsymbol{A} \cdot \boldsymbol{z} = \mathbf{0} \in \mathbb{Z}_q^n$

SIS as a Lattice Problem

- Matrix $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m) \in \mathbb{Z}_q^{n \times m}$

$$\mathcal{L}^{\perp}(\boldsymbol{A}) = \{\boldsymbol{z} \in \mathbb{Z}^m : \boldsymbol{A} \cdot \boldsymbol{z} = \mathbf{0}\}$$

Find **short** ($\|\boldsymbol{z}\| \leq \beta \ll q$) solutions for **random** $\boldsymbol{A}$



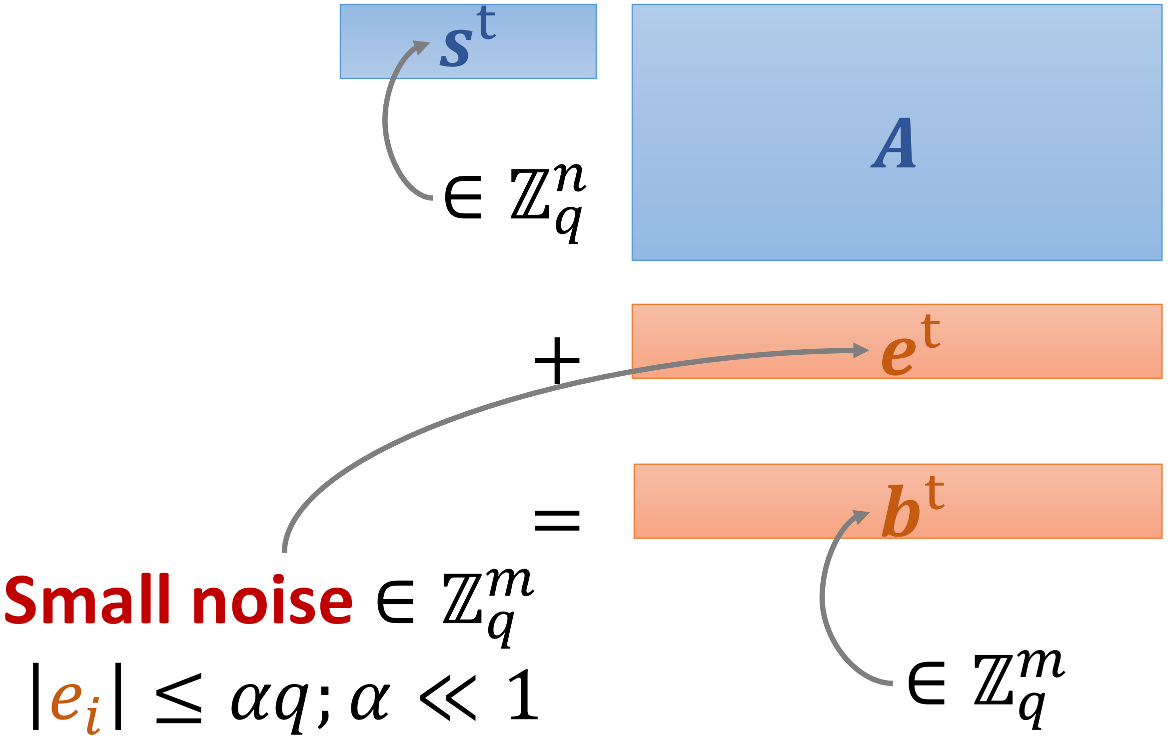
- **Theorem (Ajt96).** For **any** n-dimensional lattice, it holds that:

$$\mathbf{GapSVP}_{\beta\sqrt{n}}, \mathbf{SIVP}_{\beta\sqrt{n}} \leq \mathbf{SIS}_{\beta}$$

  - Also true for any lattice **coset** $\mathcal{L}_{\boldsymbol{u}}^{\perp}(\boldsymbol{A}) = \{\boldsymbol{z} \in \mathbb{Z}^m : \boldsymbol{A} \cdot \boldsymbol{z} = \boldsymbol{u}\} = \boldsymbol{u} + \mathcal{L}^{\perp}(\boldsymbol{A})$ (i.e., **inhomogenuous** SIS)

Learning with Errors [Reg05]

- Dimension n, modulus $q > 2$, **noise** distribution χ
- **Find $\boldsymbol{s} \in \mathbb{Z}_q^n$ given m noisy random** inner product equations



- Trivial **without** noise
- **Gaussian** distribution over $\mathbb{Z}$, with std deviation $\geq \sqrt{n}$ and $\ll q$
  - Rate parameter $\alpha \ll 1$
- Need $\alpha q > \sqrt{n}$ for **worst-case hardness** and because there is an $\exp((\alpha q)^2)$-time attack

Decisional LWE

- **Distinguish** the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ from random $(\boldsymbol{A}, \boldsymbol{b})$
  - Decisional LWE is **equivalent** to Search LWE

$$\boldsymbol{s}^{\mathrm{t}}\boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}} = \boldsymbol{b}^{\mathrm{t}} \qquad \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{b}^{\mathrm{t}} \end{pmatrix} \approx \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{b}^{\mathrm{t}} \end{pmatrix} \equiv \mathbf{U}_q^{(n+1)\times m}$$

$\in \mathbb{Z}_q^n$

Small noise $\in \mathbb{Z}_q^m$

$|e_i| \leq \alpha q; \alpha \ll 1$

$\in \mathbb{Z}_q^m$

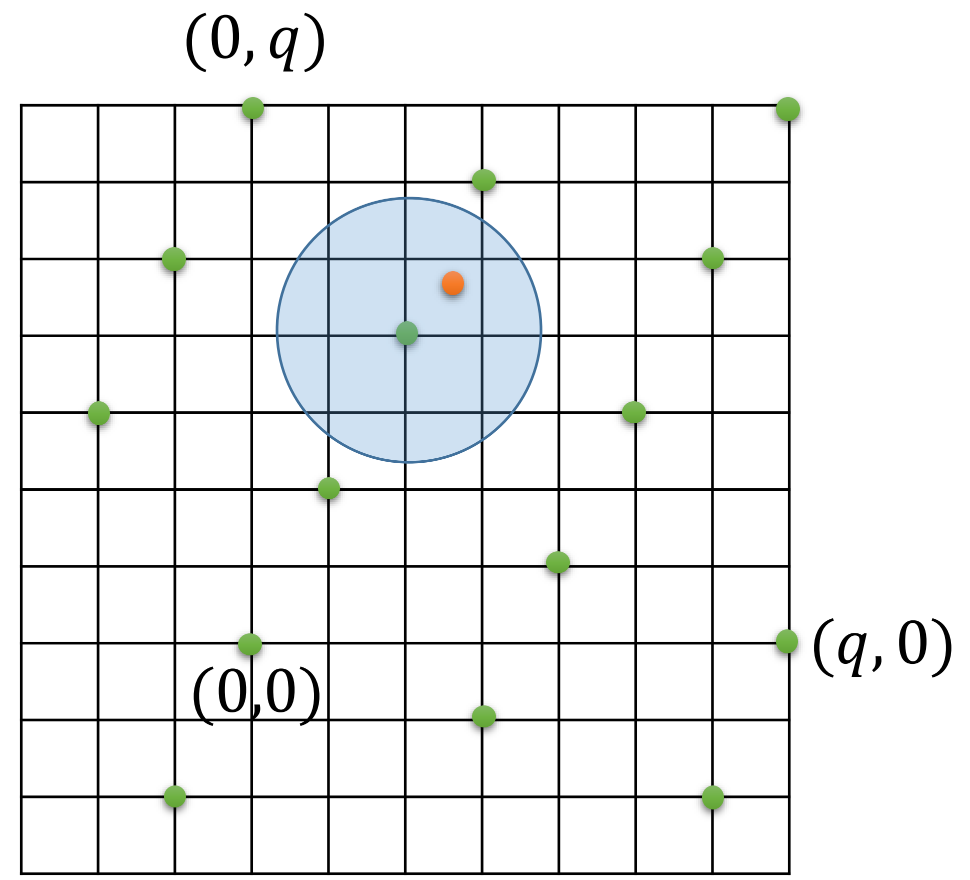
Uniform distribution over $\mathbb{Z}_q^{(n+1)\times m}$

LWE as a Lattice Problem

- Matrix $\boldsymbol{A} = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_m) \in \mathbb{Z}_q^{n \times m}$

$$\mathcal{L}(\boldsymbol{A}) = \{\boldsymbol{z} \in \mathbb{Z}^m : \boldsymbol{z}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A}\}$$

LWE is BDD on $\mathcal{L}(\boldsymbol{A})$: Given $\boldsymbol{b}^{\mathrm{t}} \approx \boldsymbol{z}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A}$ find $\boldsymbol{z}$



- **Theorem (Reg05,Pei10).** For **any** n-dimensional lattice, it holds that:

$$\mathbf{GapSVP}_{\alpha n}, \mathbf{SIVP}_{\alpha n} \leq \mathbf{LWE}$$

  - **Quantum** reduction for **broad** parameters [Reg05]
  - **Classical** reduction for **restricted** parameters (e.g., $q \approx 2^n$) [Pei10]

Hardness of LWE

- More formally define the **LWE distribution** as

$$\mathbf{LWE}[n, m, q, \chi] = \left\{ (\boldsymbol{A}, \boldsymbol{b}) : \begin{array}{c} \boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}; \boldsymbol{s} \leftarrow \mathbb{Z}_q^n; \\ \boldsymbol{e} \leftarrow \chi^m; \boldsymbol{b}^{\mathrm{t}} = [\boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}]_q \end{array} \right\}$$

- Parameters:
  - $\alpha = 1/\mathrm{poly}(n)$ or $\alpha = 2^{-n^{\epsilon}}$ (**stronger** assumption as α **decreases**)
  - $m = \Theta(n \log q)$ or $m = \mathrm{poly}(n)$ (**stronger** assumption as m **increases**)
  - $q = 2^{n^{\epsilon}}$ or $q = \mathrm{poly}(n)$ (**stronger** assumption as q **increases**)
  - Noise distribution χ such that $\mathbb{P}[|e| > \alpha q : e \leftarrow \chi] \leq \mathrm{negl}(n)$

Simple Properties

- Check a **candidate** solution $\boldsymbol{t} \in \mathbb{Z}_q^n$
  - Test if all the elements in $\boldsymbol{b} - \langle \boldsymbol{t}, \boldsymbol{a} \rangle$ are small
  - If $\boldsymbol{t} \neq \boldsymbol{s}$, then $\boldsymbol{b} - \langle \boldsymbol{t}, \boldsymbol{a} \rangle = \langle \boldsymbol{s} - \boldsymbol{t}, \boldsymbol{a} \rangle + e$ is **well-spread** in $\mathbb{Z}_q$
- **Shift** the secret by any $\boldsymbol{r} \in \mathbb{Z}_q^n$
  - Given $(\boldsymbol{a}, b = \langle \boldsymbol{s}, \boldsymbol{a} \rangle + e)$, output $(\boldsymbol{a}, b' = b + \langle \boldsymbol{r}, \boldsymbol{a} \rangle = \langle \boldsymbol{s} + \boldsymbol{r}, \boldsymbol{a} \rangle + e)$
  - Using **random** $\boldsymbol{r}$ yields a random **self-reduction**
  - **Amplification** of success probabilities (i.e., **non-negligible** success probability for **random** $\boldsymbol{s} \in \mathbb{Z}_q^n$ implies **overwhelming** success probability for **every** $\boldsymbol{s} \in \mathbb{Z}_q^n$)
- **Multiple** secrets: $(\boldsymbol{a}, b_1 = \langle \boldsymbol{s}_1, \boldsymbol{a} \rangle + e_1, \dots, \langle \boldsymbol{s}_t, \boldsymbol{a} \rangle + e_t)$ indistinguishable from **random** $(\boldsymbol{a}, b_1, \dots, b_t)$

Search/Decision Equivalence

- Suppose we are given an oracle that **perfectly distinguishes** pairs $(\boldsymbol{a}, b = \langle \boldsymbol{s}, \boldsymbol{a} \rangle + e)$ from random $(\boldsymbol{a}, b)$
- To find s_1, it suffices to **test** if $s_1 = 0$
  - Because we can **shift** s_1 by $0,1, \dots, q-1$ (assuming $q = \mathrm{poly}(n)$)
  - Then we can do the same for $s_2, \dots, s_n$
- The test: For each $(\boldsymbol{a}, b)$, choose **random** $r \in \mathbb{Z}_q$ and invoke the oracle on pairs $(\boldsymbol{a}' = \boldsymbol{a} - (r, 0, \dots, 0), b)$
- Note that $b = \langle \boldsymbol{s}, \boldsymbol{a}' \rangle + s_1 \cdot r + e$
  - If $s_1 = 0$, then $b = \langle \boldsymbol{s}, \boldsymbol{a}' \rangle + e$ and the oracle **accepts**
  - If $s_1 \neq 0$, then b is **uniform** (assuming q **prime**) and the oracle **rejects**

LWE with Short Secrets

- **Theorem [M01,ACPS09]:** LWE is **no easier** if the secret is drawn from the **error distribution** χ
  - Intuition: Finding $\boldsymbol{e}$ **equivalent** to finding $\boldsymbol{s}$ (i.e., $\boldsymbol{b}^{\mathrm{t}} - \boldsymbol{e}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A}$)
- **Transformation** from secret $\boldsymbol{s} \in \mathbb{Z}_q^n$ to secret $\bar{\boldsymbol{e}} \leftarrow \chi^n$
  - Draw samples to get $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{b}}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \bar{\boldsymbol{A}} + \bar{\boldsymbol{e}}^{\mathrm{t}})$ for square, invertible, $\bar{\boldsymbol{A}}$
  - Transform each **additional** sample $(\boldsymbol{a}, b = \langle \boldsymbol{s}, \boldsymbol{a} \rangle + e)$ to

$$\boldsymbol{a}' = -\bar{\boldsymbol{A}}^{-1} \cdot \boldsymbol{a}, b' = b + \langle \bar{\boldsymbol{b}}, \boldsymbol{a}' \rangle = \langle \bar{\boldsymbol{e}}, \boldsymbol{a}' \rangle + e$$

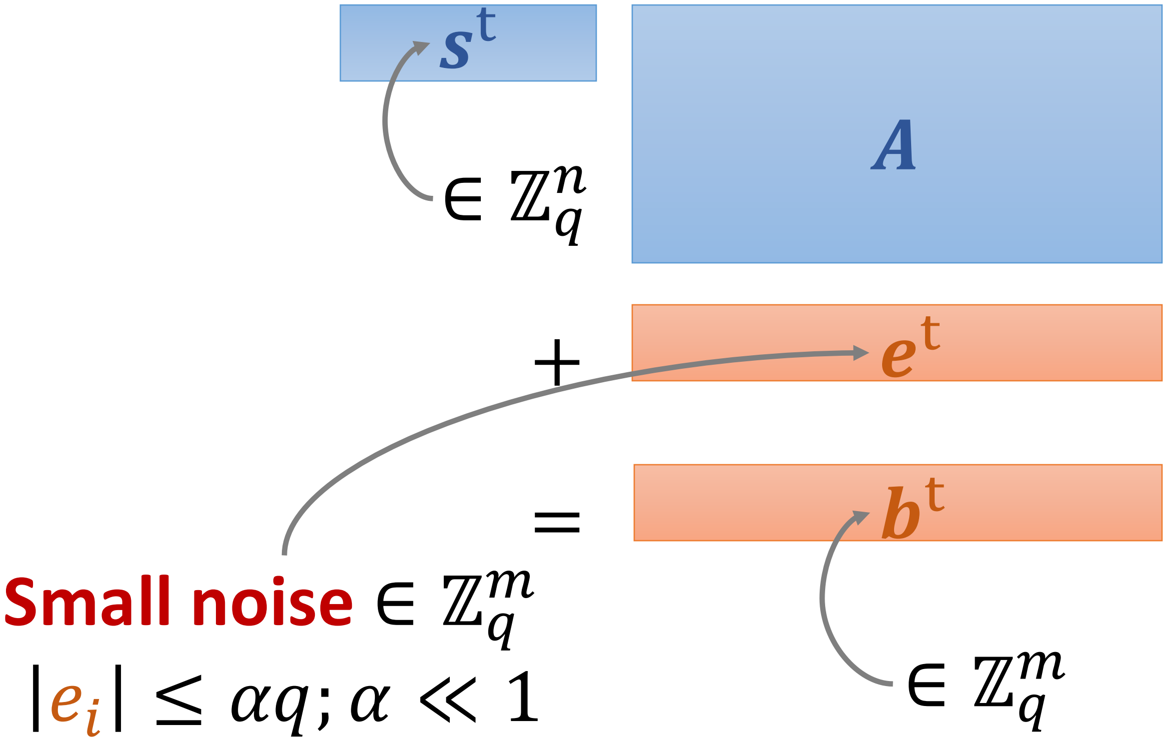
  - This maps **uniform** $(\boldsymbol{a}, b)$ to **uniform** $(\boldsymbol{a}', b')$, and thus works for **decision** LWE too

LWE vs SIS

- SIS has **many** valid solutions, whereas LWE only has **one**
- $\mathbf{LWE \leq SIS}$
  - Given $\boldsymbol{z}$ such that $\boldsymbol{A} \cdot \boldsymbol{z} = \mathbf{0}$ from an SIS oracle, compute $\boldsymbol{b}^{\mathrm{t}} \cdot \boldsymbol{z}$
  - Now, $\boldsymbol{b}^{\mathrm{t}} \cdot \boldsymbol{z} = \boldsymbol{e}^{\mathrm{t}} \cdot \boldsymbol{z}$ is **small** in the LWE case, whereas $\boldsymbol{b}^{\mathrm{t}} \cdot \boldsymbol{z}$ is **well-spread** in case $\boldsymbol{b}^{\mathrm{t}}$ is uniformly random
- What about the other direction?
  - Not known **in general**
  - True under **quantum reductions**

Efficiency of LWE/SIS

- Getting **one** random-looking scalar $b_i \in \mathbb{Z}_q$ requires an n-dimensional **inner product** $\mathrm{mod}\ q$



- Can **amortize** each column $\boldsymbol{a}_i$ over **many secrets** $\boldsymbol{s}_j$, but the latter still requires $\tilde{O}(n)$ work per scalar output
- Public keys are **rather large**, i.e. $> n^2$ time to encrypt/decrypt an n-bit message
- Can we do better?

Wishful Thinking…

$$\boldsymbol{s}^{\mathrm{t}} \star \boldsymbol{a}^{\mathrm{t}} + \boldsymbol{e}^{\mathrm{t}} = \boldsymbol{b}^{\mathrm{t}} \in \mathbb{Z}_q^d$$

- Get d **pseudorandom** scalars from just one **cheap product** operation $\star$
- Replace $\mathbb{Z}_q^{d\times d}$ **chunks** with $\mathbb{Z}_q^d$

- **Main question:** How to define the product $\star$ so that $(\boldsymbol{a}, \boldsymbol{b})$ is **pseudorandom**
  - Requires care: **coordinate-wise** product **insecure** for **small** errors
- **Answer:** Let $\star$ be multiplication **in a polynomial ring**, e.g. $\mathbb{Z}_q^d[X]/(X^d + 1)$
  - **Fast** and **practical** with the FFT: $d \log d$ operations mod q
  - The same **ring structure** used in NTRU [HPS08]

LWE over Rings/Modules

- Let $R = \mathbb{Z}[X]/(X^d + 1)$ for d a power of 2 and $R_q = R/qR$
  - Elements of R_q are degree $< d$ **polynomials** with coefficients $\bmod q$
  - Operations over R_q are **very efficient** using FFT-like algorithms
- **Search LWE:** Find secret vector of **polynomials** $\boldsymbol{s}$ in R_q^k given

$$\boldsymbol{s}^{\mathrm{t}} \star \boldsymbol{a}_i^t + \boldsymbol{e}_i^t = \boldsymbol{b}_i^t$$

$\in \mathbb{Z}_q^d$

- Each equation is d **related equations** on a secret of dimension $n = d \cdot k$
  - LWE: $d = 1, k = n$
  - Ring-LWE: $d = n, k = 1$
  - Module-LWE: Interpolate
- **Decision LWE:** Distinguish $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ from uniform $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ in $R_q^k \times R_q$

Hardness of Ring/Module-LWE

- **Theorem [LPR10]:** For any $R = \mathcal{O}_K$

$$R^k\text{–}\mathbf{GapSVP} \leq \mathbf{search}\ R^k\text{–}\mathbf{LWE}\ \leq \mathbf{decision}\ R^k\text{–}\mathbf{LWE}$$

- Can we **dequantize** the worst-case/average-case reduction?
  - The **classical** $\mathbf{GapSVP} \leq \mathbf{LWE}$ reduction is of little use: for the relevant factors, $\mathbf{GapSVP}$ for **ideals** (i.e., $k = 1$) is **easy**
- How **hard** (or not) is $\mathbf{GapSVP}$ on **ideal/module lattices**?
  - For **polynomial approximation** no significant improvement versus general lattices (even for ideals)
  - For **subexponential approximation** we have better **quantum** algorithms for **ideals**, but not for $k > 1$
- **Reverse** reductions? Seems not **without** increasing k…

Why Lattice-based Cryptography?

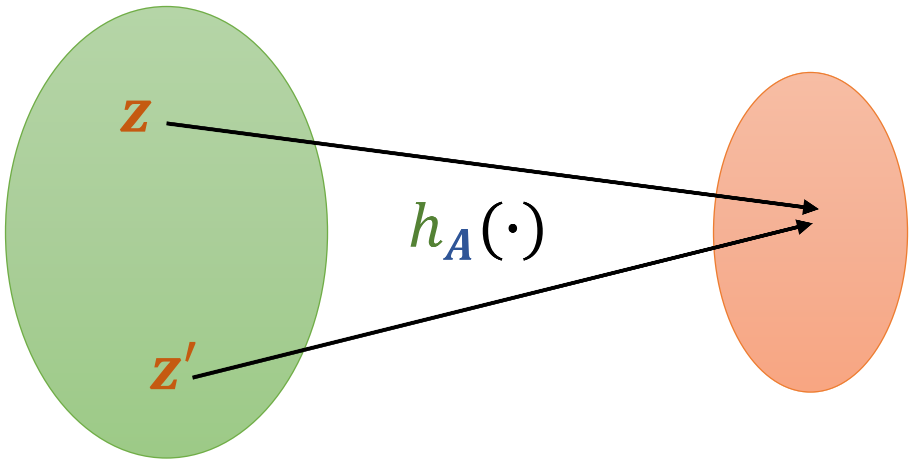
- **Provable** security
  - If scheme is **not secure**, one **can solve** hard mathematical problems
  - Not always happens in current implementations (e.g., RSA)
- **Worst-case** security
  - If scheme not secure, one can break **every** instance of lattice problems
  - Factoring and discrete log only guarantee **average-case** security
- Still **unbroken** by quantum algorithms
  - No progress over the last 50 years
  - But we don’t know: see https://eprint.iacr.org/2024/555
- Efficiency
  - Mainly additions/multiplications, no modular exponentiations

Basic Cryptographic Applications

One-Way Functions

- Parameters $m, n, q \in \mathbb{Z}$, key $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$
- Input $\boldsymbol{x} \in \{0,1\}^m$, output $f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A} \cdot \boldsymbol{x}$
- **Theorem [Ajt96]:** For $m > n \log q$, if $\mathbf{SIVP}$ is **hard** to approximate in the **worst-case**, then $f_{\boldsymbol{A}}$ is **one-way**
- Cryptanalysis: Given $\boldsymbol{A}, \boldsymbol{y}$, find $\boldsymbol{x}$ such that $\boldsymbol{y} = \boldsymbol{A} \cdot \boldsymbol{x}$
  - **Easy** problem: find **arbitrary** $\boldsymbol{u}$ such that $\boldsymbol{y} = \boldsymbol{A} \cdot \boldsymbol{u}$
  - All solutions $\boldsymbol{y} = \boldsymbol{A} \cdot \boldsymbol{x}$ are of the form $\boldsymbol{t} + \mathcal{L}^{\perp}(\boldsymbol{A})$
  - Requires to find **small** vector in $\boldsymbol{t} + \mathcal{L}^{\perp}(\boldsymbol{A})$ or to find a lattice point $\boldsymbol{v} \in \mathcal{L}^{\perp}(\boldsymbol{A})$ **close** to $\boldsymbol{t}$ (**average-case** instance of CVP w.r.t. $\mathcal{L}^{\perp}(\boldsymbol{A})$)

Collision-resistant Hash Functions



Collisions **exists inherently**, but are hard to find **efficiently**

- Given $\boldsymbol{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m)$, define $h_{\boldsymbol{A}}: \{0,1\}^m \to \mathbb{Z}_q^n$

$$h_{\boldsymbol{A}}(z_1, \ldots, z_m) = \boldsymbol{a}_1 \cdot z_1 + \cdots + \boldsymbol{a}_m \cdot z_m$$

  - Set $m > n \log q$ in order to get **compression**
  - A collision $\boldsymbol{a}_1 \cdot z_1 + \cdots + \boldsymbol{a}_m \cdot z_m = \boldsymbol{a}_1 \cdot z_1' + \cdots + \boldsymbol{a}_m \cdot z_m'$ yields $\boldsymbol{a}_1 \cdot (z_1 - z_1') + \cdots + \boldsymbol{a}_m \cdot (z_m - z_m') = 0$, with $z_m - z_m' \in \{-1,0,1\}$

Commitments

- Analogy: **lock** message in a box, give the box, keep the key
  - Later give the key to **open** the box
- Implementation:
  - **Randomized** function $\mathbf{Com}(x; r)$, where x is the message and r is the randomness
  - To **open** a commitment simply reveal (x, r)
- Security properties
  - **Hiding:** $\mathbf{Com}(x; r)$ **reveals nothing** on x
  - **Binding: Can't open** $\mathbf{Com}(x; r)$ to $x' \neq x$

Commitments

- Take two **random** SIS matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$
- The **message** is $\boldsymbol{x} \in \{0,1\}^m$ and the **randomness** is $\boldsymbol{r} \in \{0,1\}^m$
- Commitment: $\mathbf{Com}(\boldsymbol{x}; \boldsymbol{r}) = f_{\boldsymbol{A}_1,\boldsymbol{A}_2}(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{A}_1 \cdot \boldsymbol{x} + \boldsymbol{A}_2 \cdot \boldsymbol{r}$
  - **Hiding:** $\boldsymbol{A}_2 \cdot \boldsymbol{r} = f_{\boldsymbol{A}_2}(\boldsymbol{r})$ is **statistically** close to **uniform** over $\mathbb{Z}_q^n$, and thus $\boldsymbol{x}$ is information-theoretically **hidden**
  - **Binding:** Finding $(\boldsymbol{x}, \boldsymbol{r})$ and $(\boldsymbol{x}', \boldsymbol{r}')$ such that $\mathbf{Com}(\boldsymbol{x}; \boldsymbol{r}) = \mathbf{Com}(\boldsymbol{x}'; \boldsymbol{r}')$ directly contradicts the **collision resistance** of $f_{\boldsymbol{A}_1,\boldsymbol{A}_2}$

Leftover Hash Lemma

- Let $\mathcal{H}$ be a family of **universal hash functions** with domain $\mathcal{D}$ and image $\mathcal{I}$. Then, for $x \leftarrow_\$ \mathcal{D}$, $h \leftarrow_\$ \mathcal{H}$, and $u \leftarrow_\$ \mathcal{I}$:
$\mathbb{SD}\Big(\big(h, h(x)\big); (h, u)\Big) \leq 1/2 \cdot \sqrt{|\mathcal{I}|/|\mathcal{D}|}$
- Note that the function $h_{\boldsymbol{A}}(\boldsymbol{r}) = [\boldsymbol{A} \cdot \boldsymbol{r}]_q$ is **universal**
  - As $\forall \boldsymbol{r}_1 \neq \boldsymbol{r}_2$: $\mathbb{P}_{\boldsymbol{A}}[h_{\boldsymbol{A}}(\boldsymbol{r}_1) = h_{\boldsymbol{A}}(\boldsymbol{r}_2)] = \mathbb{P}_{\boldsymbol{A}}[\boldsymbol{A} \cdot (\boldsymbol{r}_1 - \boldsymbol{r}_2) = \boldsymbol{0}] = q^{-n}$
- Hence, for $\boldsymbol{r} \leftarrow_\$ \{0,1\}^m$, $\boldsymbol{A} \leftarrow_\$ \mathbb{Z}_q^{n \times m}$, and $\boldsymbol{u} \leftarrow_\$ \mathbb{Z}_q^n$, whenever $m = 2 + n \log q + 2n$

$$\mathbb{SD}\Big(\big(\boldsymbol{A}, [\boldsymbol{A} \cdot \boldsymbol{r}]_q\big); (\boldsymbol{A}, \boldsymbol{u})\Big) \leq 1/2 \cdot \sqrt{q^n/2^m} \leq 2^{-n}$$

Pseudorandom Functions [GGM84]

- Family $\mathcal{F} = \{F_s\colon \{0,1\}^k \to \mathcal{D}\}$ s.t. querying F_s, for **random** s, is indistinguishable from querying **random function** U

- Countless applications: **secret-key** encryption, message **authentication** codes, secure **identification**, …

Constructing PRFs

- **Heuristically**: AES, etc.
  - Fast, secure against **known** cryptanalytic attacks, **not** provably secure
- From **any OWF** [GGM84]:
  - For **any** length-doubling **PRG** $G(s) = (G_0, G_1)$, let

$$F_s(x_1, \ldots, x_k) = G_{x_k}(\cdots G_{x_1}(s) \cdots)$$

  - **Provably** secure
  - Inherently **sequential** (i.e., $\geq k$ iterations)
- From **any synthesizer** [NR95,NR97,NRR00]
  - **Low depth**: NC^1, NC^2 or TC^0 (i.e., $O(1)$ depth with **threshold** gates)
  - **Provably** secure

Synthetisers [NR95]

- A **deterministic** function $S: \mathcal{D} \times \mathcal{D} \to \mathcal{D}$ such that for any polynomial m, and for **uniform** $a_1, \dots, a_m, b_1, \dots, b_m \in \mathcal{D}$

$$\{S(a_i, b_j)\} \approx \{U_{i,j}\}$$

Uniform distribution over $\mathcal{D}^{m \times m}$

	b_1	b_2	$\dots$
a_1	$S(a_1, b_1)$	$S(a_1, b_2)$	
a_2	$S(a_2, b_1)$	$S(a_2, b_2)$	
$\dots$			

$\approx$

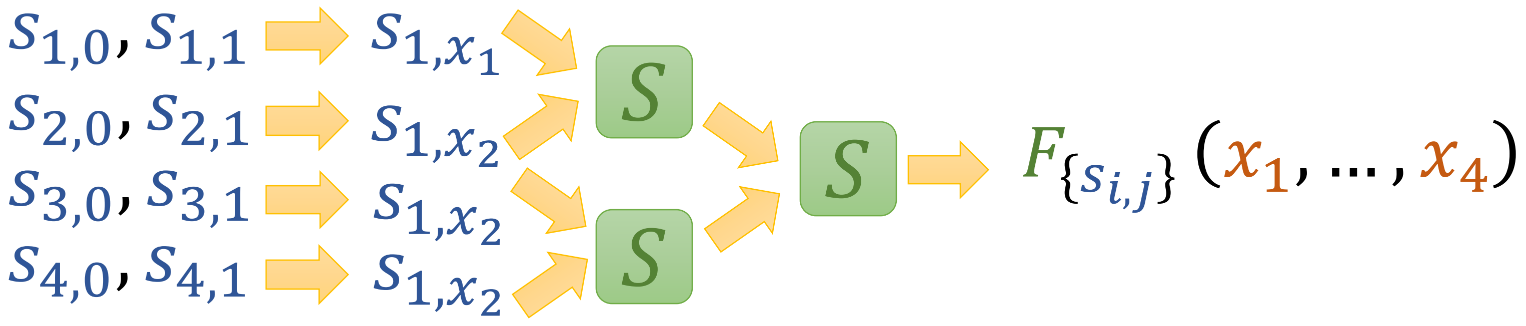
	b_1	b_2	$\dots$
a_1	$U_{1,1}$	$U_{1,2}$	
a_2	$U_{2,1}$	$U_{2,2}$	
$\dots$			

  - An almost **length-squaring** PRG with **locality**

PRFs from Synthetisers [NR95]

- **Base case:** One-bit PRF $F_{s_0,s_1}(x) = s_x \in \mathcal{D}$
- **Inductive step:** Given a k-bit PRF family $\mathcal{F} = \{F_s : \{0,1\}^k \to \mathcal{D}\}$ define $F_{s_L,s_R} : \{0,1\}^{2k} \to \mathcal{D}$

$$F_{s_L,s_R}(x_L, x_R) = S(F_{s_L}(x_L), F_{s_R}(x_R))$$



- **Security:** Every query to $F_{s_L}(x_L), F_{s_R}(x_R)$ defines **pseudorandom** inputs $a_1, \ldots, a_m, b_1, \ldots, b_m$ for the synthetiser

Synthetisers from LWE?

- Hard to **tell apart** $(\boldsymbol{a}_i, b_i = \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i)$ from **random** $(\boldsymbol{a}, b)$
- By a **hybrid argument**, the following are **pseudorandom**

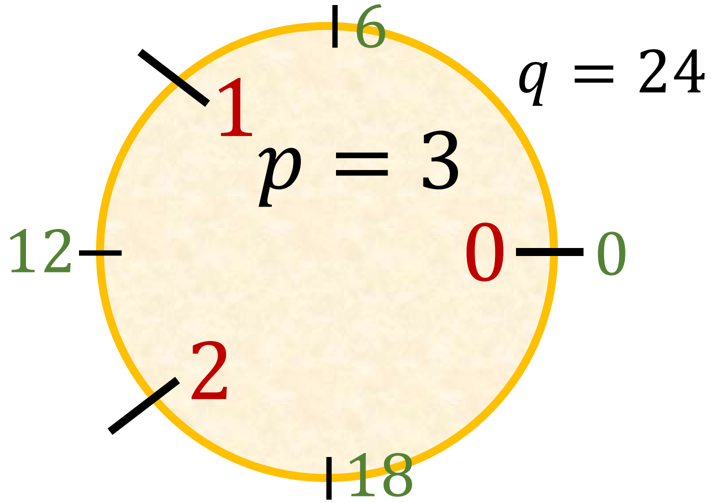
$$\boldsymbol{A}_i \in \mathbb{Z}_q^n, \boldsymbol{A}_i \cdot \boldsymbol{S}_1 + \boldsymbol{E}_{1,1} \in \mathbb{Z}_q^{n \times n}, \boldsymbol{A}_i \cdot \boldsymbol{S}_2 + \boldsymbol{E}_{2,1} \in \mathbb{Z}_q^{n \times n}, \ldots$$

- This suggests the following synthetiser from LWE

	$\boldsymbol{S}_1$	$\boldsymbol{S}_2$	…
$\boldsymbol{A}_1$	$\boldsymbol{A}_1 \cdot \boldsymbol{S}_1 + \boldsymbol{E}_{1,1}$	$\boldsymbol{A}_1 \cdot \boldsymbol{S}_2 + \boldsymbol{E}_{1,2}$	
$\boldsymbol{A}_2$	$\boldsymbol{A}_2 \cdot \boldsymbol{S}_1 + \boldsymbol{E}_{2,1}$	$\boldsymbol{A}_2 \cdot \boldsymbol{S}_2 + \boldsymbol{E}_{2,2}$	
…			

  - But synthetisers must be **deterministic**!

Learning with Rounding [BPR12]



- Generate errors **deterministically**
  - Round $\mathbb{Z}_q$ to a **sparse** subset $\mathbb{Z}_p$
  - For $p < q$, let $\lfloor x \rceil_p = \lfloor (p/q) \cdot x \rceil \bmod p$
- The LWR problem: Tell apart $(\boldsymbol{a}, b = \lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle \rceil_p) \in \mathbb{Z}_q \times \mathbb{Z}_p$ from **random** $(\boldsymbol{a}, b)$
  - LWE **conceals** low-order bits by adding **small random error**
  - LWR just **discards** those bits instead
- $\mathbf{LWE} \leq \mathbf{LWR}$ for $q \geq p \cdot n^{\omega(1)}$ (seems 2^n-hard for $q \geq p \cdot \sqrt{n}$)
  - Proof idea: w.h.p. $(\boldsymbol{a}, \lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e \rceil_p) \approx (\boldsymbol{a}, \lfloor \langle \boldsymbol{a}, \boldsymbol{s} \rangle \rceil_p)$ and $\left(\boldsymbol{a}, \left\lfloor U(\mathbb{Z}_q) \right\rceil_p\right) \approx (\boldsymbol{a}, U(\mathbb{Z}_p))$ where $U(\mathbb{Z}_q)$ is uniform over $\mathbb{Z}_q$
  - Reduction with Improved parameters in [AKPW13]

Synthetiser-based PRF from LWR

- Synthetiser: $S: \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times n} \to \mathbb{Z}_p^{n\times n}$ is $S(\boldsymbol{A}, \boldsymbol{S}) = \lfloor \boldsymbol{A} \cdot \boldsymbol{S} \rceil_p$
  - Note that the range $\mathbb{Z}_p$ is **slightly smaller** than the domain $\mathbb{Z}_q$
- Construction of PRF with domain $\{0,1\}^k$ for $k = 2^d$
  - **Tower** of power moduli $q_d > q_{d-1} > \cdots > q_0$
  - The secret key is $2k$ matrices $\boldsymbol{S}_{i,b} \in \mathbb{Z}_{q_d}^{n\times n}$, for $i \in [k], b \in \{0,1\}$
  - Depth $d = \log k$ of LWR synthetisers

$$\left\lfloor \left\lfloor \left\lfloor \boldsymbol{S}_{1,x_1} \cdot \boldsymbol{S}_{2,x_2} \right\rceil_{q_2} \cdot \left\lfloor \boldsymbol{S}_{3,x_3} \cdot \boldsymbol{S}_{4,x_4} \right\rceil_{q_2} \right\rceil_{q_1} \cdot \left\lfloor \left\lfloor \boldsymbol{S}_{5,x_5} \cdot \boldsymbol{S}_{6,x_6} \right\rceil_{q_2} \cdot \left\lfloor \boldsymbol{S}_{7,x_7} \cdot \boldsymbol{S}_{8,x_8} \right\rceil_{q_2} \right\rceil_{q_1} \right\rceil_{q_0}$$

  - Each synthetiser is in NC^1, and thus the PRF is in NC^2

Direct Construction

- Simple **direct** PRF construction from DDH [NR97,NRR00]:

$$F_{g,s_1,\ldots,s_k}(x_1,\ldots,x_k) = g^{\prod_i {s_i}^{x_i}}$$

  - This can be implemented in $TC^0 \subseteq NC^0$ (albeit with **huge** circuit)
- Direct construction from LWE
  - Public moduli $q > p$
  - The secret key is **uniform** $\boldsymbol{A}$ and **short** $\boldsymbol{S}_1,\ldots,\boldsymbol{S}_k$ over $\mathbb{Z}_q$
  - The PRF evaluates a **rounded subset-product** function

$$F_{\boldsymbol{A},\boldsymbol{S}_1,\ldots,\boldsymbol{S}_k}(x_1,\ldots,x_k) = \left\lfloor \boldsymbol{A} \cdot \prod_i \boldsymbol{S}_i^{x_i} \right\rceil_p$$

Proof Sketch

- Similar to the $\mathbf{LWE} \leq \mathbf{LWR}$ proof
- Thought experiment: answer queries with

$$\tilde{F}_{\boldsymbol{A},\boldsymbol{S}_1,\ldots,\boldsymbol{S}_k}(x_1,\ldots,x_k) = \left\lfloor (\boldsymbol{A} \cdot \boldsymbol{S}_1^{x_1} + x_1 \cdot \boldsymbol{E}) \cdot \boldsymbol{S}_2^{x_2} \cdot \ldots \cdot \boldsymbol{S}_k^{x_k} \right\rceil_p$$

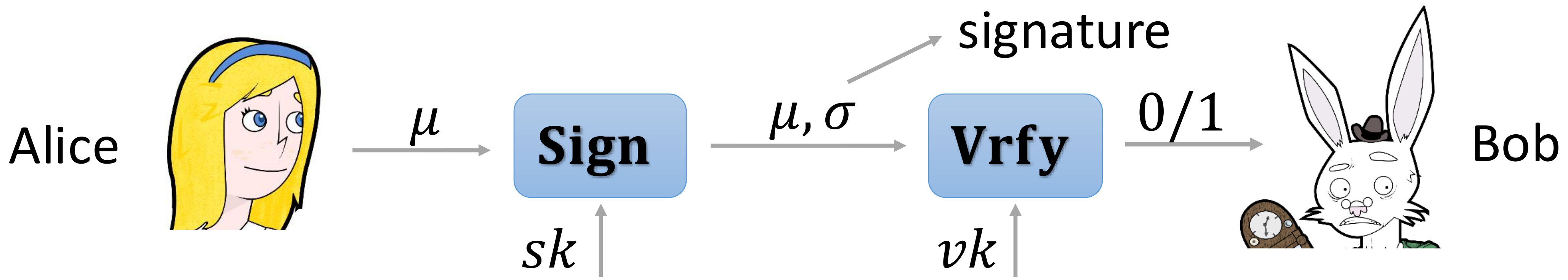
$$= \left\lfloor \boldsymbol{A} \cdot \prod_{i=1}^{k} \boldsymbol{S}_i^{x_i} + x_1 \cdot \boldsymbol{E} \cdot \prod_{i=2}^{k} \boldsymbol{S}_i^{x_i} \right\rceil_p$$

  - W.h.p. $\tilde{F}(x) = F(x)$ due to **small error** and **rounding**
- Using LWE replace $(\boldsymbol{A}, \boldsymbol{A} \cdot \boldsymbol{S}_1 + \boldsymbol{E})$ with uniform $(\boldsymbol{A}_0, \boldsymbol{A}_1)$
  - New function $F(x) = \left\lfloor \boldsymbol{A}_{x_1} \cdot \boldsymbol{S}_2^{x_2} \cdot \ldots \cdot \boldsymbol{S}_k^{x_k} \right\rceil_p$
  - Repeat for $\boldsymbol{S}_2, \ldots, \boldsymbol{S}_k$ to get $F'^{\cdots\prime}(x) = \lfloor \boldsymbol{A}_x \rceil_p = U(x)$

NIST Standards

Falcon

Digital Signatures



- Syntax $\Pi = (\mathbf{KGen}, \mathbf{Sign}, \mathbf{Vrfy})$
  - $\mathbf{KGen}(1^\lambda)$: Takes the **security parameter** $\lambda \in \mathbb{N}$, and outputs (vk, sk)
  - $\mathbf{Sign}(sk, \mu)$: Takes plaintext μ, and outputs a **signature** σ
  - $\mathbf{Vrfy}(vk, \mu, \sigma)$: Takes plaintext μ and signature σ, and outputs a **bit**
- **Correctness:** $\forall \lambda \in \mathbb{N}, \forall (vk, sk) \in \mathbf{KGen}(1^\lambda), \forall \mu$

$$\mathbb{P}[\mathbf{Vrfy}(vk, \mathbf{Sign}(sk, \mu)) = 1] = 1$$

Lattice Trapdoors

- Recall: Lattice-based **one-way functions**

$$f_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A} \cdot \boldsymbol{x} \bmod q \in \mathbb{Z}_q^n$$

(short $\boldsymbol{x}$, surjective)

$$f_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{e}) = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}} \bmod q \in \mathbb{Z}_q^m$$

(short $\boldsymbol{e}$, injective)

- Task: **Invert** $f_{\boldsymbol{A}}$
  - Find the **unique** $\boldsymbol{s}$ (or $\boldsymbol{e}$) such that $f_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{e}) = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}} \bmod q$
  - Given $\boldsymbol{u} = f_{\boldsymbol{A}}(\boldsymbol{x}') = \boldsymbol{A} \cdot \boldsymbol{x}' \bmod q$, **sample random** $\boldsymbol{x} \leftarrow f_{\boldsymbol{A}}^{-1}(\boldsymbol{u})$ with probability proportional to $\exp(-\|\boldsymbol{x}\|^2/s^2)$
- How? Via a **strong trapdoor** for $\boldsymbol{A}$ (a **short basis** of $\mathcal{L}^{\perp}(\boldsymbol{A})$)
  - Deeply studied question [Babai86,Ajtai99,Klein01,GPV08,AP09,P10]

A Different Kind of Trapdoor [MP12]

- Drawbacks of previous solutions
  - Generating $\boldsymbol{A}$ with short basis is **complex** and **slow**
  - Inversion algorithms trade-off **quality** (i.e., length of basis vectors which depends on the Gaussian std parameter s) for **efficiency**
- Alternative: The trapdoor is **not a basis**
  - But just **as powerful**
  - **Simpler** and **faster**
- Overview of method
  - Start with **fixed**, **public**, lattice defined by **gadget matrix** $\boldsymbol{G}$ which admits very **fast**, and **parallel**, algorithms for $f_{\boldsymbol{G}}^{-1}$
  - **Randomize** $\boldsymbol{G}$ into $\boldsymbol{A}$ via nice **unimodular** transform (the trapdoor)
  - **Reduce** $f_{\boldsymbol{A}}^{-1}$ to $f_{\boldsymbol{G}}^{-1}$ plus some pre/post-processing

Step 1: The Gadget Matrix

- Let $q = 2^k$ and take $\boldsymbol{g} = [1 \quad 2 \quad \cdots \quad 2^{k-1}] \in \mathbb{Z}_q^{1\times k}$
- To invert $f_{\boldsymbol{g}}: \mathbb{Z}_q \times \mathbb{Z}^k \to \mathbb{Z}_q^k$

$$f_{\boldsymbol{g}}(s, \boldsymbol{e}) = s \cdot \boldsymbol{g} + \boldsymbol{e} = [s + e_0 \quad 2s + e_1 \quad \cdots \quad 2^{k-1}s + e_{k-1}] \bmod q$$

  - Get lsb of s from $2^{k-1}s + e_{k-1}$, then repeat for the next bits of s
  - Works when $e_{k-1} \in [-q/4, q/4)$
- To sample Gaussian preimage for $u = f_{\boldsymbol{g}}(\boldsymbol{x}) = \langle \boldsymbol{g}, \boldsymbol{x} \rangle$
  - For $i \in [0, k-1]$, choose $x_i \leftarrow (2\mathbb{Z} + u)$ and let $u \leftarrow (u - x_i)/2 \in \mathbb{Z}$
  - E.g., $k = 2$: $x_0 \leftarrow (2z_0 + u)$, $u \leftarrow (u - 2z_0 - u)/2 = -z_0$, $x_1 \leftarrow (2z_1 - z_0)$, $\langle \boldsymbol{g}, \boldsymbol{x} \rangle = 2z_0 + u + 2(2z_1 - z_0) = u + 4z_1 = u \bmod 4$

Step 1: The Gadget Matrix $\boldsymbol{G}$

- Alternative view: The **lattice** $\mathcal{L}^{\perp}(\boldsymbol{g})$ has **basis**

$$\boldsymbol{S} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & \ddots & 2 & \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{Z}^{k\times k}, \text{with } \tilde{\boldsymbol{S}} = 2 \cdot \boldsymbol{I}_k$$

  - The above inversion algorithms are special cases of the randomized **nearest-plan algorithm** [Bab86,Kle01,GPV08]
- Define $\boldsymbol{G} = \boldsymbol{I}_n \otimes \boldsymbol{g} \in \mathbb{Z}^{n\times nk}$ (where $\otimes$ is the **tensor** product)
  - Computing $f_{\boldsymbol{G}}^{-1}$ reduces to n **parallel calls** to $f_{\boldsymbol{g}}^{-1}$
  - Also applies to $\boldsymbol{H} \cdot \boldsymbol{G}$, for any **invertible** $\boldsymbol{H} \in \mathbb{Z}_q^{n\times n}$

Step 2: Randomize $\boldsymbol{G}$

- Define **semi-random** $[\bar{\boldsymbol{A}}|\boldsymbol{G}]$ for **uniform** $\bar{\boldsymbol{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$
  - It can be seen that inverting $f^{-1}_{[\bar{\boldsymbol{A}}|\boldsymbol{G}]}$ **reduces** to inverting $f^{-1}_{\boldsymbol{G}}$ [CHKP10]
- Choose a **short Gaussian** $\boldsymbol{R} \in \mathbb{Z}^{\bar{m}\times n\log q}$ and let

$$\boldsymbol{A} = [\bar{\boldsymbol{A}}|\boldsymbol{G}] \cdot \begin{bmatrix} \boldsymbol{I} & \boldsymbol{R} \\ & \boldsymbol{I} \end{bmatrix} = [\bar{\boldsymbol{A}}|\boldsymbol{G} - \bar{\boldsymbol{A}}\boldsymbol{R}]$$

  - $\boldsymbol{A}$ is **uniform** because, by the **leftover hash lemma**, $[\bar{\boldsymbol{A}}|\bar{\boldsymbol{A}}\boldsymbol{R}]$ is **statistically close** to uniform when $\bar{m} \approx n\log q$
  - Alternatively, $[\boldsymbol{I}|\bar{\boldsymbol{A}}| - \bar{\boldsymbol{A}} \cdot \boldsymbol{R}_1 + \boldsymbol{R}_2]$ is **pseudorandom** under the LWE assumption (in normal form)

A New Trapdoor Notion

- We constructed $\boldsymbol{A} = [\bar{\boldsymbol{A}} | \boldsymbol{G} - \bar{\boldsymbol{A}}\boldsymbol{R}]$
- Say that $\boldsymbol{R}$ is a **trapdoor** for $\boldsymbol{A}$ with **tag** $\boldsymbol{H} \in \mathbb{Z}_q^{n \times n}$ (invertible) if

$$\boldsymbol{A} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} = \boldsymbol{H} \cdot \boldsymbol{G}$$

  - The **quality** of $\boldsymbol{R}$ is $s_1(\boldsymbol{R}) = \max_{\boldsymbol{u}:\|\boldsymbol{u}\|=1} \|\boldsymbol{R} \cdot \boldsymbol{u}\|$
  - **Fact:** $s_1(\boldsymbol{R}) \approx (\sqrt{\mathrm{rows}} + \sqrt{\mathrm{cols}}) \cdot r$ for Gaussian entries w/ std dev r
  - Also $\boldsymbol{R}$ is a trapdoor for $\boldsymbol{A} - [\boldsymbol{0} | \boldsymbol{H}' \cdot \boldsymbol{G}]$ with tag $\boldsymbol{H} - \boldsymbol{H}'$ [ABB10]
- Relating new and old trapdoors
  - Given basis $\boldsymbol{S}$ for $\mathcal{L}^\perp(\boldsymbol{G})$ and trapdoor $\boldsymbol{R}$ for $\boldsymbol{A}$, one can **efficiently** construct **basis** $\boldsymbol{S}_{\boldsymbol{A}}$ for $\mathcal{L}^\perp(\boldsymbol{G})$ where $\|\tilde{\boldsymbol{S}}_{\boldsymbol{A}}\| \leq (s_1(\boldsymbol{R}) + 1) \cdot \|\tilde{\boldsymbol{S}}\|$

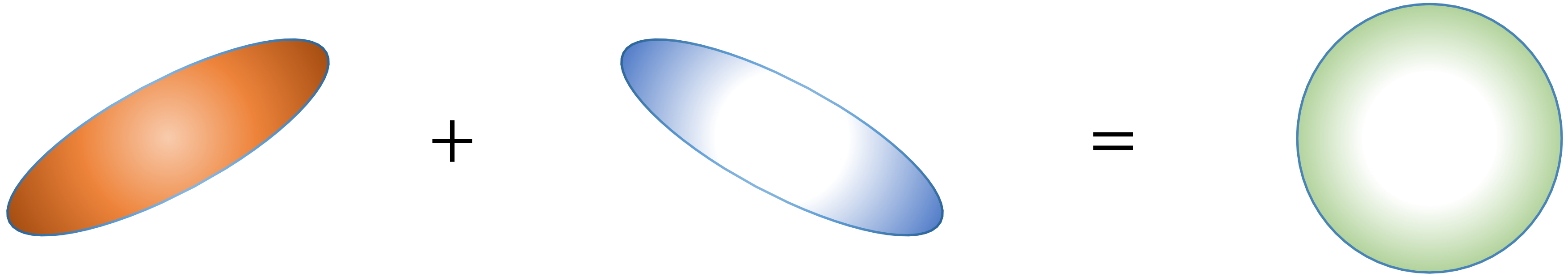
Step 3: Reduce $f_{\boldsymbol{A}}^{-1}$ to $f_{\boldsymbol{G}}^{-1}$

- Let $\boldsymbol{R}$ be a **trapdoor** for $\boldsymbol{A}$ with **tag** $\boldsymbol{H} = \boldsymbol{I}$: $\boldsymbol{A} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} = \boldsymbol{G}$
- Inverting LWE
  - Given $\boldsymbol{b}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}$, recover $\boldsymbol{s}$ from $\boldsymbol{b}^{\mathrm{t}} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{G} + \boldsymbol{e}^{\mathrm{t}} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix}$
  - Works if **each entry** of $\boldsymbol{e}^{\mathrm{t}} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \in [-q/4, q/4)$
- Inverting SIS
  - Given $\boldsymbol{u}$, sample $\boldsymbol{z} \leftarrow f_{\boldsymbol{G}}^{-1}(\boldsymbol{u})$ and output $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \cdot \boldsymbol{z} \in f_{\boldsymbol{A}}^{-1}(\boldsymbol{u})$
  - Indeed, $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{G} \cdot \boldsymbol{z} = \boldsymbol{u}$

Leaks about $\boldsymbol{R}$!

$$\boldsymbol{\Sigma} = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{x} \cdot \boldsymbol{x}^{\mathrm{t}}] = \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{R} \cdot \boldsymbol{z} \cdot \boldsymbol{z}^{\mathrm{t}} \cdot \boldsymbol{R}^{\mathrm{t}}] \approx \boldsymbol{R} \cdot \boldsymbol{R}^{\mathrm{t}}$$

Step 3: Perturbation Method [P10]



$\boldsymbol{\Sigma}_1$ $\qquad$ $\boldsymbol{\Sigma}_2 = s^2 \cdot \boldsymbol{I} - \boldsymbol{\Sigma}_1$ $\qquad$ $\boldsymbol{\Sigma} = s^2 \cdot \boldsymbol{I}$

$$\boldsymbol{u}^{\mathrm{t}} \cdot \boldsymbol{\Sigma}_2 \cdot \boldsymbol{u} = s^2 - \boldsymbol{u}^{\mathrm{t}} \cdot \boldsymbol{\Sigma}_1 \cdot \boldsymbol{u} > 0$$

- To **fix** the covariance
  - Generate **perturbation** vector $\boldsymbol{p}$ with covariance $s^2 \cdot \boldsymbol{I} - \boldsymbol{R} \cdot \boldsymbol{R}^{\mathrm{t}}$
  - Sample **spherical** $\boldsymbol{z}$ such that $\boldsymbol{G} \cdot \boldsymbol{z} = \boldsymbol{u} - \boldsymbol{A} \cdot \boldsymbol{p}$
  - Output $\boldsymbol{x} = \boldsymbol{p} + \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \cdot \boldsymbol{z}$

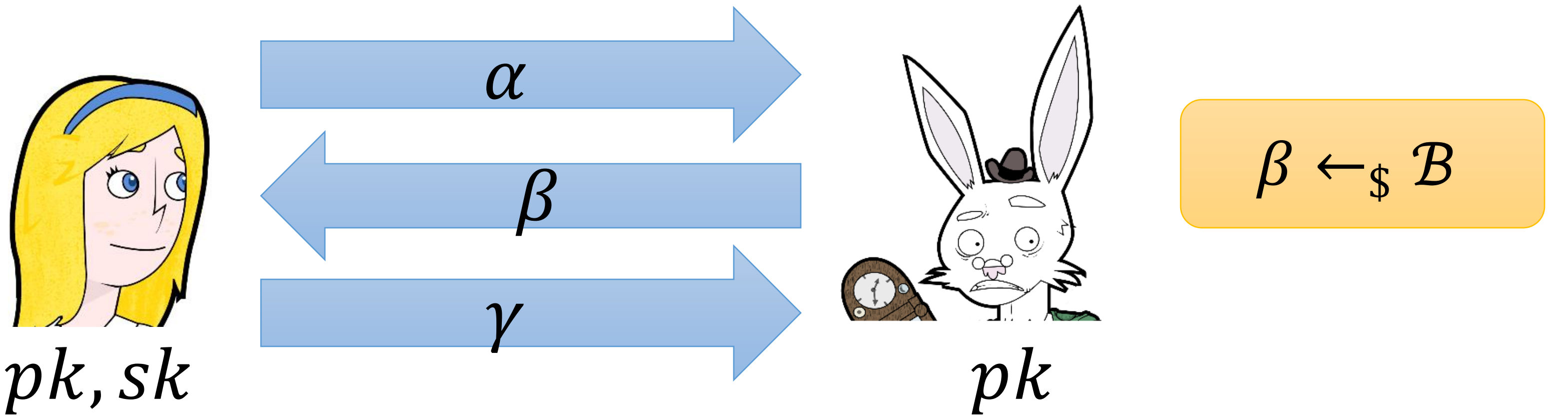
$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{A} \cdot \boldsymbol{p} + \boldsymbol{A} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \cdot \boldsymbol{z} = \boldsymbol{A} \cdot \boldsymbol{p} + \boldsymbol{G} \cdot \boldsymbol{z} = \boldsymbol{u}$$

Falcon: Digital Signatures from SIS

- Generate **uniform** $vk = \boldsymbol{A}$ with **trapdoor** $sk = \boldsymbol{T}$
- To sign μ, use $\boldsymbol{T}$ to **sample** $\sigma = \boldsymbol{x} \in \mathbb{Z}^m$ such that $\boldsymbol{A} \cdot \boldsymbol{x} = H(\mu)$, where H is a **public** hash function
  - Recall that $\boldsymbol{x}$ is drawn from a **Gaussian distribution**, which **reveals nothing** about the trapdoor $\boldsymbol{T}$
- To verify $(\mu, \sigma = \boldsymbol{x})$ under $vk = \boldsymbol{A}$ simply check $\boldsymbol{A} \cdot \boldsymbol{x} = H(\mu)$ and that $\boldsymbol{x}$ is **sufficiently short**
- Security: **Forging** a signature for a new message μ^* requires finding a **short** $\boldsymbol{x}^*$ such that $\boldsymbol{A} \cdot \boldsymbol{x}^* = H(\mu^*)$
  - This is **equivalent** to solving the SIS problem
  - Signatures queries **do not help** because they **reveal nothing** about the trapdoor $\boldsymbol{T}$

Crystals-Dilithium

Canonical Identification Schemes



- **Completeness:** The **honest** prover convinces the **honest** verifier (with all but a negligible probability)
- **Passive Security:** No (**efficient**) **malicious** prover knowing only pk can convince the **honest** verifier
  - Even in case the attacker knows many **accepting transcripts** corresponding to **honest** protocol executions

The Fiat-Shamir Transform

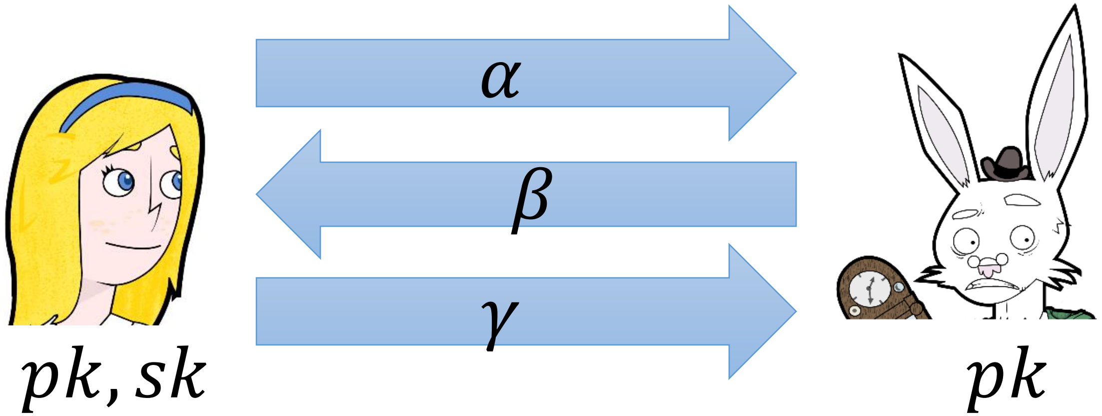
- Given a **canonical** ID scheme, we can derive a **signature scheme** as follows:
  - Alice obtains $\sigma = (\alpha, \gamma)$ from the **prover**, using the **secret key** sk and choosing $\beta = H(x, \alpha)$
  - Bob checks that (α, β, γ) is a **valid transcript**, with $\beta = H(x, \alpha)$

The Fiat-Shamir Transform

Theorem [FS86]. If the ID scheme is **passively** secure, the signature derived via the **Fiat-Shamir** transform is **UF-CMA**

- **Remark:** The original proof requires to model H as an **ideal** hash function (**random oracle**)
  - It is **debatable** in the community what such a proof means in **practice**
- Can we prove security in the **plain model** (i.e., no random oracles)?
  - Many **impossibility** results for **general** ID schemes
  - **Possible** for **some** classes of ID schemes assuming so-called **correlation intractability**

Sufficient Criteria for Passive Security



$\beta \leftarrow_{\$} \mathcal{B}$

- One can show the following criteria are **sufficient** for achieving **passive security**:
  - **Special soundness:** Given any pk and two **accepting** transcripts (α, β, γ) and $(\alpha, \beta', \gamma')$ for pk with $\beta \neq \beta'$, there is a polynomial-time algorithm **outputting** sk
  - **HVZK: Honest** proofs **reveal nothing** about the secret key sk

Proofs of Knowledge

- The **special soundness** property implies that any successful prover must essentially **know the secret key**
- In fact, any such prover can be used to **extract** the secret key:
  - Run the prover upon input pk in order to obtain a transcript (α, β, γ)
  - **Rewind** the prover after it already sent α and forward it **another random challenge** β', which yields a transcript $(\alpha, \beta', \gamma')$
  - As long as $\beta \neq \beta'$, **special soundness** allows us to obtain sk
- The above can be formalized, but the proof requires **some care**
  - Because the transcripts (α, β, γ) and $(\alpha, \beta', \gamma')$ are **correlated**

Honest-Verifier Zero-Knowledge

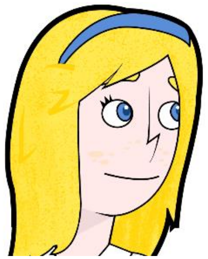
- How do we formalize that a trascript **reveals nothing** on sk?
  - This is tricky: transcripts shall not reveal even **one bit** of sk
- Require that honest transcripts can be **efficiently simulated** given just pk (but not sk)
  - Whatever the verifier could compute via the protocol, he could have computed by **talking to himself** (i.e., by running the simulator)
- A canonical ID scheme is **perfect honest-verifier zero-knowledge** (HVZK) if $\exists$ PPT $\mathcal{S}$ such that:

$$\big(pk, sk, \mathcal{S}(pk)\big) \equiv \big(pk, sk, \langle \mathcal{P}(pk, sk), \mathcal{V}(pk)\rangle\big)$$

Canonical ID Scheme from Discrete Log

$\text{params} = (\mathbb{G}, g, q)$

$a \leftarrow_\$ \mathbb{Z}_q$
$\gamma = \beta \cdot w + a$



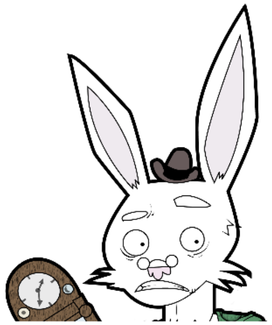
x, w

$\alpha = g^a \longrightarrow$

$\longleftarrow \beta$

$\gamma \longrightarrow$

$x = g^w$



$\beta \leftarrow_\$ \mathbb{Z}_q$

Check $g^\gamma = x^\beta \cdot \alpha$

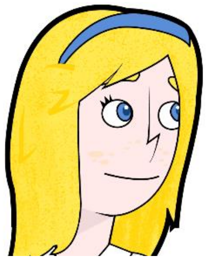
- **Special HVZK:** Upon input $pk = x$, **simulator** $\mathcal{S}$ outputs (α, β, γ) such that $\alpha = g^\gamma / x^\beta$ and $\beta, \gamma \leftarrow_\$ \mathbb{Z}_q$
- **Special soundness:** Assume we are given two accepting transcripts (α, β, γ) and $(\alpha, \beta', \gamma')$ for $pk = x$, with $\beta \neq \beta'$
  - This implies $g^{\gamma - \gamma'} = x^{\beta - \beta'}$
  - Thus, $w = (\gamma - \gamma') \cdot (\beta - \beta')^{-1}$ is the **discrete logarithm** of x

Let's Try the Same Idea using Lattices

$\text{params} = q$

$\boldsymbol{u} \leftarrow_\$ \mathbb{Z}_q^m$

$\boldsymbol{\gamma} = \beta \cdot \boldsymbol{s} + \boldsymbol{u}$

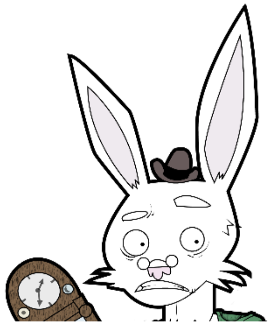


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

$\xrightarrow{\boldsymbol{\alpha} = \boldsymbol{A} \cdot \boldsymbol{u}}$

$\xleftarrow{\beta}$

$\xrightarrow{\boldsymbol{\gamma}}$



$\boldsymbol{A} \cdot \boldsymbol{s} = \boldsymbol{t}$

$\beta \leftarrow_\$ \mathbb{Z}_q$

Check $\boldsymbol{A} \cdot \boldsymbol{\gamma} = \beta \cdot \boldsymbol{t} + \boldsymbol{\alpha}$

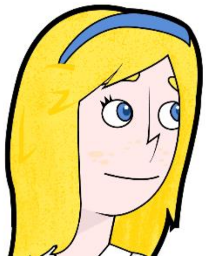
- **HVZK:** Upon input $pk = (\boldsymbol{A}, \boldsymbol{t})$, **simulator** $\mathcal{S}$ outputs $(\boldsymbol{\alpha}, \beta, \boldsymbol{\gamma})$ such that $\boldsymbol{\alpha} = \boldsymbol{A} \cdot \boldsymbol{\gamma} - \beta \cdot \boldsymbol{t}$ and $\beta \leftarrow_\$ \mathbb{Z}_q, \boldsymbol{\gamma} \leftarrow_\$ \mathbb{Z}_q^m$
- **Special soundness:** Assume we are given two accepting transcripts$(\boldsymbol{\alpha}, \beta, \boldsymbol{\gamma})$ and$(\boldsymbol{\alpha}, \beta', \boldsymbol{\gamma}')$ for $pk = (\boldsymbol{A}, \boldsymbol{t})$, with $\beta \neq \beta'$
  - This implies $\boldsymbol{A} \cdot (\boldsymbol{\gamma} - \boldsymbol{\gamma}') = (\beta - \beta') \cdot \boldsymbol{t}$
  - Thus, $\boldsymbol{s} = (\boldsymbol{\gamma} - \boldsymbol{\gamma}') \cdot (\beta - \beta')^{-1}$ is the **solution** for $\boldsymbol{A} \cdot \boldsymbol{s} = \boldsymbol{t}$

Many Problems…

- The challenge space is **small**
  - $q \approx 2^{12}$ for **encryption**
  - $q \approx 2^{30}$ for **signatures**
  - $q \approx 2^{32}$ for **advanced applications**
- This means that a **successful prover** can just **guess** β
- The vector $\boldsymbol{s}$ we extract is **not guaranteed to be small**
  - Recall that **removing** the requirement of $\boldsymbol{s}$ being **small** makes lattice problems **trivial**
- **Solution:** Choose **small** $\boldsymbol{u}, \beta$ and **repeat** the protocol in **parallel**

Modified Protocol (Take 1)

$\boldsymbol{u}_1, \dots, \boldsymbol{u}_k \leftarrow_\$ \{0,1\}^m$
$\boldsymbol{\alpha}_j = \boldsymbol{A} \cdot \boldsymbol{u}_j$
$\boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{s} + \boldsymbol{u}_j$

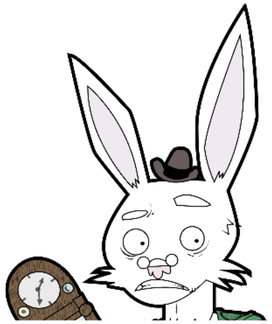


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

$(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_k) \rightarrow$

$\leftarrow (\beta_1, \dots, \beta_k)$

$(\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_k) \rightarrow$



$\boldsymbol{A} \cdot \boldsymbol{s} = \boldsymbol{t}$

$\beta_j \leftarrow_\$ \{0,1\}$

Check $\boldsymbol{A} \cdot \boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$

- The verifier checks the above $\forall j = 1, \dots, k$ and that the coefficients of each $\boldsymbol{\gamma}_j$ are **small** (i.e., in $\{0,1,2\}$)
- **Special soundness:** Given $\boldsymbol{A} \cdot \boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$ and $\boldsymbol{A} \cdot \boldsymbol{\gamma}'_j = \beta'_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$ with $\beta_j \neq \beta'_j$, extract $\boldsymbol{s} = \left(\boldsymbol{\gamma}_j - \boldsymbol{\gamma}'_j\right) \cdot (\beta_j - \beta'_j)^{-1}$
  - The elements of $\boldsymbol{\gamma}_j - \boldsymbol{\gamma}'_j$ are in $\{-2,-1,0,1,2\}$, and $\beta_j - \beta'_j$ is in $\{-1,1\}$, so $\boldsymbol{s}$ also lies in $\{-2,-1,0,1,2\}$

Insecurity of the Protocol

- There are some **caveats**:
  - We **extracted** a **slightly bigger** secret
  - We need to **repeat** for $k = 128$ or $k = 256$ times
- Even worse, the protocol **does not** satisfy **HVZK**
  - Suppose that the challenge is $\beta = 1$

0	?	1	?	1	0	?	0	?	?	$\beta \cdot \boldsymbol{s} = \boldsymbol{s}$ has coefficients in $\{0,1\}$
				+						
0	?	1	?	1	0	?	0	?	?	$\boldsymbol{u}$ has coefficients in $\{0,1\}$
				=						
0	1	2	1	2	0	1	0	1	1	γ coefficients

Possible Fix?

- Maybe we can sample $\boldsymbol{u}$ from a **larger domain**?
  - Suppose that the challenge is $\beta = 1$

0	?	?	?	1	?	0	?	?	?

$\beta \cdot \boldsymbol{s} = \boldsymbol{s}$ has coefficients in $\{0,1\}$

$+$

0	?	?	?	5	?	0	?	?	?

$\boldsymbol{u}$ has coefficients in $\{0,1,2,3,4,5\}$

$=$

0	4	2	3	6	5	0	2	4	1

$\boldsymbol{\gamma}$ coefficients

  - Whenever a $\boldsymbol{\gamma}$ coefficient is 0 or 6 we know that $\boldsymbol{s}$ is 0 or 1, but the other coefficients are **hidden** (i.e., they could be **equally** 0 or 1)
  - So, $\boldsymbol{s}$ **only** effects the probability that a $\boldsymbol{\gamma}$ coefficient is 0 or 6

Possible Fix?

- Maybe we can sample $\boldsymbol{u}$ from a **larger domain**?
  - Suppose that the challenge is $\beta = 1$

0	?	?	?	1	?	0	?	?	?

$\beta \cdot \boldsymbol{s} = \boldsymbol{s}$ has coefficients in $\{0,1\}$

$+$

0	?	?	?	5	?	0	?	?	?

$\boldsymbol{u}$ has coefficients in $\{0,1,2,3,4,5\}$

$=$

0	4	2	3	6	5	0	2	4	1

$\boldsymbol{\gamma}$ coefficients

  - In other words, the coefficients $1,2,3,4,5$ are **equally likely** to appear **regardless** of the **secret key**
  - Natural idea: Send $\boldsymbol{\gamma}$ only when **all the coefficients** are **in this range**

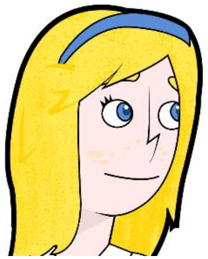
In General…

- Suppose $\boldsymbol{s}$ has coefficients in $\{0,1,\ldots,a\}$ and that $\boldsymbol{u}$ has coefficients in $\{0,1,\ldots,b-1\}$
  - Here, $b > a$
- Then, for all $a \leq i < b$, we have $\mathbb{P}[s + u = i] = 1/b$
  - Moreover, there are $b - a$ such i's and thus $1 - a/b$ **probability** of keeping the value s **secret**
- The probability that a $\boldsymbol{\gamma}$ coefficient is in $\{1,\ldots,b-1\}$ is $1 - 1/b$
  - The probability that they **all are** is $(1 - 1/b)^m$
  - The probability that they **all are for all** $\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_k$ is $(1 - 1/b)^{mk}$
  - By setting $b = mk$, we get $(1 - 1/b)^{mk} \approx 1/e$

Modified Protocol (Take 2)

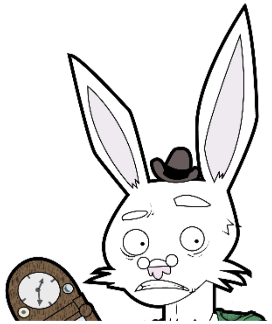
$$\boldsymbol{A}\cdot\boldsymbol{s}=\boldsymbol{t}$$

$$\boldsymbol{u}_1,\ldots,\boldsymbol{u}_k \leftarrow_\$ \{0,\ldots,mk\}^m$$
$$\boldsymbol{\alpha}_j=\boldsymbol{A}\cdot\boldsymbol{u}_j$$
$$\boldsymbol{\gamma}_j=\beta_j\cdot\boldsymbol{s}+\boldsymbol{u}_j$$



$(\boldsymbol{A},\boldsymbol{t}),\boldsymbol{s}$

$$(\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_k) \rightarrow$$
$$\leftarrow (\beta_1,\ldots,\beta_k)$$
$$(\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_k) \rightarrow$$



$$\beta_j \leftarrow_\$ \{0,1\}$$

Check $\boldsymbol{A}\cdot\boldsymbol{\gamma}_j=\beta_j\cdot\boldsymbol{t}+\boldsymbol{\alpha}_j$

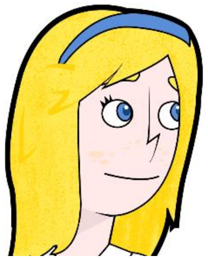
- The prover checks whether **any** of the coefficients contained in $\boldsymbol{\gamma}_j$ is 0 or $mk+1$
  - If it is, **abort** and **restart** the protocol
- The verifier checks the above $\forall j=1,\ldots,k$ and that the coefficients of each $\boldsymbol{\gamma}_j$ are **small** (i.e., in $\{0,\ldots,mk\}$)

Modified Protocol (Take 2)

$\boldsymbol{u}_1, \dots, \boldsymbol{u}_k \leftarrow_\$ \{0, \dots, mk\}^m$

$\boldsymbol{\alpha}_j = \boldsymbol{A} \cdot \boldsymbol{u}_j$

$\boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{s} + \boldsymbol{u}_j$

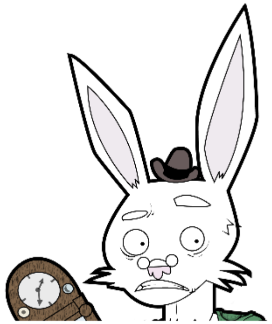


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

$(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_k) \rightarrow$

$\leftarrow (\beta_1, \dots, \beta_k)$

$(\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_k) \rightarrow$



$\boxed{\boldsymbol{A} \cdot \boldsymbol{s} = \boldsymbol{t}}$

$\beta_j \leftarrow_\$ \{0,1\}$

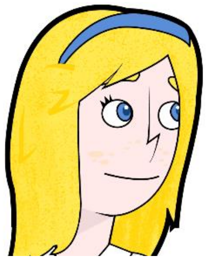
Check $\boldsymbol{A} \cdot \boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$

- **Special soundness:** Given $\boldsymbol{A} \cdot \boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$ and $\boldsymbol{A} \cdot \boldsymbol{\gamma}'_j = \beta'_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$ with $\beta_j \neq \beta'_j$, extract $\boldsymbol{s} = \left(\boldsymbol{\gamma}_j - \boldsymbol{\gamma}'_j\right) \cdot (\beta_j - \beta'_j)^{-1}$
  - The elements of $\boldsymbol{\gamma}_j - \boldsymbol{\gamma}'_j$ are in $\{-mk, \dots mk\}$, and $\beta_j - \beta'_j$ is in $\{-1,1\}$, so $\boldsymbol{s}$ also lies in $\{-mk, \dots, mk\}$
- **HVZK:** Yes, as now $\boldsymbol{\gamma}_j$ **never depends** on $\boldsymbol{s}$
  - **Caveat:** What is $\boldsymbol{\alpha}_j$ in case of **abort**?

Modified Protocol (Take 3)

$\boxed{\boldsymbol{A} \cdot \boldsymbol{s} = \boldsymbol{t}}$

$\boldsymbol{u}_1, \dots, \boldsymbol{u}_k \leftarrow_\$ \{0, \dots, mk\}^m$
$\boldsymbol{\alpha}_j = \boldsymbol{A} \cdot \boldsymbol{u}_j$
$\boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{s} + \boldsymbol{u}_j$

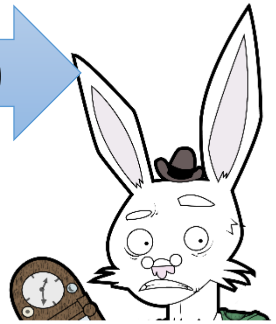


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

$\alpha = \mathbf{H}(\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_k) \rightarrow$

$\leftarrow (\beta_1, \dots, \beta_k)$

$(\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_k) \rightarrow$



$\beta_j \leftarrow_\$ \{0,1\}$

Check $\boldsymbol{A} \cdot \boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$

- The verifier checks the above $\forall j = 1, \dots, k$ and that the coefficients of each $\boldsymbol{\gamma}_j$ are **small** (i.e., in $\{0, \dots, mk\}$)
- But now it also **additionally checks** that

$$\boxed{\alpha = \mathbf{H}(\boldsymbol{A} \cdot \boldsymbol{\gamma}_1 - \beta_1 \cdot \boldsymbol{t}, \dots, \boldsymbol{A} \cdot \boldsymbol{\gamma}_k - \beta_k \cdot \boldsymbol{t})}$$

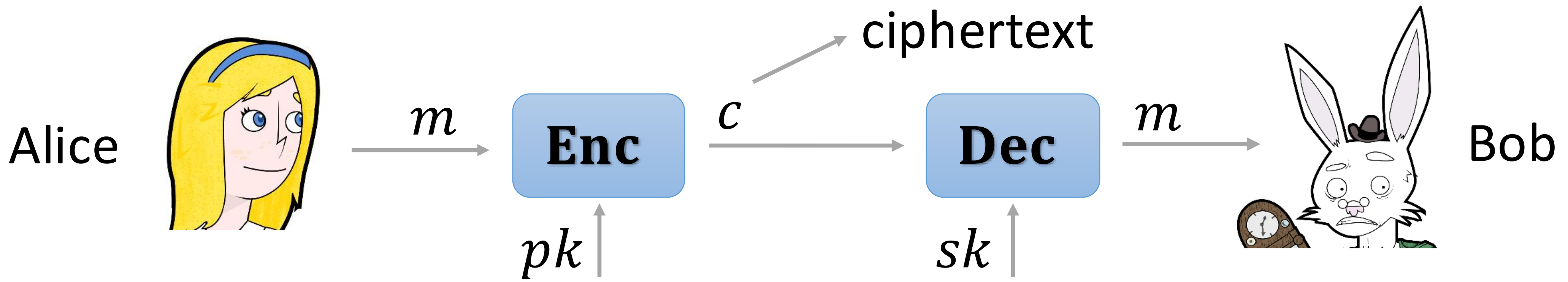
  - In case of **abort**, the HVZK simulator can still send a **random** α

In Practice

- The previous protocol still needs to be **repeated in parallel** $k = 128$ or 256 times
  - And this is the best one can get for **arbitrary** lattices
- However:
  - The proof size for **one equation** is roughly the same as the proof size for **many equations** (amortization with **logarithmic** growth)
  - Working with **polynomial rings** instead of $\mathbb{Z}_q$ allows for **one-shot approximate** proofs (i.e., the coefficients of $\boldsymbol{s}$ are **small**)
  - Using more **complex techniques**, one obtains **almost one-shot exact** proofs (i.e., the coefficients of $\boldsymbol{s}$ are in $\{0,1\}$)

Crystals-Kyber

Public-Key Encryption



- **Proposed** by Diffie and Hellman in their seminal paper [DH76]
- First **realization** by Rivest, Shamir and Adelman based on the hardness of **factoring** [RSA78]

Chosen-Plaintext Attack (CPA) Security

Eve

pk ←

x_0, x_1 →

c ←

guess b →

Challenger

$pk, sk,$ random b

$c \leftarrow \mathbf{Enc}(pk, x_b)$

- The attacker cannot even guess a **single bit** of the plaintext
  - Remember that the messages are chosen by the adversary
  - CPA security implies hardness of **recovering the message**
  - CPA security implies hardness of **recovering the secret key**

Regev PKE [Reg05]

- **Key Generation:** $pk = (\boldsymbol{A}, \boldsymbol{b})$ and $sk = \boldsymbol{s}$, where $\boldsymbol{b}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}$ and $\boldsymbol{s} \in \mathbb{Z}_q^n, \boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$
- **Encryption:** The encryption of x w.r.t. pk is made of two parts
  - Ciphertext preamble $\boldsymbol{c}_0 = \boldsymbol{A} \cdot \boldsymbol{r}$ for random $\boldsymbol{r} \in \{0,1\}^m$
  - Ciphertext payload $c_1 = \boldsymbol{b}^{\mathrm{t}} \cdot \boldsymbol{r} + x \cdot q/2$
  - Bob outputs $c_1 - \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{c}_0 \approx x \cdot q/2$
- **Security:** By LWE we can switch $(\boldsymbol{A}, \boldsymbol{b})$ with $(\boldsymbol{A}, \boldsymbol{b})$ for uniformly random $\boldsymbol{b}^{\mathrm{t}}$
  - By the **leftover hash lemma**, we can finally replace $\boldsymbol{c}_0$ with uniformly random $\boldsymbol{c}_0$, so that c_1 hides x **information theoretically**

Dual Regev [GPV08]

- **Key Generation:** $pk = (\boldsymbol{A}, \boldsymbol{u})$ and $sk = \boldsymbol{r}$, where $\boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{r}$ and $\boldsymbol{r} \in \{0,1\}^m$, $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$
- **Encryption:** The encryption of x w.r.t. pk is made of two parts
  - Ciphertext preamble $\boldsymbol{c}_0 = \boldsymbol{b}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}$ for random $\boldsymbol{s} \in \mathbb{Z}_q^n$
  - Ciphertext payload $c_1 = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2$
  - Bob outputs $c_1 - \boldsymbol{c}_0 \cdot \boldsymbol{r} \approx x \cdot q/2$
- **Security:** By the leftover hash lemma, we can switch $\boldsymbol{u}$ with **uniformly random** $\boldsymbol{u}$
  - By LWE we can switch $(\boldsymbol{c}_0, c_1)$ with **uniformly random** $(\boldsymbol{c}_0, c_1)$

Primal versus Dual

- Public key
  - Primal: pk is **pseudorandom** with **unique** sk
  - Dual: pk is **statistically random** with **many possible** sk
- Ciphertext
  - Primal: A fresh LWE sample with **many possible** coins
  - Dual: Multiple LWE samples with **unique** coins
- Security
  - Primal: Encrypting with **uniform** pk induces **random** ciphertext
  - Dual: By LWE can switch the ciphertext to **random**
- Efficiency: The matrix A can be **shared** by different users

Most Efficient [LP11]

- **Key Generation:** $pk = (\boldsymbol{A}, \boldsymbol{u})$ and $sk = \boldsymbol{s}$, where $\boldsymbol{u}^{\mathrm{t}} = \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}$ and $\boldsymbol{s} \in \chi^n, \boldsymbol{A} \in \mathbb{Z}_q^{n \times n}$
- **Encryption:** The encryption of x w.r.t. pk is made of two parts
  - Ciphertext preamble $\boldsymbol{c}_0 = \boldsymbol{A} \cdot \boldsymbol{r} + \boldsymbol{e}'$ for $\boldsymbol{r} \in \chi^n$
  - Ciphertext payload $c_1 = \boldsymbol{u}^{\mathrm{t}} \cdot \boldsymbol{r} + e' + x \cdot q/2$
  - Bob outputs $c_1 - \boldsymbol{s}^{\mathrm{t}} \cdot \boldsymbol{c}_0 \approx x \cdot q/2$
- **Security:** By LWE we can switch $(\boldsymbol{A}, \boldsymbol{u})$ with $(\boldsymbol{A}, \boldsymbol{u})$ for **uniformly random $\boldsymbol{u}$**
  - This requires LWE with secrets from the **error distribution**
  - Next, we can replace $(\boldsymbol{c}_0, c_1)$ with **uniformly random** $(\boldsymbol{c}_0, c_1)$

Chosen-Ciphertext Attack (CCA) Security

Eve

$\xleftarrow{\quad pk \quad}$

$\xrightarrow{\quad c' \quad}$

$\xleftarrow{\quad m' \quad}$

$\xrightarrow{\quad x_0, x_1 \quad}$

$\xleftarrow{\quad c \quad}$

$\xrightarrow{\quad c' \neq c \quad}$

$\xleftarrow{\quad m' \quad}$

$\xrightarrow{\quad \text{guess } b \quad}$

Challenger

$pk, sk,$ random b

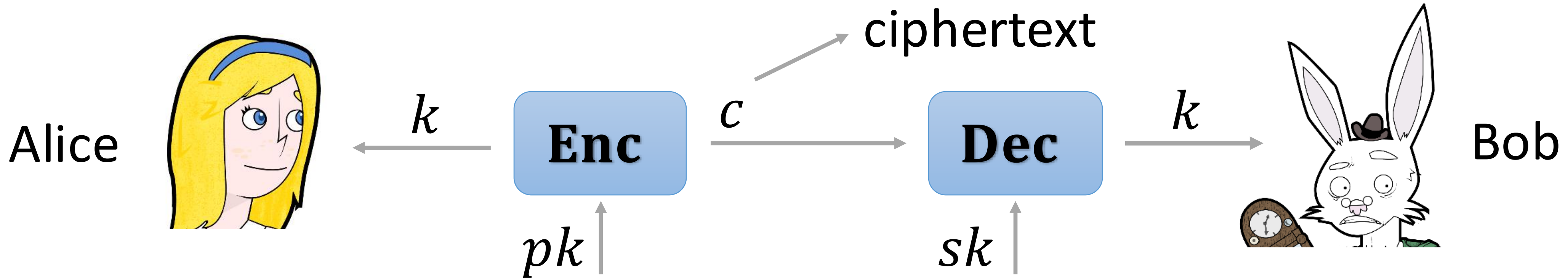
$m' = \mathbf{Dec}(sk, c')$

$c \leftarrow \mathbf{Enc}(pk, x_b)$

- The above notion captures a strong **non-malleability** guarantee
  - No attacker can **maul** a ciphertext c for message m into a ciphertext $\tilde{c}$ for message $\tilde{m}$ **related** to m
  - The **gold standard** for security of PKE in **practice**

Fujisaki-Okamoto Transform

- The **FO transform** [FO99,FO13] turns **passively** (**IND-CPA**) secure PKE schemes into **actively** (**IND-CCA**) secure ones
  - The transformation requires two **hash functions** (random oracles)
  - The obtained scheme is better understood as a **key encapsulation mechanism** (KEM)



  - We can combine a **KEM** with an **SKE** scheme to get a **PKE** scheme

One-Wayness of PKE

Eve

$\xleftarrow{\quad pk, c^* \quad}$

$\xrightarrow{\quad m, c \quad}$

$\xleftarrow{\quad \text{yes/no} \quad}$

$\xrightarrow{\quad m' \quad}$

Challenger

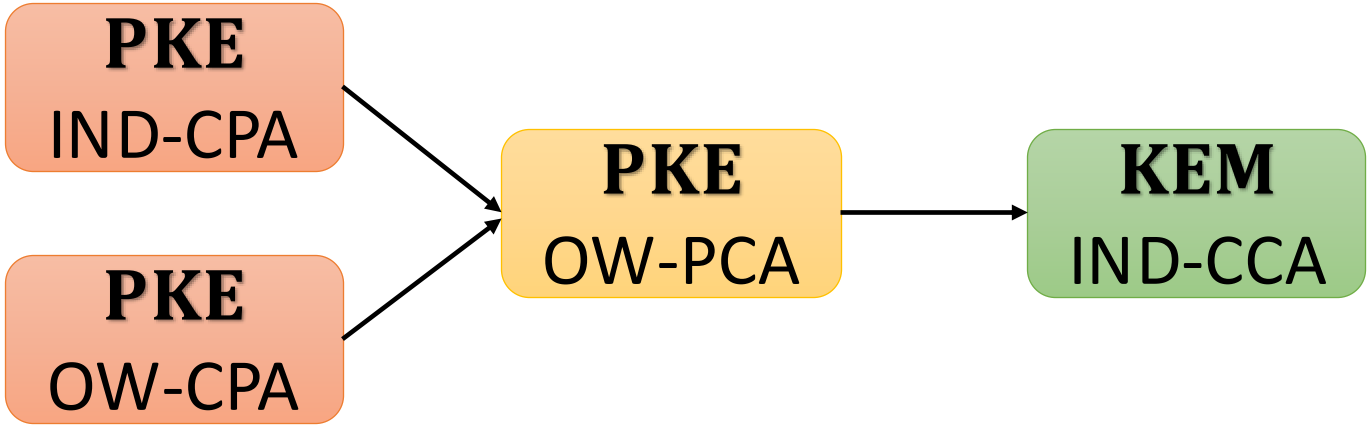
pk, sk

$m^* \leftarrow \mathcal{M}$

$c^* \leftarrow \mathbf{Enc}(pk, m^*)$

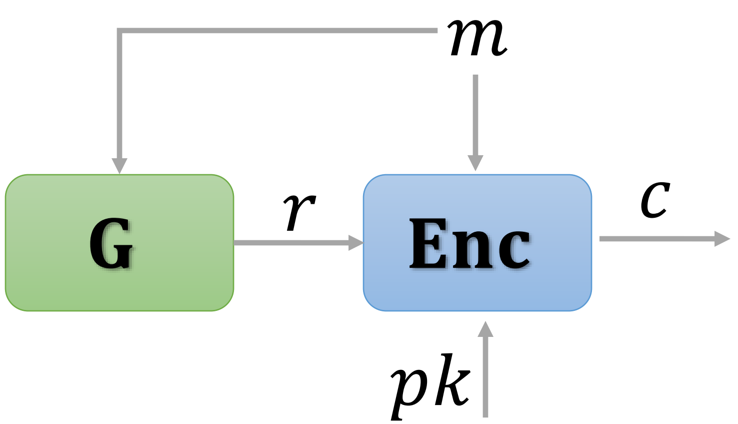
- **OW-CPA:** PKE makes it **hard to guess** the message
  - The message is **uniformly random** and **unknown** to the attacker
- **OW-PCA:** As before but now the attacker can query a **plaintext-checking oracle** which allows to check if $\mathbf{Dec}(sk, c) = m$

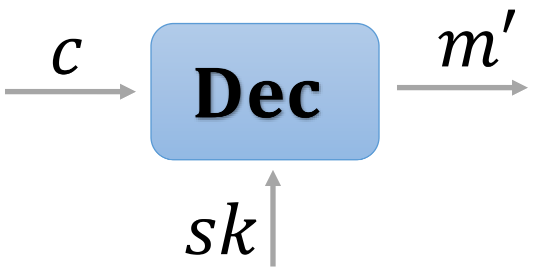
Modularization of the FO Transform



- We can view FO as the **concatenation** of **two transforms** $\mathbf{U} \circ \mathbf{T}$
  - The first transformation takes care of **derandomization** and allows to go from **IND-CPA** to **OW-PCA**
  - The second transformation takes care of **hashing** and allows to go from **OW-PCA** to **IND-CCA**

Transformation $\mathbf{T}$: From IND-CPA to OW-PCA

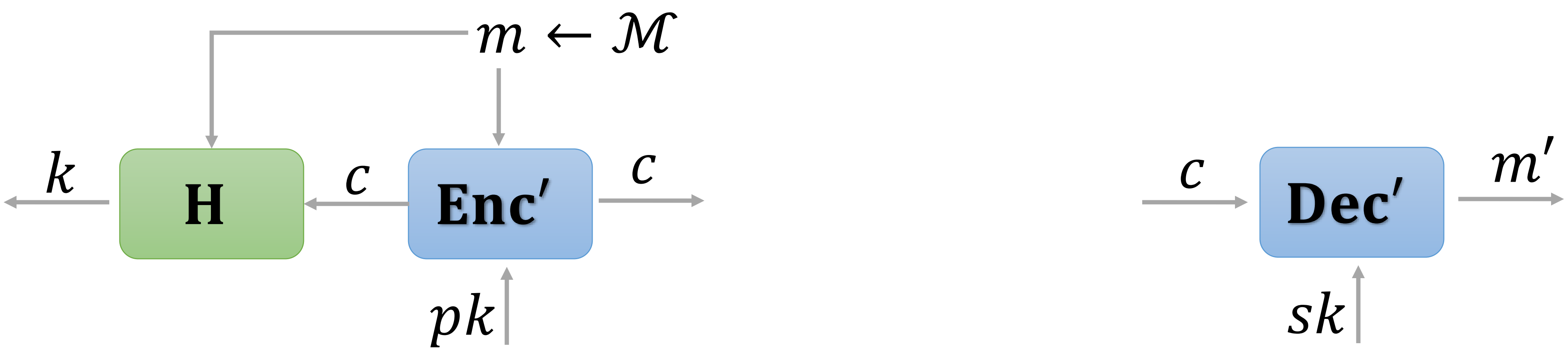




- Encryption becomes **deterministic** (the **randomness** is $\mathbf{G}(m)$)
- Decryption **re-encrypts** m' using randomness $\mathbf{G}(m')$ and outputs m' if and only if it obtains c
- **Theorem [HKK17]:** Assuming $(\mathbf{Enc}, \mathbf{Dec})$ is **IND-CPA** (**OW-CPA**), $(\mathbf{Enc}', \mathbf{Dec}')$ is **OW-PCA**

SAPIENZA UNIVERSITÀ DI ROMA

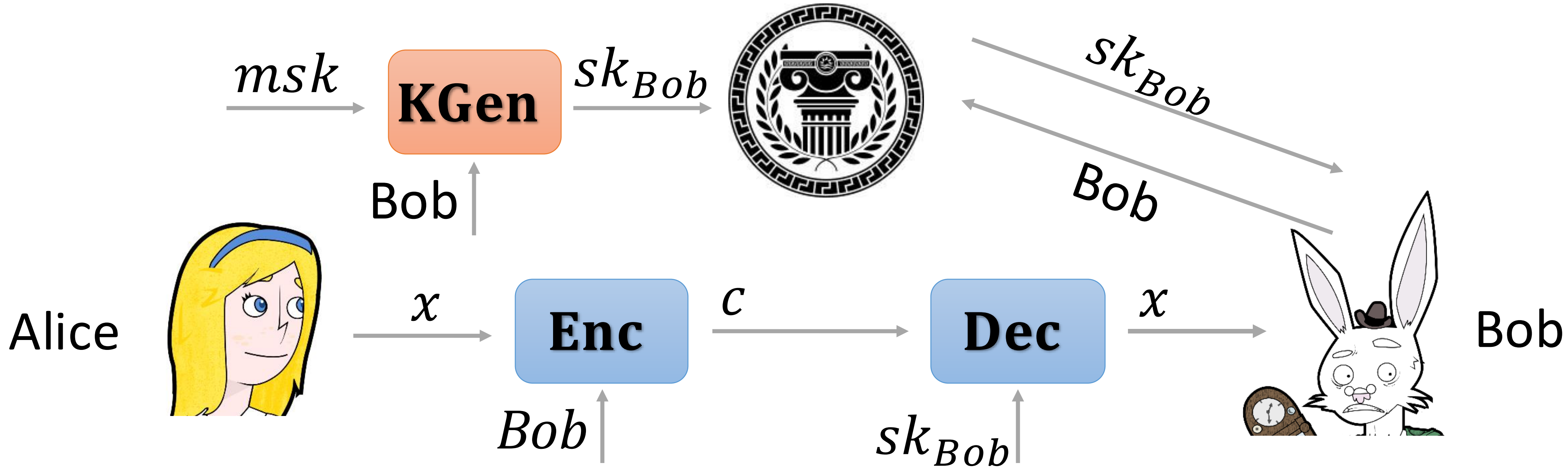
Transformation $\mathbf{U}$: From OW-PCA to IND-CCA



- Encapsulation outputs $k = \mathbf{H}(c, m)$ and c
- Decapsulation obtains $m' = \mathbf{Dec}(sk, c)$ and outputs m'
  - Here, m' could be $\perp$ (**explicit rejection**)
- **Theorem [HKK17]:** Assuming $(\mathbf{Enc}', \mathbf{Dec}')$ is **OW-PCA**, $(\mathbf{Encaps}, \mathbf{Decaps})$ is **IND-CCA**

Advanced Cryptographic Applications

Identity-Based Encryption



- **Postulated** by Shamir in 1984 [Sha84]
  - Avoids the need of **certificates**
  - Introduces the so-called **key escrow** problem
- First **realization** by Boneh and Franklin in 2001 [BF01]

Selective Security of IBE

Eve

Challenger

$mpk, msk,$ random b

$c \leftarrow \mathbf{Enc}(ID^*, x_b)$

$\xrightarrow{ID^*}$

$\xleftarrow{mpk}$

$\xrightarrow{ID}$

$\xleftarrow{sk_{ID} = \mathbf{KGen}(msk, ID)}$

$\xrightarrow{x_0, x_1}$

$\xleftarrow{c}$

$\xrightarrow{\text{guess } b}$

- Every **selectively** secure IBE is also **fully** secure with an **exponential** loss in the parameters
  - Also, general **transformations** are known

Warm-up Construction [CHKP10]

- **Public parameters:** $mpk = (\boldsymbol{A_0}, \boldsymbol{A}_1^0, \boldsymbol{A}_1^1, \boldsymbol{A}_2^0, \boldsymbol{A}_2^1, \boldsymbol{u})$
  - Assume, for simplicity, $|ID| = 2$
- **Master secret key:** Trapdoor for $\boldsymbol{A}_0$
  - Secret key for identity $ID = 01$: **Short vector $\boldsymbol{s}$** s.t. $\boldsymbol{F}_{01} \cdot \boldsymbol{s} = \boldsymbol{u} \bmod q$, where $\boldsymbol{F}_{01} = [\boldsymbol{A}_0 | \boldsymbol{A}_1^0 | \boldsymbol{A}_2^1]$
  - Note: A trapdoor for $\boldsymbol{A}_0$ **implies** a trapdoor for $\boldsymbol{F}_{01}$
- **Encryption: Dual** Regev encryption of x w.r.t. matrix $\boldsymbol{F}_{01}$
  - The ciphertext is $\boldsymbol{c}_0^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{F}_{01} + \boldsymbol{e}^{\mathrm{t}}$ and $c_1 = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2$
  - Bob outputs $c_1 - \boldsymbol{c}_0^{\mathrm{t}} \cdot \boldsymbol{s} \approx x \cdot q/2$

Simulation

- Assume the **challenge** identity is $ID^* = 11$
  - The reduction **can't know** the secret key for ID^*
- Choose $\boldsymbol{A}_0, \boldsymbol{A}_1^1, \boldsymbol{A}_2^1$ uniformly at **random**, but sample $\boldsymbol{A}_1^0, \boldsymbol{A}_2^0$ with the corresponding **trapdoors**
- The reduction can derive trapdoors for $\boldsymbol{F}_{00} = [\boldsymbol{A}_0|\boldsymbol{A}_1^0|\boldsymbol{A}_2^0]$, $\boldsymbol{F}_{01} = [\boldsymbol{A}_0|\boldsymbol{A}_1^0|\boldsymbol{A}_2^1]$, and $\boldsymbol{F}_{10} = [\boldsymbol{A}_0|\boldsymbol{A}_1^1|\boldsymbol{A}_2^0]$ but not for $\boldsymbol{F}_{11} = [\boldsymbol{A}_0|\boldsymbol{A}_1^1|\boldsymbol{A}_2^1]$
  - This allows the reduction to **simulate** key extraction queries while **embedding** the LWE challenge in the simulation

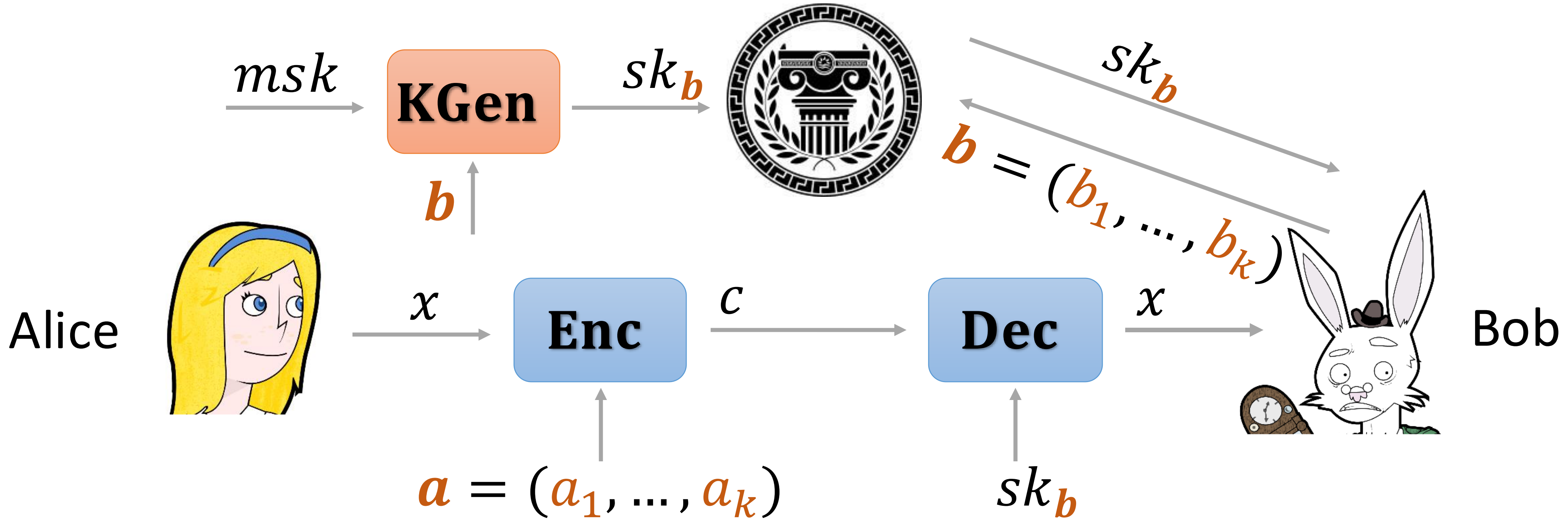
A More Efficient Construction [ABB10]

- **Public parameters:** $mpk = (\boldsymbol{A_0}, \boldsymbol{A}_1, \boldsymbol{G}, \boldsymbol{u})$
- **Master secret key:** Trapdoor for $\boldsymbol{A}_0$
  - Secret key for identity ID: **Short vector $\boldsymbol{s}$** s.t. $\boldsymbol{F}_{ID} \cdot \boldsymbol{s} = \boldsymbol{u} \bmod q$, where $\boldsymbol{F}_{ID} = [\boldsymbol{A}_0 | \boldsymbol{A}_1 + ID \cdot \boldsymbol{G}]$
  - As before, a trapdoor for $\boldsymbol{A}_0$ **implies** a trapdoor for $\boldsymbol{F}_{ID}$
- **Encryption: Dual** Regev encryption of x w.r.t. matrix $\boldsymbol{F}_{ID}$
  - The ciphertext is $\boldsymbol{c}_0^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{F}_{ID} + \boldsymbol{e}^{\mathrm{t}}$ and $c_1 = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2$
  - Bob outputs $c_1 - \boldsymbol{c}_0^{\mathrm{t}} \cdot \boldsymbol{s} = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2 - \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{F}_{ID} \cdot \boldsymbol{s} + \boldsymbol{e}^{\mathrm{t}} \cdot \boldsymbol{s} = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2 - \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + \boldsymbol{e}^{\mathrm{t}} \cdot \boldsymbol{s} \approx x \cdot q/2$

Simulation Revisited

- Assume the **challenge** identity is ID^*
  - The reduction **can't know** the secret key for ID^*
- The reduction does **not** know a trapdoor for $\boldsymbol{A}_0$, but it knows a trapdoor for the gadget matrix $\boldsymbol{G}$
- Let $\boldsymbol{A}_1 = [\boldsymbol{A}_0 \cdot \boldsymbol{R} - ID^* \cdot \boldsymbol{G}]$, where $\boldsymbol{R}$ is **random** and **low-norm**
  - This is **indistinguishable** from the real $\boldsymbol{A}_1$
- Note that $\boldsymbol{F}_{ID} = [\boldsymbol{A}_0 | \boldsymbol{A}_0 \cdot \boldsymbol{R} + (ID - ID^*) \cdot \boldsymbol{G}]$
  - Using the technique of [MP12], we can **derive** a trapdoor for $\boldsymbol{F}_{ID}$ given a trapdoor for $\boldsymbol{A}_0$
  - This allows to **simulate** key extraction queries for all $ID \neq ID^*$
  - The LWE challenge can be **embedded** as before

Inner-product Encryption [KSW08]

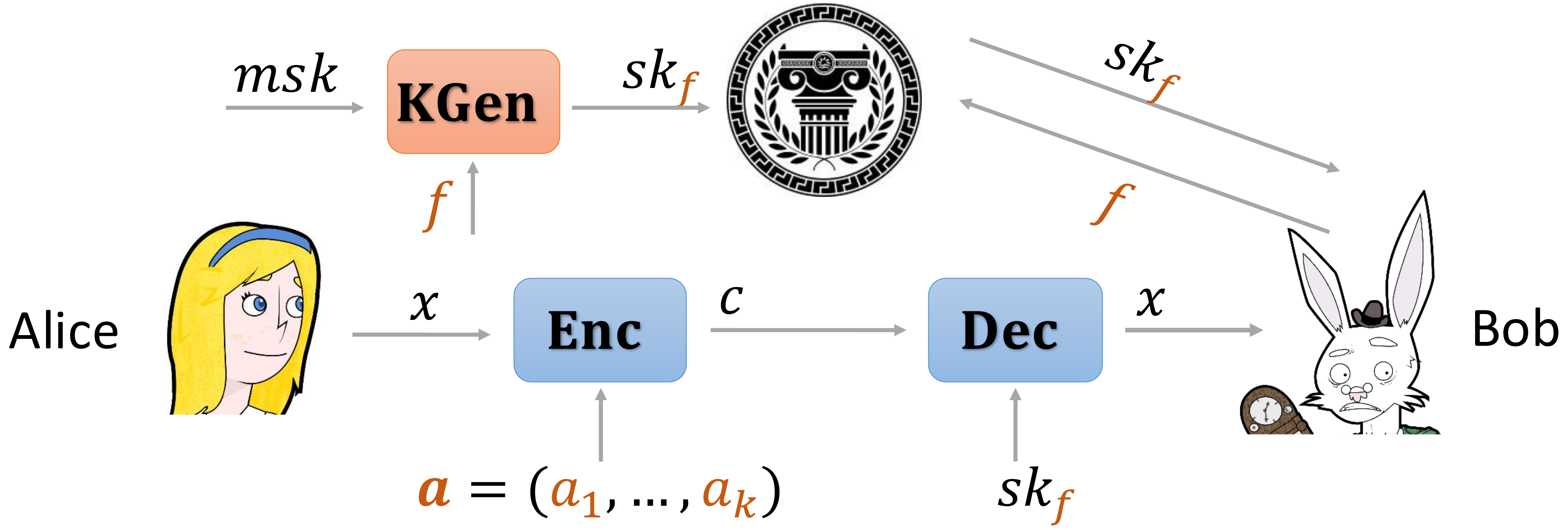


- Decryption reveals x **if and only if** $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = 0$
  - Here, we can also be interested in **attributes privacy**
- Can be used to obtain **predicate encryption** for polynomial evaluation, CNFs/DNFs of bounded degree, and **fuzzy** IBE

Generalizing to Inner Products [AFV11]

- **Public parameters:** $mpk = (\boldsymbol{A}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_k, \boldsymbol{G}, \boldsymbol{u})$
- **Master secret key:** Trapdoor for $\boldsymbol{A}$
  - Secret key for b: **Short vector** $\boldsymbol{s}_b$ s.t. $\boldsymbol{F}_b \cdot \boldsymbol{s}_b = \boldsymbol{u} \bmod q$, where $\boldsymbol{F}_b = [\boldsymbol{A} | \sum_i b_i \cdot \boldsymbol{A}_i]$
- **Encryption: Dual** Regev encryption of x w.r.t. matrix $\boldsymbol{A}$
  - The ciphertext is $\boldsymbol{c}_0^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}$, $c' = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2$, and $\boldsymbol{c}_i^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot (\boldsymbol{A}_i + a_i \cdot \boldsymbol{G}) + \boldsymbol{e}_i^{\mathrm{t}}$ (so it indeed hides $\boldsymbol{a}$)
  - Bob sets $\boldsymbol{c}_b = \sum_i b_i \cdot \boldsymbol{c}_i = \boldsymbol{r}^{\mathrm{t}} \cdot (\sum_i b_i \cdot \boldsymbol{A}_i + \sum_i a_i \cdot b_i \cdot \boldsymbol{G}) + \sum_i b_i \cdot e_i$ which equals $\boldsymbol{r}^{\mathrm{t}} \cdot \sum_i b_i \cdot \boldsymbol{A}_i + \sum_i b_i \cdot e_i$
  - Hence, $[\boldsymbol{c}_0 | \boldsymbol{c}_b] \approx \boldsymbol{r}^{\mathrm{t}} \cdot [\boldsymbol{A} | \sum_i b_i \cdot \boldsymbol{A}_i]$ is a dual Regev ciphertext
  - Bob outputs $c' - \boldsymbol{c}_0^{\mathrm{t}} \cdot \boldsymbol{s}_b - \boldsymbol{c}_b^{\mathrm{t}} \cdot \boldsymbol{s}_b \approx x \cdot q/2$

Attribute-based Encryption [SW04]



- Decryption reveals x **if and only if $f(\boldsymbol{a}) = 0$**
  - Here, we are not interested in **attributes privacy**
- Plenty of applications for **privacy-preserving data mining** and in cryptography for **big data**

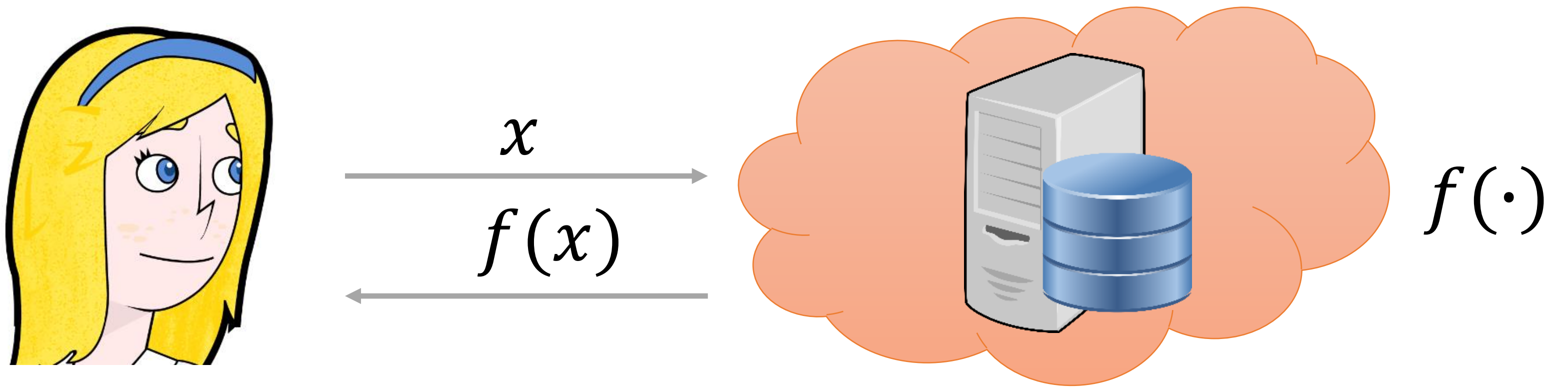
Handling Multiplications [BGG+14]

- Let $\boldsymbol{c}_1^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot (\boldsymbol{A}_1 + a_1 \cdot \boldsymbol{G}) + \boldsymbol{e}_1^{\mathrm{t}}$ and $\boldsymbol{c}_2^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot (\boldsymbol{A}_2 + a_2 \cdot \boldsymbol{G}) + \boldsymbol{e}_2^{\mathrm{t}}$
- Want: $\boldsymbol{c}_{12}^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot (\boldsymbol{A}_{12} + a_1 \cdot a_2 \cdot \boldsymbol{G}) + \boldsymbol{e}_{12}^{\mathrm{t}}$
  - Compute $(\boldsymbol{A}_1 + a_1 \cdot \boldsymbol{G}) \cdot \boldsymbol{G}^{-1}(-\boldsymbol{A}_2) = \boldsymbol{A}_1 \cdot \boldsymbol{G}^{-1}(-\boldsymbol{A}_2) - a_1 \cdot \boldsymbol{A}_2$
  - Compute $(\boldsymbol{A}_2 + a_2 \cdot \boldsymbol{G}) \cdot a_1 = a_1 \cdot \boldsymbol{A}_2 + a_1 \cdot a_2 \cdot \boldsymbol{G}$
  - The **difference** is $\boldsymbol{A}_{12} + a_1 \cdot a_2 \cdot \boldsymbol{G}$
- So, we let $\boldsymbol{c}_{12}^{\mathrm{t}} = \boldsymbol{c}_1^{\mathrm{t}} \cdot \boldsymbol{G}^{-1}(-\boldsymbol{A}_2) + \boldsymbol{c}_2^{\mathrm{t}} \cdot a_1$
  - $\boldsymbol{G}^{-1}(-\boldsymbol{A}_2)$ and a_1 are **small** and **do not effect noise**
  - As usual, additionally let $\boldsymbol{c}_0^{\mathrm{t}} = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{A} + \boldsymbol{e}^{\mathrm{t}}$, $c' = \boldsymbol{r}^{\mathrm{t}} \cdot \boldsymbol{u} + e' + x \cdot q/2$
  - If $a_1 \cdot a_2 = 0$, then $[\boldsymbol{c}_0 | \boldsymbol{c}_{12}] \approx \boldsymbol{r}^{\mathrm{t}} \cdot [\boldsymbol{A} | \boldsymbol{A}_{12}]$
  - The secret key is a **short vector** $\boldsymbol{s}_{12}$ s.t. $[\boldsymbol{A} | \boldsymbol{A}_{12}] \cdot \boldsymbol{s}_{12} = \boldsymbol{u} \bmod q$
  - Bob outputs $c' - \boldsymbol{c}_0^{\mathrm{t}} \cdot \boldsymbol{s}_{12} - \boldsymbol{c}_{12}^{\mathrm{t}} \cdot \boldsymbol{s}_{12} \approx x \cdot q/2$

Computing over Encrypted Data

- Can we have a (public-key) encryption scheme which allows to run **computations** over **encrypted data**?
- Question dating back to the late 70s
  - Ron Rivest and "privacy homomorphisms"
- Partial solutions known
  - E.g., RSA and Elgamal enjoy limited forms of homomorphism
- First solution by Craig Gentry after 30 years
  - The "Swiss Army knife of cryptography"

Motivation: Outsourcing of Computation

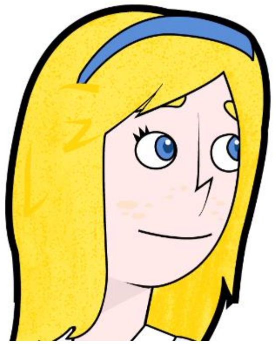


- Email, web search, navigation, social networking, …
- What about **private** x?

SAPIENZA UNIVERSITÀ DI ROMA

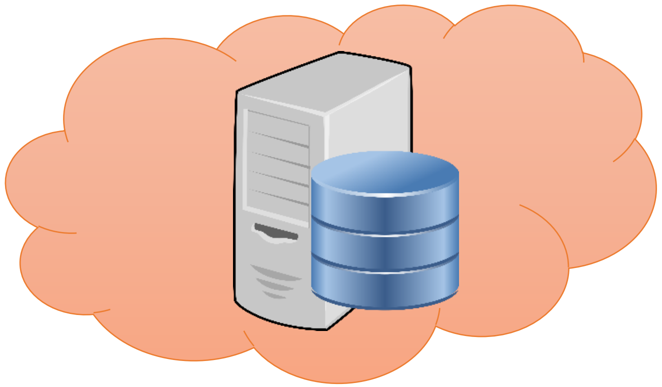
Outsourcing of Computation - Privately

$\mathbf{Dec}(sk, y) = f(x)$



$\mathbf{Enc}(pk, x)$

y

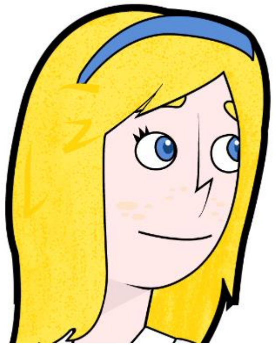


$f(\cdot)$

Wish: Homomorphic **evaluation** function:

$\mathbf{Eval}: pk, f, \mathbf{Enc}(pk, x) \rightarrow \mathbf{Enc}(pk, f(x))$

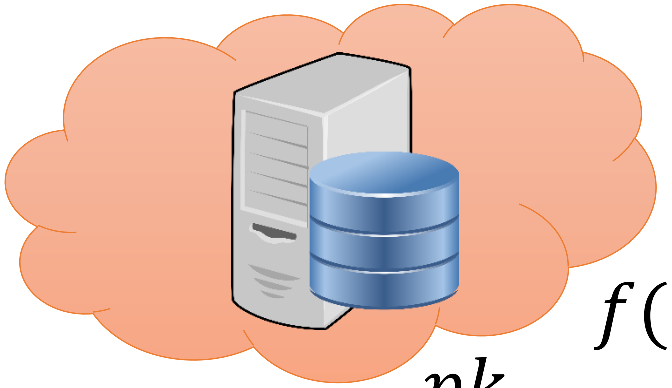
Fully-Homomorphic Encryption (FHE)



pk, sk

$c = \mathbf{Enc}(pk, x)$

$y = \mathbf{Eval}(pk, f, c)$



$f(\cdot)$

pk

Correctness:

$\mathbf{Dec}(sk, y) = f(x)$

Privacy:

$\mathbf{Enc}(pk, x) \approx \mathbf{Enc}\left(pk, 0^{|x|}\right)$

FHE = Correctness ∀ efficient f = Correctness for universal set

Levelled FHE: **Bounded** depth f

- NAND
- $(+,\times)$ over a ring

A Paradox (and its Resolution)

$$c_1 = \mathbf{Enc}(pk, x_1)$$
$$c_2 = \mathbf{Enc}(pk, x_2)$$
$$c_3 = \mathbf{Enc}(pk, x_3)$$

$$f(x_1, x_2, x_3) = \begin{cases} x_2 \text{ if } x_1 = 0 \\ x_3 \text{ if } x_1 = 1 \end{cases}$$

$$\mathbf{Enc}(pk, x_2)$$

AH! So $x_1 = 0$

$$\mathbf{Eval}(pk, f, (c_1, c_2, c_3))$$

- But remember that encryption is **randomized**!
- Output of $\mathbf{Eval}$ will look as a **fresh and random** ciphertext

Syntax of FHE

- More formally: $\Pi = (\mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec}, \mathbf{Eval})$
  - $\mathbf{KGen}(1^\lambda, 1^\tau)$: Takes the security parameter $\lambda \in \mathbb{N}$ and another parameter $\tau \in \mathbb{N}$, and outputs (pk, sk)
  - $\mathbf{Enc}(pk, x)$: Takes a plaintext bit x, and outputs a ciphertext c
  - $\mathbf{Dec}(sk, c)$: Takes a ciphertext c, and outputs a bit x
  - $\mathbf{Eval}(pk, \Gamma, \vec{c})$: Takes $\vec{c} = (c_1, \ldots, c_t)$, and outputs another vector $\vec{c}'$
- **Correctness:** Let $C = \{C_\tau\}_{\tau \in \mathbb{N}}$. Then Π is correct for C if $\forall \lambda, \tau \in \mathbb{N}, \forall (pk, sk) \in \mathbf{KGen}(1^\lambda, 1^\tau)$:

$$\forall x \in \{0,1\}:\ \mathbb{P}[\mathbf{Dec}(sk, \mathbf{Enc}(pk, x)) = x] = 1$$

$$\forall \Gamma \in C_\tau, \forall \vec{x} \in \{0,1\}^t:\ \mathbb{P}[\mathbf{Dec}(sk, \mathbf{Eval}(pk, \Gamma, \mathbf{Enc}(pk, \vec{x}))) = \Gamma(\vec{x})] = 1$$

Degrees of Homomorphism

- **Fully-Homomorphic Encryption:** Correctness holds for C such that C_1 already contains **all** Boolean circuits
  - No need to consider the additional parameter τ
- **Somewhat/Levelled Homomorphic encryption:** Correctness holds for the family C such that for all $\tau \in \mathbb{N}$ the set C_τ contains all Boolean circuits **with depth** τ
- **Additively Homomorphic Encryption:** Correctness holds for C such that C_1 contains all Boolean circuits **with only XOR gates**
  - No need to consider the additional parameter τ

Trivial FHE?

- Let $(\mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ be **any PKE** scheme
- Define the following **fully-homomorphic** PKE $(\mathbf{KGen}, \mathbf{Enc}, \mathbf{Eval}', \mathbf{Dec}')$:
  - $\mathbf{Eval}'(pk, \Gamma, c) = (\Gamma, c)$
  - $\mathbf{Dec}'(sk, c) = \Gamma(\mathbf{Dec}(sk, c))$

Wish: Complexity of decryption **much less** than running the circuit from scratch

Strong Homomorphism

- The simplest (and strongest) requirement is to ask that fresh and evaluated ciphertexts **look the same**
- We say that Π is **strongly homomorphic** for $C = \{C_\tau\}_{\tau\in\mathbb{N}}$, if for all $\tau \in \mathbb{N}$, every $\Gamma \in C_\tau$ and $\vec{x} \in \{0,1\}^t$, it holds

$$\mathbf{Fresh}_{\Pi,\vec{x}}(\lambda) = \left\{(pk,\vec{c},\vec{c}'): \begin{array}{c} (pk,sk) \leftarrow_\$ \mathbf{KGen}(1^\lambda,1^\tau) \\ \vec{c} \leftarrow_\$ \mathbf{Enc}(pk,\vec{x}), \vec{c}' \leftarrow_\$ \mathbf{Enc}(pk,\Gamma(\vec{x})) \end{array}\right\}$$

$$\approx_s \text{ or } \approx_c$$

$$\mathbf{Eval}_{\Pi,\vec{x}}(\lambda) = \left\{(pk,\vec{c},\vec{c}'): \begin{array}{c} (pk,sk) \leftarrow_\$ \mathbf{KGen}(1^\lambda,1^\tau) \\ \vec{c} \leftarrow_\$ \mathbf{Enc}(pk,\vec{x}), \vec{c}' \leftarrow_\$ \mathbf{Eval}(pk,\Gamma,\vec{c}) \end{array}\right\}$$

Strong Homomorphism

- Assume the class C contains some C_{τ^*} which includes AND and XOR (or NAND) gates
- Then we can evaluate every circuit by repeatedly evaluating each gate on the outputs of preceedings gates
  - By **strong homomorphism**, the output distribution when evaluating any Γ is at most $\mathrm{negl}(\lambda) \cdot \mathrm{size}(\Gamma)$ far from that of a fresh encryption of the output
- Hence, we have obtained a **strongly fully-homomorphic** PKE!

Compactness

- The following **weaker property** is often **sufficient**
- We say that Π is **compact** if there is a **fixed polynomial bound** $B(\cdot)$ such that for all $\lambda, \tau \in \mathbb{N}$, any circuit Γ with t-bit inputs and 1-bit output, and all $\vec{x} \in \{0,1\}^t$:

$$\mathbb{P}\left[|c'| \leq B(\lambda) : \begin{array}{c} (pk, sk) \leftarrow_\$ \mathbf{KGen}(1^\lambda, 1^\tau) \\ \vec{c} \leftarrow_\$ \mathbf{Enc}(pk, \vec{x}), c' \leftarrow_\$ \mathbf{Eval}(pk, \Gamma, \vec{c}) \end{array}\right] = 1$$

- Note that B **does not depend** on τ
  - An even weaker condition (dubbed **weak compactness**) is to have $B(\lambda, \tau)$, but still say $B(\lambda, \tau) = \mathrm{poly}(\lambda) \cdot o(\log |C_\tau|)$

Secret-Key versus Public-Key FHE

- There is also a **secret-key** variant of FHE
  - Just set $pk = \varepsilon$, and have both $\mathbf{Enc}$, $\mathbf{Dec}$ take only sk as input, whereas $\mathbf{Eval}$ takes only Γ, c
- Simple transform from SK-FHE to PK-FHE: Given $\Pi = (\mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec}, \mathbf{Eval})$ let $\Pi' = (\mathbf{KGen}', \mathbf{Enc}', \mathbf{Dec}, \mathbf{Eval})$
  - $\mathbf{KGen}'$ runs $\mathbf{KGen}$ and lets $pk = (c_0, c_1)$ where $c_0 \leftarrow_\$ \mathbf{Enc}(sk, 0)$ and $c_1 \leftarrow_\$ \mathbf{Enc}(sk, 1)$
  - $\mathbf{Enc}'(pk, x)$ outputs $\mathbf{Eval}(\Gamma_{\text{id}}, c_x)$ where Γ_{id} represents the identity
  - If Π is **strongly homomorphic**, the output of $\mathbf{Enc}'$ is **statistically close** to that of $\mathbf{Enc}(sk, x)$
  - Both strong homomorphism and semantic security are **preserved**!

The Gentry-Sahai-Waters FHE Scheme

- In what follows we will present the FHE scheme due to:
  - C. Gentry, A. Sahai, B. Waters: "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based." CRYPTO 2013
- Based on the **Learning with Errors (LWE)** assumption
- Only achieves **levelled homomorphism**
  - But can be **bootstrapped** to **full homomorphism** using a trick by Gentry (under additional assumptions)
- Plaintext space will be $\mathbb{Z}_q = [-q/2, q/2)$, for a large prime q
  - For simplicity let us write $[a]_q$ for $a \bmod q$

SAPIENZA UNIVERSITÀ DI ROMA

Eigenvectors Method (Basic Idea)

- Let C_1 and C_2 be matrices for **eigenvector** $\vec{s}$, and **eigenvalues** x_1, x_2 (i.e., $\vec{s} \times C_i = x_i \cdot \vec{s}$)
  - $C_1 + C_2$ has eigenvalue $x_1 + x_2$ w.r.t. $\vec{s}$
  - $C_1 \times C_2$ has eigenvalue $x_1 \cdot x_2$ w.r.t. $\vec{s}$
- Idea: Let C be the ciphertext, $\vec{s}$ be the secret key and x be the plaintext (say over $\mathbb{Z}_q$)
  - Homomorphism for **addition/multiplication**
  - But **insecure**: Easy to compute eigenvalues

Approximate Eigenvectors (1/2)

- Approximate variant: $\vec{s} \times C = x \cdot \vec{s} + \vec{e} \approx x \cdot \vec{s}$
  - Decryption **works** as long as $||\vec{e}||_\infty \ll q$

$$\vec{s} \times C_1 = x_1 \cdot \vec{s} + \vec{e}_1 \qquad \vec{s} \times C_2 = x_2 \cdot \vec{s} + \vec{e}_2$$
$$||\vec{e}_1||_\infty \ll q \qquad ||\vec{e}_2||_\infty \ll q$$

- Goal: Define **homomorphic** operations

$$C_{\text{add}} = C_1 + C_2:$$
$$\begin{aligned}\vec{s} \times (C_1 + C_2) &= \vec{s} \times C_1 + \vec{s} \times C_2 \\ &= x_1 \cdot \vec{s} + \vec{e}_1 + x_2 \cdot \vec{s} + \vec{e}_2 \\ &= (x_1 + x_2) \cdot \vec{s} + (\vec{e}_1 + \vec{e}_2)\end{aligned}$$

Noise **grows** a little!

Approximate Eigenvectors (2/2)

- Approximate variant: $\vec{s} \times C = x \cdot \vec{s} + \vec{e} \approx x \cdot \vec{s}$
  - Decryption **works** as long as $||\vec{e}||_\infty \ll q$

$$\vec{s} \times C_1 = x_1 \cdot \vec{s} + \vec{e}_1 \qquad \vec{s} \times C_2 = x_2 \cdot \vec{s} + \vec{e}_2$$
$$||\vec{e}_1||_\infty \ll q \qquad ||\vec{e}_2||_\infty \ll q$$

- Goal: Define **homomorphic** operations

$C_{\text{mult}} = C_1 \times C_2$:

$$\begin{aligned}\vec{s} \times (C_1 \times C_2) &= (x_1 \cdot \vec{s} + \vec{e}_1) \times C_2 \\ &= x_1 \cdot (x_2 \cdot \vec{s} + \vec{e}_2) + \vec{e}_1 \times C_2 \\ &= x_1 \cdot x_2 \cdot \vec{s} + (x_1 \cdot \vec{e}_2 + \vec{e}_1 \times C_2)\end{aligned}$$

Noise **grows**! Needs to be **small**!

Shrinking Gadgets

- Write entries in C using **binary decomposition**; e.g.

$$C = \begin{bmatrix} 3 & 5 \\ 1 & 4 \end{bmatrix} \pmod 8 \xrightarrow{\text{yields}} \text{bits}(C) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \pmod 8$$

- **Reverse** operation:

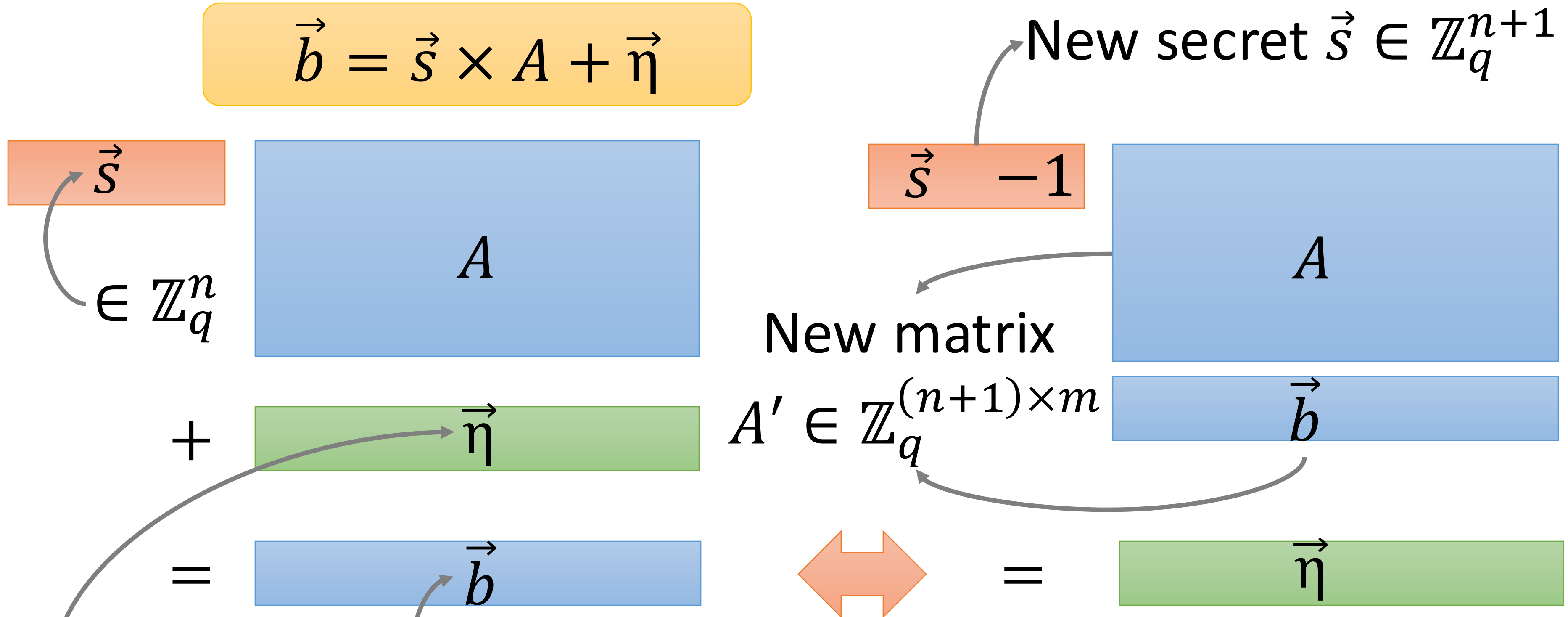
$$C = G \times G^{-1}(C) = \begin{bmatrix} \underbrace{2^{N-1} & \dots}_{k \cdot N = k\lceil \log q \rceil} & 2 & 1 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 & 2^{N-1} & \dots & 2 & 1 \end{bmatrix} \times \text{bits}(C)$$

$$\Rightarrow \vec{s} \times C = \vec{s} \times G \times G^{-1}(C)$$

SAPIENZA UNIVERSITÀ DI ROMA

LWE – Rearranging Notation

$$\vec{b} = \vec{s} \times A + \vec{\eta}$$



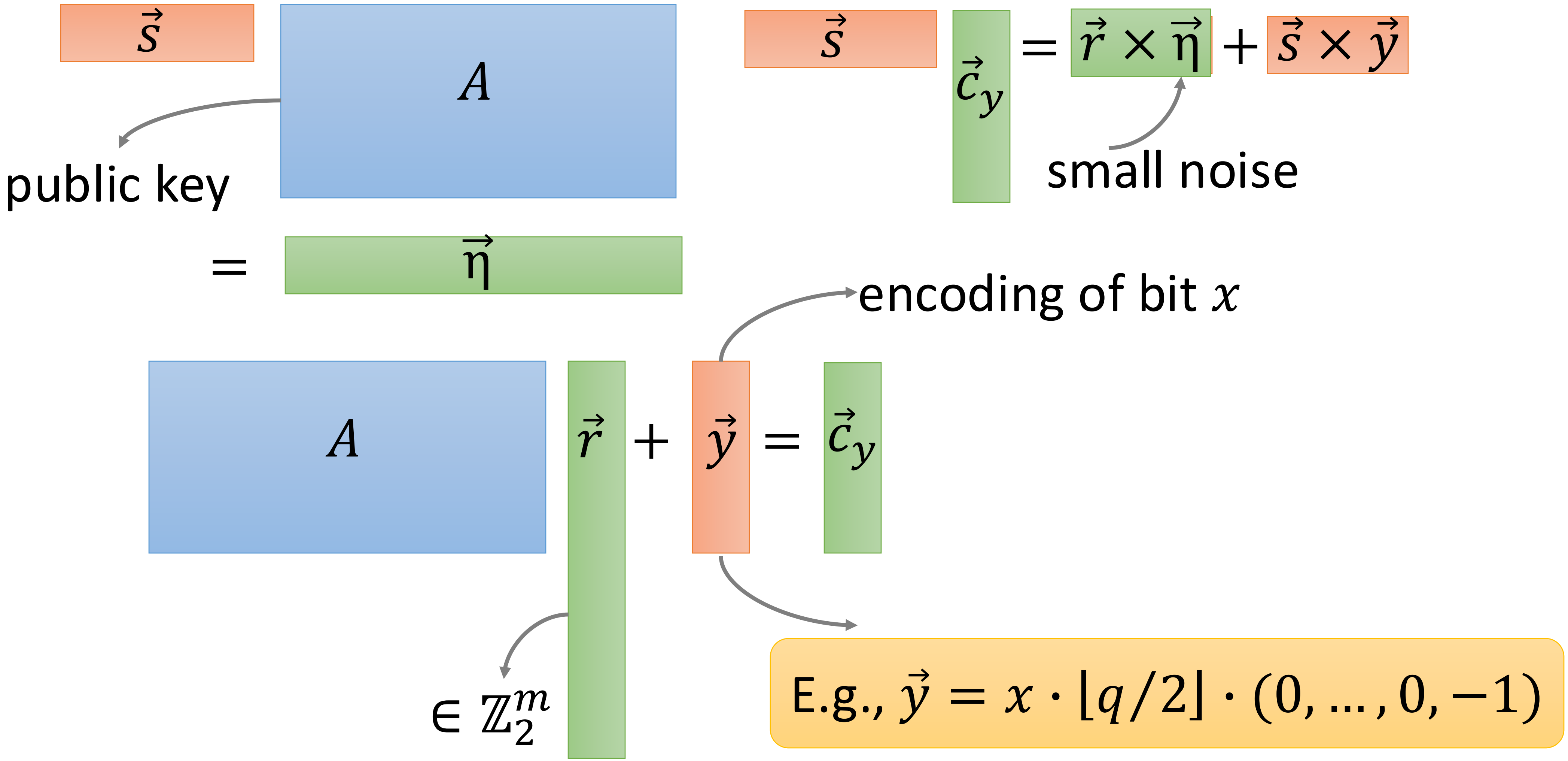
Small noise $\in \mathbb{Z}_q^m$

$|\eta_i| \leq \alpha q; \alpha \ll 1$

$\in \mathbb{Z}_q^m$

LWE: $A' = (A||\vec{b}) \approx_c \mathbf{U}_q^{(n+1)\times m}$

Regev PKE – Pictorially



The GSW Scheme

$\vec{s}$

A

public key $\in \mathbb{Z}_q^{n\times m}$ $=$ $\vec{\eta}$

$\in \mathbb{Z}_2^{m\times N} = \mathbb{Z}_2^{m\times n\cdot\lceil \log q\rceil}$

$$\mathbf{Enc}(A, x; R) = [A \times R + x\cdot G]_q = C_{x\cdot G}$$

$$\|\vec{e}\|_\infty = \|\vec{\eta} \times R\|_\infty \leq (\alpha q)\cdot m = n \cdot m$$

Invariant: $\vec{s} \times C = \vec{e} + x\cdot \vec{s} \times G$

$$\mathbf{Dec}(\vec{s}, C) = \vec{s} \times C \times G^{-1}((0, \dots, 0, -\lfloor q/2 \rceil))$$
$$= \vec{e} \times G^{-1}(\cdots) + x\cdot \vec{s} \times G \times G^{-1}((0, \dots, 0, -\lfloor q/2 \rceil))$$
$$= \vec{e} \times G^{-1}(\cdots) + \lfloor q/2 \rceil \cdot x = z$$

Output: $0 \Leftrightarrow |z| < q/4$

The GSW Scheme – Homomorphism

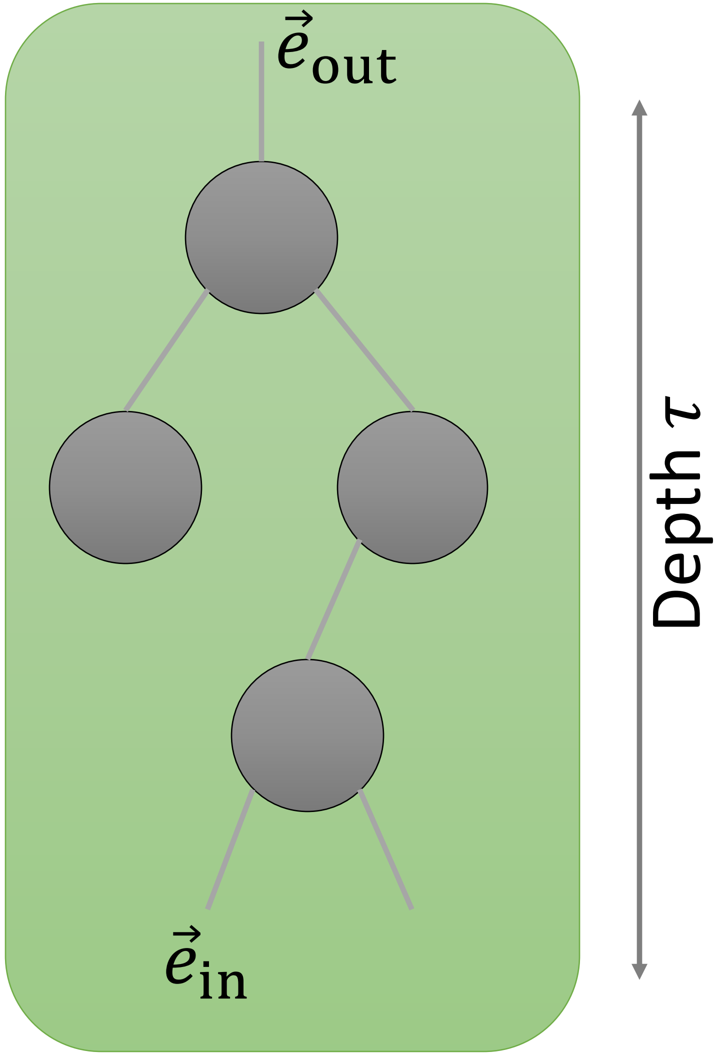
Invariant: $\vec{s} \times C = \vec{e} + x \cdot \vec{s} \times G$

$$C_{\text{mult}} = C_1 \times G^{-1}(C_2)$$

$$\begin{aligned}
\vec{s} \times C_1 \times G^{-1}(C_2) &= (\vec{e}_1 + x_1 \cdot \vec{s} \times G) \cdot G^{-1}(C_2) \\
&= \vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot \vec{s} \times G \times G^{-1}(C_2) \\
&= \vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot \vec{s} \times C_2 \\
&= \vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot (\vec{e}_2 + x_2 \cdot \vec{s} \times G) \\
&= (\vec{e}_1 \times G^{-1}(C_2) + x_1 \cdot \vec{e}_2) + x_1 x_2 \cdot \vec{s} \times G \\
&= \vec{e}_{\text{mult}} + x_1 x_2 \cdot \vec{s} \times G
\end{aligned}$$

$$||\vec{e}_{\text{mult}}||_\infty \leq N \cdot ||\vec{e}_1||_\infty + ||\vec{e}_2||_\infty \leq (N+1) \cdot \max\{||\vec{e}_1||, ||\vec{e}_2||\}$$

The GSW Scheme – Correctness



$$\|\vec{e}_{\text{out}}\|_\infty \leq (N+1)^{\tau+1} m \cdot \alpha q$$

Correctness:

$$n \cdot m \cdot (N+1)^{\tau+1} < q/4$$

$$\|\vec{e}_{\text{i+1}}\|_\infty \leq (N+1)\|\vec{e}_{\text{i}}\|_\infty$$

$$\|\vec{e}_{\text{in}}\|_\infty \leq m \cdot n = m \cdot \alpha q$$

The GSW Scheme – Semantic Security

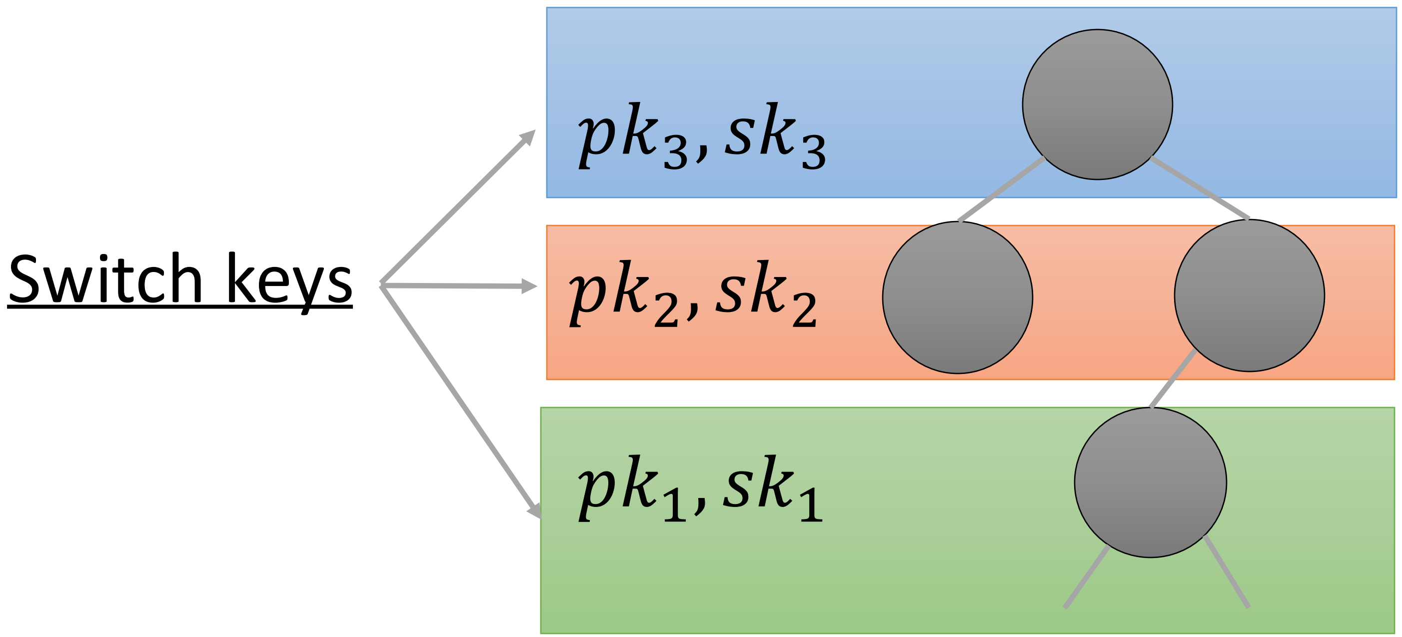
- Similar as in the proof of Regev PKE
- Using LWE we move to a **mental experiment** with $A \leftarrow_\$ \mathbb{Z}_q^{n\times m}$
- Hence, by the **leftover hash lemma**, with $m = \Theta(n \log q)$, the statistical distance between $(A, A \times \vec{r})$ and uniform is negligible
  - By a **hybrid argument** over the columns of R, it follows that the statistical distance between $(A, A \times R)$ and uniform is also negligible
  - Thus, the ciphertext **statistically hides** the plaintext

The GSW Scheme – Parameters

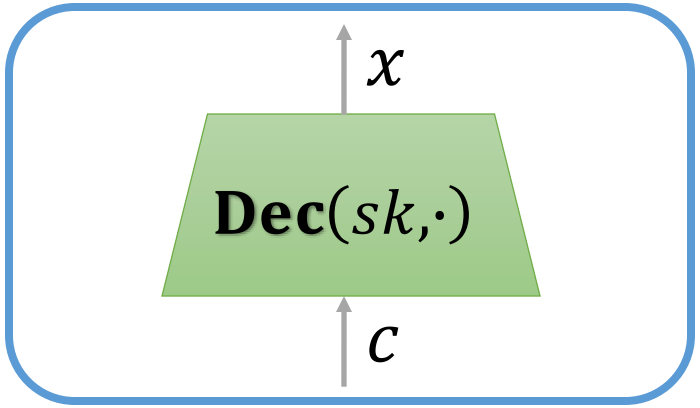
- **Correctness** requires $n \cdot m \cdot (N+1)^{\tau+1} < q/4$
- **Security** requires $m = \Theta(n \log q)$, e.g. $m \geq 1 + 2n(2 + \log q)$
- **Hardness** of LWE requires $q \leq 2^{n^\epsilon}$ for $\epsilon < 1$
  - Substituting we get $q > (2n \log q)^{\tau+3}$
  - And thus $n^\epsilon > (\tau + 3)(\log n + \log\log q + 1)$ which for large τ, n yields $n^\epsilon > 2\tau \log n$
  - So we set $n = \max(\lambda, \lceil 4\tau/\epsilon \log \tau^{1/\epsilon} \rceil)$, $q = \lceil 2^{n^\epsilon} \rceil$, $m = O(n^{1+\epsilon})$, and $\alpha = n/q = n \cdot 2^{-n^\epsilon}$
- Hence, the size of ciphertexts is polynomial in λ, τ thus yielding a **weakly-compact** FHE

Increasing the Homomorphic Capacity

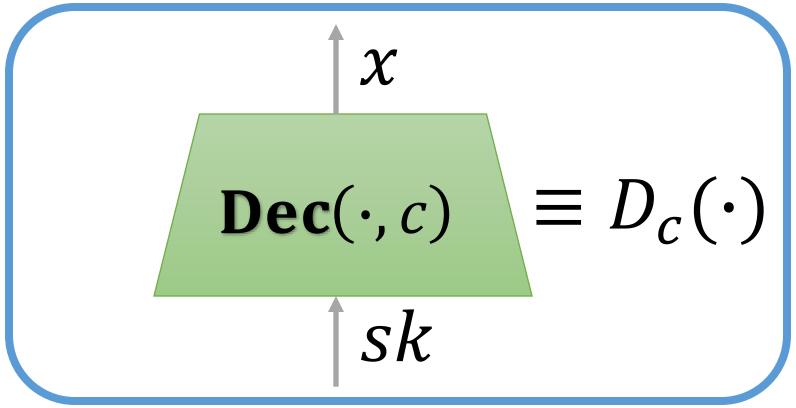
- The only way to increase the homomorphic capacity of GSW is to pick **larger parameters**
- This dependence can be **broken** using a trick by Gentry
- Main idea: Do a few operations, then **switch keys**



How to Switch Keys



Decryption circuit



Dual view

$$\begin{aligned}\mathbf{Eval}_{pk'}(D_c, aux) &= \mathbf{Eval}_{pk'}\left(D_c, \mathbf{Enc}_{pk'}(sk)\right)\\ &= \mathbf{Enc}_{pk'}(D_c(sk))\\ &= \mathbf{Enc}_{pk'}(x)\end{aligned}$$

Bootstrappable Encryption

- Let $W_\Pi(\lambda, \tau)$ be the set of all **fresh** and **evaluated** ciphertexts w.r.t. circuits class C_τ
  - For all possible keys and all possible inputs to the circuit
- Given $c_1, c_2 \in W_\Pi(\lambda, \tau)$, let $D^*_{c_1,c_2}(sk)$ be the **augmented decryption circuit**, defined by

$$\boxed{D^*_{c_1,c_2}(sk) = NAND(D_{c_1}(sk), D_{c_2}(sk))}$$

- We say that Π is **bootstrappable** if its homomorphic capacity includes all the augmented decryption circuits
  - I.e., $\exists \tau$ s.t. $\forall \lambda \in \mathbb{N}, c_1, c_2 \in W_\Pi(\lambda, \tau(\lambda))$, we have $D^*_{c_1,c_2} \in C_{\tau(\lambda)}$

Bootstrapping Theorem

Theorem. Any **bootstrappable homomorphic** encryption scheme can be transformed into a **compact somewhat homomorphic** encryption scheme

- One can show that the GSW scheme **is bootstrappable**
- Let Π be the bootstrappable scheme; construct Π' as follows:
  - $\mathbf{KGen}'(1^\lambda, 1^d)$: For each $i \in [0, d]$, run $(pk_i, sk_i) \leftarrow_\$ \mathbf{KGen}(1^\lambda, 1^\tau)$ and $\vec{c}_i^* \leftarrow_\$ \mathbf{Enc}(pk_{i+1}, sk_i)$, and output $sk' = (sk_0, \dots, sk_d), pk' = (pk_0, \vec{c}_1^*, \dots, \vec{c}_{d-1}^*, pk_d)$
  - $\mathbf{Enc}'(pk', x)$: Return $(0, c)$ where $c \leftarrow_\$ \mathbf{Enc}(pk_0, x)$
  - $\mathbf{Dec}'(sk', c')$: Return $\mathbf{Dec}(sk_i, c)$ where $c' = (i, c)$

Bootstrapping Theorem

- $\mathbf{Eval}'(pk', \Gamma, \vec{c})$: Go over the circuit in topological order from inputs to outputs; for every gate at level i with inputs $(i-1, c_1)$ and $(i-1, c_2)$, run $c' \leftarrow_\$ \mathbf{Eval}(pk_i, D^*_{c_1,c_2}, \vec{c}^*_{i-1})$ and use (i, c') as the gate output

- To prove **correctness**, we proceed by **induction**
  - The **auxiliary ciphertexts** $\vec{c}^*_{i-1}$, and fresh ciphertexts are correct
  - Assume that at level i two ciphertexts $c_1, c_2 \in W_\Pi(\lambda, \tau)$ are correct
  - Let $c' \leftarrow_\$ \mathbf{Eval}(pk_i, D^*_{c_1,c_2}, \vec{c}^*_{i-1})$; as Π is bootstrappable:

$$\mathbf{Dec}(sk_i, c') = D^*_{c_1,c_2}(sk_{i-1})$$
$$= NAND(D_{c_1}(sk_{i-1}), D_{c_2}(sk_{i-1})) = NAND(x_1, x_2)$$

  - Moreover, $c' \in W_\Pi(\lambda, \tau)$

SAPIENZA UNIVERSITÀ DI ROMA

Bootstrapping Theorem

- To prove **semantic security**, we use a **hybrid argument**
- In hybrid $\mathbf{H}_k(\lambda, b)$ we modify key generation by picking all ciphertexts $\vec{c}_i^*$ such that $i \geq k$ as fresh encryptions of $\vec{0}$
  - Note that $\mathbf{H}_d(\lambda, b)$ is just the semantic security game for Π'
  - By semantic security of Π, $\mathbf{H}_k(\lambda, b) \approx_c \mathbf{H}_{k-1}(\lambda, b)$ for each $k \in [0, d]$ and $b \in \{0,1\}$
  - Finally, $\mathbf{H}_0(\lambda, b)$ never uses sk_0, and thus by semantic security of Π **no PPT adversary** can distinguish between $\mathbf{H}_0(\lambda, 0)$ and $\mathbf{H}_0(\lambda, 1)$ with better than negligible probability

Circular Security

- The above scheme is **compact**, but **not fully homomorphic**, as we need a pair of keys **for each level** in the circuit
- A natural idea is to use a **single pair** (pk, sk) and include in pk' a ciphertext $\vec{c}^* \leftarrow_\$ \mathbf{Enc}(pk, sk)$
  - Correctness still holds for this variant, but the reduction to **semantic security breaks**
- Workaround: Assume **circular security**
  - I.e., $\mathbf{Enc}(pk, 0) \approx_c \mathbf{Enc}(pk, 1)$ even given $\vec{c}^* \leftarrow_\$ \mathbf{Enc}(pk, sk)$
  - GSW is **conjectured** to have this property, but no proof of this fact is currently known

Fully-Homomorphic Commitments

- Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times w}$ and $\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{R} + x \cdot \boldsymbol{G}$ for $\boldsymbol{R} \in \mathbb{Z}^{w\times m}$ and $x \in \mathbb{Z}_q$
  - Think of $\boldsymbol{C}$ as a **commitment** to x w.r.t. $\boldsymbol{A}$ under **randomness** $\boldsymbol{R}$
- **Homomorphic** operations:

$$\boldsymbol{G} - \boldsymbol{C}_1 = \boldsymbol{A}(-\boldsymbol{R}_1) + (1 - x_1) \cdot \boldsymbol{G}$$
$$\boldsymbol{C}_+ = \boldsymbol{C}_1 + \boldsymbol{C}_2 = \boldsymbol{A} \cdot (\boldsymbol{R}_1 + \boldsymbol{R}_2) + (x_1 + x_2) \cdot \boldsymbol{G}$$
$$\boldsymbol{C}_\times = \boldsymbol{C}_1 \cdot \boldsymbol{G}^{-1}[\boldsymbol{C}_2]$$
$$= \boldsymbol{A} \cdot (\boldsymbol{R}_1 \cdot \boldsymbol{G}^{-1}[\boldsymbol{C}_2]) + x_1\boldsymbol{G} \cdot \boldsymbol{G}^{-1}[\boldsymbol{A} \cdot \boldsymbol{R}_2 + x_2 \cdot \boldsymbol{G}])$$
$$\boldsymbol{A} \cdot (\boldsymbol{R}_1 \cdot \boldsymbol{G}^{-1}[\boldsymbol{C}_2] + x_1 \cdot \boldsymbol{R}_2) + x_1 x_2 \boldsymbol{G}$$

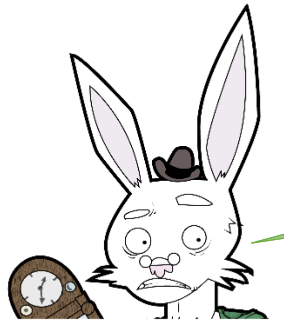
- Can be extended to **vectors** $\boldsymbol{x} \in \mathbb{Z}_q^L$

$$\boldsymbol{C} = \boldsymbol{A} \cdot \boldsymbol{R} + \boldsymbol{x}^{\mathrm{t}} \otimes \boldsymbol{G}$$

Proof Systems

$$L = \{x : \exists \zeta, \mathcal{V}(x,\zeta) = 1\}$$

Proof ζ

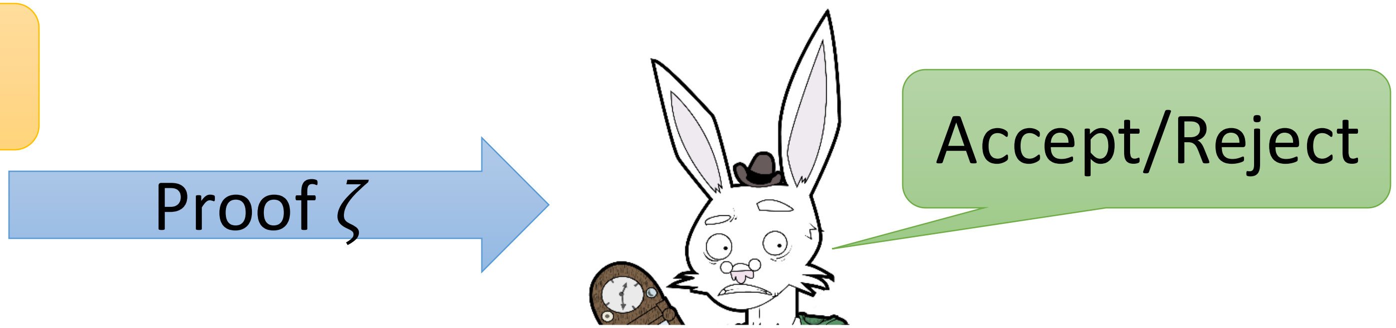


Accept/Reject

- A proof system π for **membership** in L is an algorithm $\mathcal{V}$ s.t.
  - **Completeness:** For all $x \in L$, then $\exists \zeta$ for which $\mathcal{V}(x,\zeta) = 1$
  - **Soundness:** For all $x \notin L$, then $\forall \zeta$ we have $\mathcal{V}(x,\zeta) = 0$
- Note the fact that a proof exists **might not** be efficiently verifiable
  - I.e., we would like the verifier to run in **polynomial time**

NP Proof Systems

$$L = \{x : \exists \zeta, \mathcal{V}(x, \zeta) = 1\}$$



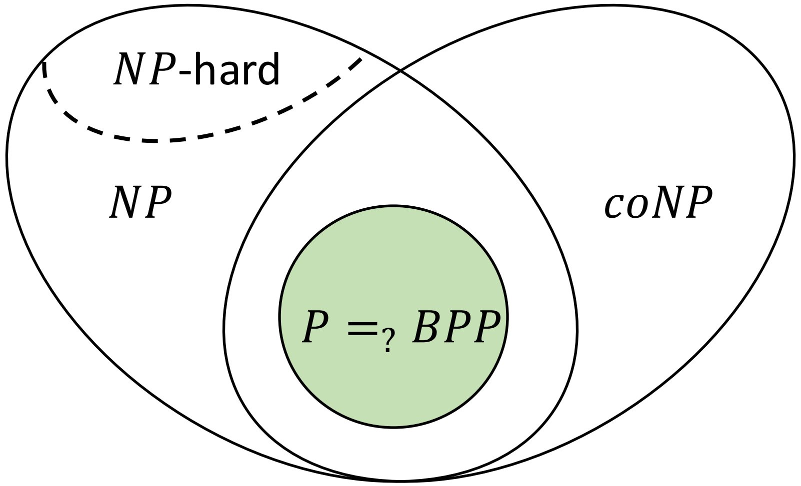
- An **NP** proof system π for membership in L is an algorithm $\mathcal{V}$ s.t.
  - **Completeness:** For all $x \in L$, then $\exists \zeta$ for which $\mathcal{V}(x, \zeta) = 1$
  - **Soundness:** For all $x \notin L$, then $\forall \zeta$ we have $\mathcal{V}(x, \zeta) = 0$
  - **Efficiency:** For all x, we have that $\mathcal{V}(x, \zeta)$ halts after $\mathrm{poly}(|x|)$ steps
- Note the running time is measured in terms of $|x|$
  - Necessarily, $|\zeta| = \mathrm{poly}(|x|)$

Examples

- Boolean satisfiability: $SAT = \{\phi(\cdot) : \exists w \in \{0,1\}^\lambda, \phi(w) = 1\}$
  - **Complete:** Every $L \in NP$ reduces to SAT
  - **Unstructured:** Decidable in time $e^{O(\lambda)}$
- Linear equations: $LIN = \{(A, b) : \exists w, A \cdot w = b\}$
  - **Structured:** Decidable in time $O(\lambda^{2.373}) = \mathrm{poly}(\lambda)$
- Quadratic residuosity: $QR_n = \{x : \exists w, x \equiv w^2 \bmod n\}$
  - **Structured:** QR_n is a subgroup of $\mathbb{Z}_n^*$
  - Yet, when $n = p \cdot q$ with $|p| = |q| = \lambda$ finding square roots is equivalent to factoring the modulus (time $e^{\tilde{O}(\lambda^{1/3})}$ on average)

The Class P

- $L \in P$ if there is a **polynomial-time** $\mathcal{A}$ such that
$$L = \{x : \mathcal{A}(x) = 1\}$$
  - $L \in BPP$: $\mathcal{A}$ is PPT and **errs** with probability $\leq 1/3$
- $L \in coNP$ if and only if its **complement** $\bar{L} \in NP$

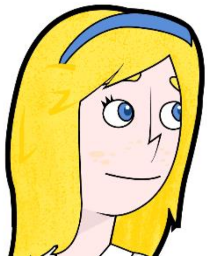


Proving Non-Membership

- How can we prove **non-membership**?
  - Showing $\phi \notin SAT$ requires to check that $\forall i \in [2^\lambda], \phi(w_i) = 0$
  - Showing $x \notin QR_n$ requires to check that $\forall i \in [\varphi(n)], x \not\equiv w_i^2 \bmod n$
- So, a naive proof is **exponentially** large
- We can avoid this if we allow the proof to use
  - **Randomness** (tolerate "error")
  - **Interaction** (add a computationally **unbounded** "prover")
  - S. Goldwasser, S. Micali, C. Rackoff. "The Knowledge Complexity of Interactive Proof-Systems." STOC 1985

Interactive Proof for $\overline{QR_n}$

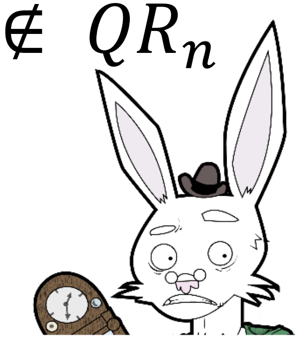
$$b'(z) = \begin{cases} 0 \text{ if } z \in QR_n \\ 1 \text{ if } z \notin QR_n \end{cases}$$



z

b'

$x \notin QR_n$



Check $b' = b$

$b \leftarrow_\$ \{0,1\}$

$y \leftarrow_\$ \mathbb{Z}_n^*$

$$z = \begin{cases} y^2 & \text{if } b = 0 \\ xy^2 & \text{if } b = 1 \end{cases}$$

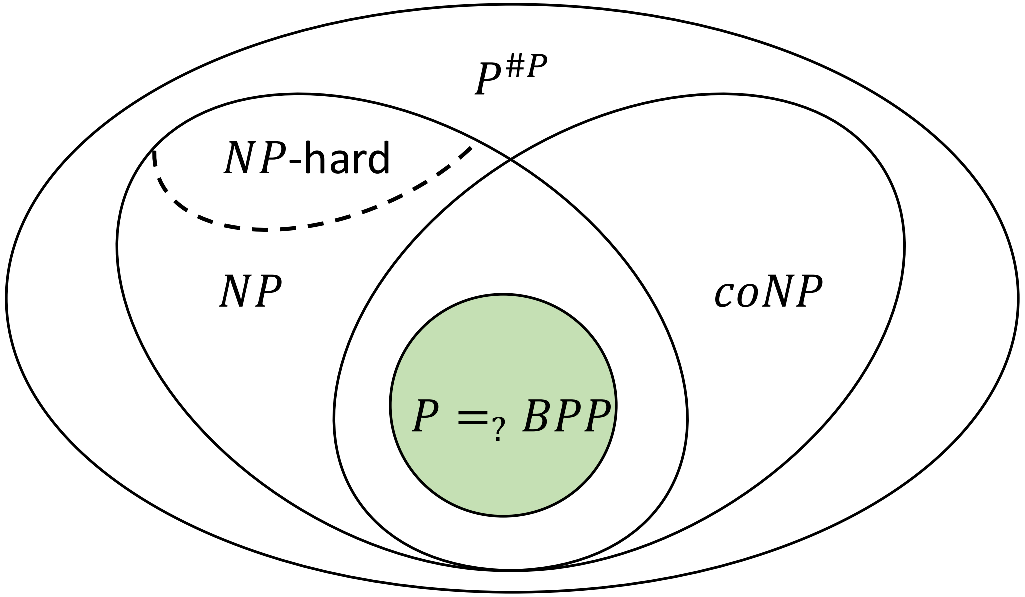
- **Completeness:**
  - We have $x \notin QR_n \Rightarrow y^2 \in QR_n \wedge xy^2 \notin QR_n$
- **Soundness:**
  - We have $x \in QR_n \Rightarrow y^2 \in QR_n \wedge xy^2 \in QR_n$
  - Hence, all even **unbounded** provers $\mathcal{P}^*$ succeed w.p. $1/2$

Interactive Proof Systems

- An interactive proof system π for L consists of a PPT $\mathcal{V}$ and an **unbounded** $\mathcal{P}$ such that
  - **Completeness:** For all $x \in L$, then $\mathbb{P}[\langle \mathcal{P}, \mathcal{V}(x) \rangle = 1] \geq 2/3$
  - **Soundness:** For all $x \notin L$, for all $\mathcal{P}^*$, then $\mathbb{P}[\langle \mathcal{P}^*, \mathcal{V}(x) \rangle = 1] \leq 1/3$
- Completeness and soundness can be bounded by any $c, s \colon \mathbb{N} \to [0,1]$ as long as
  - $c(|x|) \geq 1/2 + 1/\text{poly}(|x|)$ and $s(|x|) \leq 1/2 - 1/\text{poly}(|x|)$
  - So, $\text{poly}(|x|)$ repetitions yield $s(|x|) - c(|x|) \geq 1 - 2^{-\text{poly}(|x|)}$
  - The class NP has $c(|x|) = 1$ and $s(|x|) = 0$, whereas the class BPP requires **no interaction**

The Power of IP

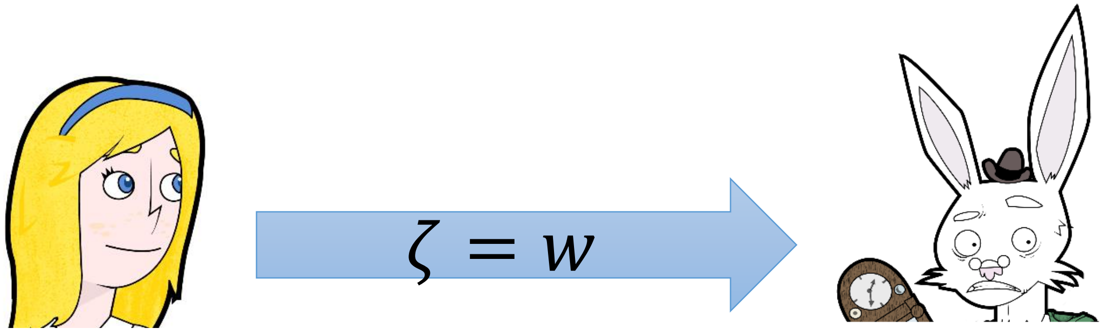
- We have shown that $\overline{QR_n} \in IP$
  - NP proof for $\overline{QR_n}$ **not self-evident**
  - This suggests that maybe $NP \subseteq IP$
  - Turns out that $\overline{SAT} \in IP$, and thus $coNP \subseteq IP$
  - In fact, $P^{\#P} \subseteq IP = PSPACE$



What Does a Proof Reveal?

- Consider the following **non-interactive** proof for QR_n

$x \in QR_n$



$x \in QR_n$ Check $x \equiv w^2 \bmod n$

  - **Generating** ζ requires exponential time
  - **Verifying** the proof requires $O(\lambda^2)$ time
- The verifier got something **for free** from seeing ζ
  - Recall that finding w is equivalent to factoring the modulus n

How to Define Zero-Knowledge?

- Intuitively, we might want that
  - The verifier does not learn w
  - The verifier does not learn any symbol of w
  - The verifier does not learn any information about w
  - The verifier does not learn anything (beyond $x \in L$)
- When does the verifier learn something?
  - If at the end of the protocol he can compute something he could not compute without running the protocol
- **Zero-knowledge:** Whatever can be computed while running the protocol could have been computed **without doing so**

Honest-Verifier Zero-Knowledge

- Hence, we must require that $\forall x \in L$ the verifier's view can be **efficiently simulated** given just x (but not w)
  - In other words, the verifier learns whether $x \in L$ but **nothing more**
  - Whatever he could compute via the protocol he could have computed by talking to himself (i.e., by running the simulator)
- An interactive proof system $\pi = (\mathcal{P}, \mathcal{V})$ for L is **perfect honest-verifier zero-knowledge** (HVZK) if $\exists$ PPT $\mathcal{S}$ such that $\forall x \in L$:

$$\mathcal{S}(x) \equiv \langle \mathcal{P}(x, w), \mathcal{V}(x) \rangle$$

  - Sanity check: Previous proof is **not** HVZK

SAPIENZA UNIVERSITÀ DI ROMA

Perfect Zero-Knowledge

- An interactive proof system $\pi = (\mathcal{P}, \mathcal{V})$ for L is **perfect zero-knowledge** (PZK) if $\forall$ PPT $\mathcal{V}^*$ $\exists$ PPT $\mathcal{S}$ s.t. $\forall x \in L, \forall z \in \{0,1\}^*$:

$$\boxed{\mathcal{S}^{\mathcal{V}^*}(x,z) \equiv \langle \mathcal{P}(x,w), \mathcal{V}^*(x,z) \rangle}$$

  - This is also known as **black-box zero-knowledge**
  - Simulator runs in time $\mathrm{poly}(|x|)$, but sometimes we will consider also simulation in **expected polynomial time**
- Auxiliary input captures **context**
  - Other protocol executions
  - A-priori information (in particular about w)

Can SAT be Proved in ZK?

- Why should we care?
  - Because it is an **NP-complete** language
  - If $SAT \in NP$, then **every** $L \in NP$ is provable in zero-knowledge

Theorem: If $SAT \in PZK$, then the polynomial-time hierarchy **collapses to the second level**

- Natural idea: Relax the definition of zero-knowledge
  - **Statistical zero-knowledge (SZK):** Simulator's output **statistically close** to the verifier's view (above theorem even holds for SZK)
  - **Computational zero-knowledge (CZK):** Simulator's output **computationally close** to the verifier's view (recall $\lambda = |x|$)

NP is in CZK

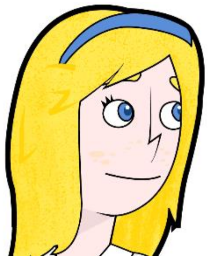
- One can show the following fundamental result:

Theorem: If **OWFs exist**, then $NP \subseteq CZK$.

- In fact, we will show that $HAM \subseteq CZK$, where HAM is the language of all graphs with an Hamiltonian cycle
  - This problem is NP complete

Zero-Knowledge for NP from FHE

$(pk, sk) \leftarrow_\$ \mathbf{KGen}(1^\lambda)$
$\vec{c} \leftarrow_\$ \mathbf{Enc}(pk, \vec{w})$
$d = \mathbf{Dec}(sk, c')$

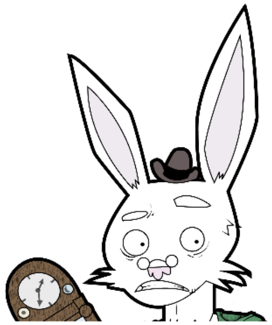


x, w

$\xrightarrow{pk, \vec{c}}$

$\xleftarrow{c'}$

$\xrightarrow{d}$



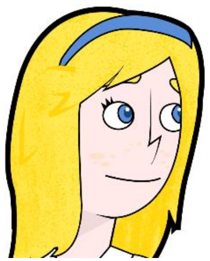
$x \in L$

$c' \leftarrow_\$ \mathbf{Eval}(pk, \Gamma_{R,x}, \vec{c})$

- Let $L \in NP$ with relation R
  - This means $L = \{x : \exists w \text{ s.t. } R(x, w) = 1\}$
  - Consider the circuit $\Gamma_{R,x}(w) = R(x, w)$
- The above protocol is **not sound**!
  - Can you say why?

Adding Soundness

$(pk, sk) \leftarrow_\$ \mathbf{KGen}(1^\lambda)$
$\vec{c} \leftarrow_\$ \mathbf{Enc}(pk, \vec{w})$
$d = \mathbf{Dec}(sk, c')$

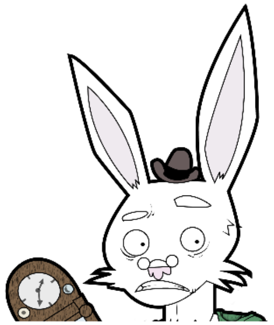


x, w

$pk, \vec{c}$ →

← c'

d →



$x \in L$

$\beta \leftarrow_\$ \{0,1\}$
$$c' \leftarrow_\$ \begin{cases} \mathbf{Eval}(pk, \Gamma_{R,x}, \vec{c}) & \text{if } \beta = 1 \\ \mathbf{Enc}(pk, 0) & \text{if } \beta = 0 \end{cases}$$
Check $\beta = d$

- Now soundness follows by the fact that, for $x \notin L$, **both ciphertexts** will be encryptions of zero
  - Since those are indistinguishable, Alice can cheat with probability 1/2
- However, we need to ensure that $pk, \vec{c}$ are **well formed**
  - Alice generates pk_1, pk_2 and Bob asks her to "open" one **at random**
  - With the other key Alice encrypts $\vec{w}_1, \vec{w}_2$ s.t. $\vec{w}_1 \oplus \vec{w}_2 = \vec{w}$, and Bob asks her to "open" one of the encryptions **at random**

Adding Zero-Knowledge

- The previous protocol is only **honest-verifier zero-knowledge**
  - In fact, malicious Bob could send to Alice the first ciphertext in the vector $\vec{c}$, so that d reveals **the first bit** of w
- This can be fixed using **commitments**
  - Namely, Alice sends a commitment to d
  - Hence, Bob must **reveal his randomness** in order to prove he run the computation as needed
  - Finally, Alice opens the commitment revealing d

Non-Interactive Proofs

- So far, we have seen how to obtain zero-knowledge proofs relying on **randomness** and **interaction**
- Can we remove interaction?
  - I.e., Alice sends a single message ζ to Bob to prove that $x \in L$
- As we shall see, **non-interactive** zero-knowledge (NIZK) proofs have exciting applications
  - E.g., post a proof on a website, or on a blockchain

A Negative Result

Theorem: If L admits a **NIZK** proof $(\mathcal{P}, \mathcal{V})$, then $L \in BPP$.

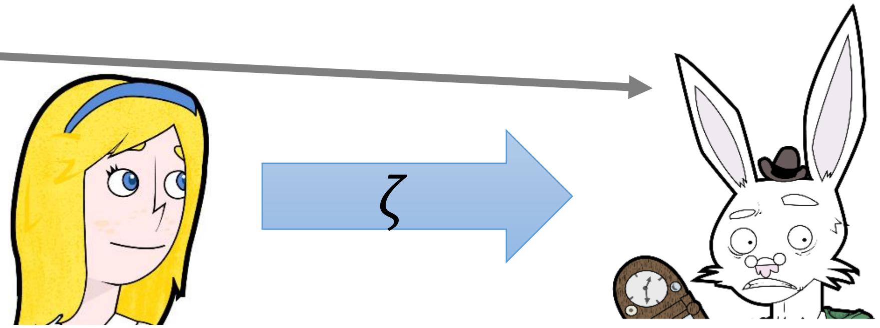
- Consider the following PPT machine deciding L:
  - Given x, run the simulator to obtain $\zeta \leftarrow_{\$} \mathcal{S}(x)$
  - Output the same as $\mathcal{V}(x, \zeta)$
- **Completeness:** If $x \in L$, the zero-knowledge property implies that a simulated proof should be accepting
- **Soundness:** If $x \notin L$, the verifier $\mathcal{V}$ rejects all proofs with high probability (in particular a simulated proof)

Common Reference String Model

- Main idea: Assume a **trusted setup**
  - Typically a common reference string (CRS) accessible to all parties
  - Sometimes just a uniformly random string
  - Need a **trusted party** to generate the CRS in a reliable manner
- Formally, a **non-interactive** proof system is a tuple $(\mathcal{G}, \mathcal{P}, \mathcal{V})$
  - $\mathcal{G}(1^\lambda)$: Outputs a CRS ω
  - $\mathcal{P}(\omega, x, w)$: Outputs a proof ζ
  - $\mathcal{V}(\omega, x, \zeta)$: Outputs a decision bit

Properties of NIZKs

$\omega \leftarrow_\$ \mathcal{G}(1^\lambda)$



x, w

$x \in L$

- **Completeness:** $\forall x \in L$,

$$\mathbb{P}\left[\mathcal{V}(\omega, x, \zeta) = 1 : \omega \leftarrow_\$ \mathcal{G}(1^\lambda), \zeta \leftarrow_\$ \mathcal{P}(\omega, x, w)\right] = 1$$

- **Soundness:** $\forall x \notin L, \forall \mathcal{P}^*$,

$$\mathbb{P}\left[\mathcal{V}(\omega, x, \zeta) = 1 : \omega \leftarrow_\$ \mathcal{G}(1^\lambda), \zeta \leftarrow_\$ \mathcal{P}^*(\omega, x)\right] \in \mathrm{negl}(\lambda)$$

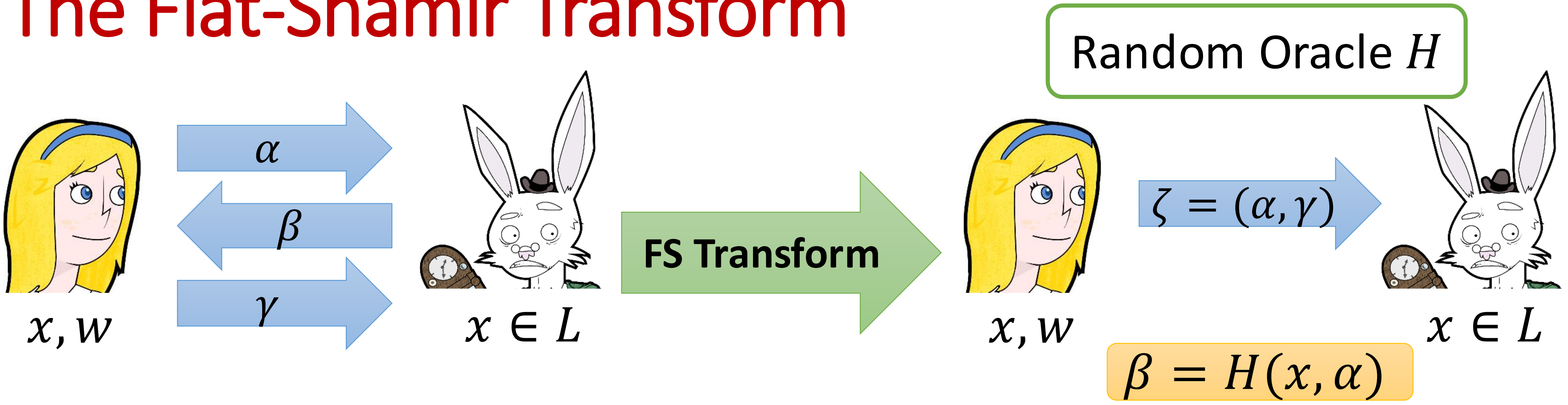
- **Zero-Knowledge:** $\exists$ PPT $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ s.t. $\forall x \in L$,

$$\{\omega, \mathcal{S}_1(\tau, x) : (\omega, \tau) \leftarrow_\$ \mathcal{S}_0(1^\lambda)\} \approx_c \{\omega, \mathcal{P}(\omega, x, w) : \omega \leftarrow_\$ \mathcal{G}(1^\lambda)\}$$

But Do NIZKs Exist?

- In the **random oracle** model:
  - A. Fiat, A. Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signatures Problems." CRYPTO 1986
- Assuming **Factoring**
  - U. Feige, D. Lapidot, A. Shamir. "Multiple Non-Interactive Zero-Knowledge Proofs based on a Single Random String." FOCS 1990
- In **bilinear** groups:
  - J. Groth, A. Sahai. "Efficient Non-Interactive Proof Systems for Bilinear Groups." SIAM Journal of Computing 41(5), 2012
- Assuming **LWE**
  - C. Peikert, S. Shiehian. "Non-Interactive Zero-Knowledge for NP from (Plain) LWE."

The Fiat-Shamir Transform



- Given **public-coin 3-round** protocol $(\mathcal{P}, \mathcal{V})$ we define its **FS-collapse** $(\mathcal{P}_{\mathrm{FS}}, \mathcal{V}_{\mathrm{FS}})$ as depicted above
  - $\mathcal{P}_{\mathrm{FS}}$ obtains α, γ from $\mathcal{P}$, using $\beta = H(x, \alpha)$
  - $\mathcal{V}_{\mathrm{FS}}$ checks that $\mathcal{V}$ accepts (α, β, γ), with $\beta = H(x, \alpha)$

The Fiat-Shamir Transform

Theorem: Assuming $(\mathcal{P}, \mathcal{V})$ is a 3-round public-coin **argument** for L with negligible **soundness** and **HVZK**, its FS-collapse $(\mathcal{P}_{\mathrm{FS}}, \mathcal{V}_{\mathrm{FS}})$ is a **NIZK argument** for L in the ROM

- **Remark:** Arguments versus proofs
  - An argument has only **computational** (rather than statistical) **soundness**
- Actually, the FS-collapse is even a **NIZK-PoK** in the ROM
  - S. Faust, G. A. Marson, M. Kholweiss, D. Venturi. "On the Non-Malleability of the Fiat-Shamir Transform." Indocrypt 2012

Analysis in the ROM

- Suppose $\exists x \notin L$ and some $\mathcal{P}_{\mathrm{FS}}^*$ producing an **accepting proof**
  - Assume $\mathcal{P}_{\mathrm{FS}}^*$ makes $p \in \mathrm{poly}(\lambda)$ queries to the RO, and makes $\mathcal{V}_{\mathrm{FS}}$ accept with probability $\epsilon(\lambda)$
  - We will construct $\mathcal{P}^*$ **breaking soundness** w.p. $\mathrm{poly}(\epsilon, 1/p)$
- We rely on the following useful fact:
  - Let $\mathbf{X}, \mathbf{Y}$ be **correlated** random variables such that $\mathbb{P}[E(\mathbf{X}, \mathbf{Y})] \geq \epsilon$ where E is some event
  - Then for at least an $\epsilon/2$ fraction of x's, $\mathbb{P}[E(x, \mathbf{Y})] \geq \epsilon/2$
  - Assume not, and call good an x for which the statement holds

$$\mathbb{P}[E(\mathbf{X}, \mathbf{Y})] = \mathbb{P}[\mathbf{Good}] \cdot \mathbb{P}[E(\mathbf{X}, \mathbf{Y}) | \mathbf{Good}] + \mathbb{P}[\mathbf{Bad}] \cdot \mathbb{P}[E(\mathbf{X}, \mathbf{Y}) | \mathbf{Bad}] < \epsilon/2 \cdot 1 + 1 \cdot \epsilon/2$$

SAPIENZA UNIVERSITÀ DI ROMA

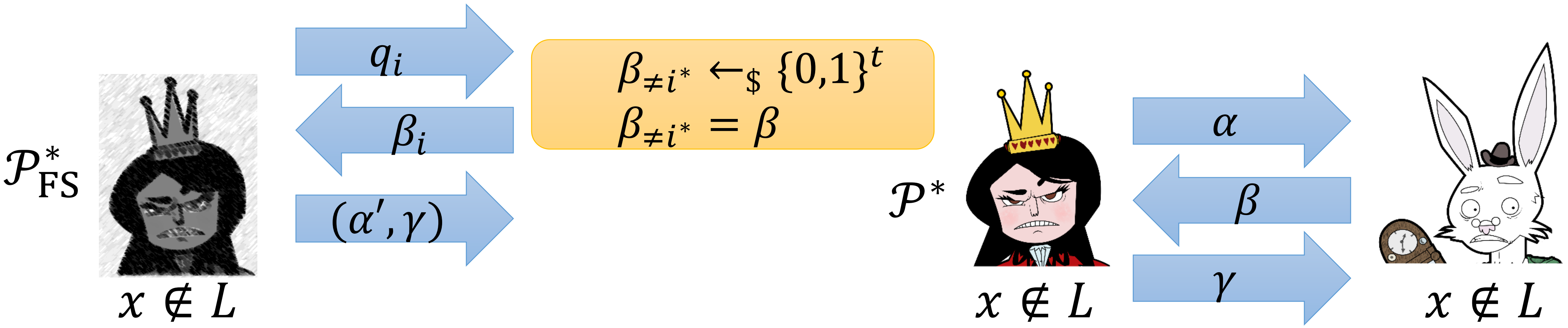
Analysis in the ROM

- Let (α, γ) be the proof output by $\mathcal{P}^*_{\mathrm{FS}}$
- Denote by $(q_1, \dots, q_p)$ the RO queries asked by $\mathcal{P}^*_{\mathrm{FS}}$
  - Each query is a pair (x_i, α_i)
  - Wlog. assume all queries are **distinct** and $\exists i^* \in [p]$ s.t. $q_{i^*} = (\alpha, x)$

Forking Lemma. For an $\epsilon/2p$ fraction of $(q_1, \dots, q_{i^*})$ it holds that $\mathcal{P}^*_{\mathrm{FS}}$ **wins** w.p. $\epsilon/2p$ **conditioned** on $\mathbf{q}_{i^*} = (\alpha, x)$ and $\mathbf{q}_i = q_i$ $(\forall i \leq i^*)$

- Proof: $\exists i^*$ s.t. $\mathcal{P}^*_{\mathrm{FS}}$ wins w.p. ϵ/p conditioned on $\mathbf{q}_{i^*} = (\alpha, x)$
  - As otherwise $\mathcal{P}^*_{\mathrm{FS}}$ does not have advantage $\geq \epsilon$
  - The statement then follows directly by the **useful fact**

Analysis in the ROM



- The prover $\mathcal{P}^*$ acts as follows
  - Run $\mathcal{P}^*_{\mathrm{FS}}$ and answer all RO queries q_i with $i < i^*$ at **random**
  - Upon input the query q_{i^*} with $\alpha \in q_{i^*}$, forward α to $\mathcal{V}$ and receive β
  - Use β as the answer to RO query q_{i^*}
  - Upon (α', γ), **hope** that $\alpha' = \alpha$

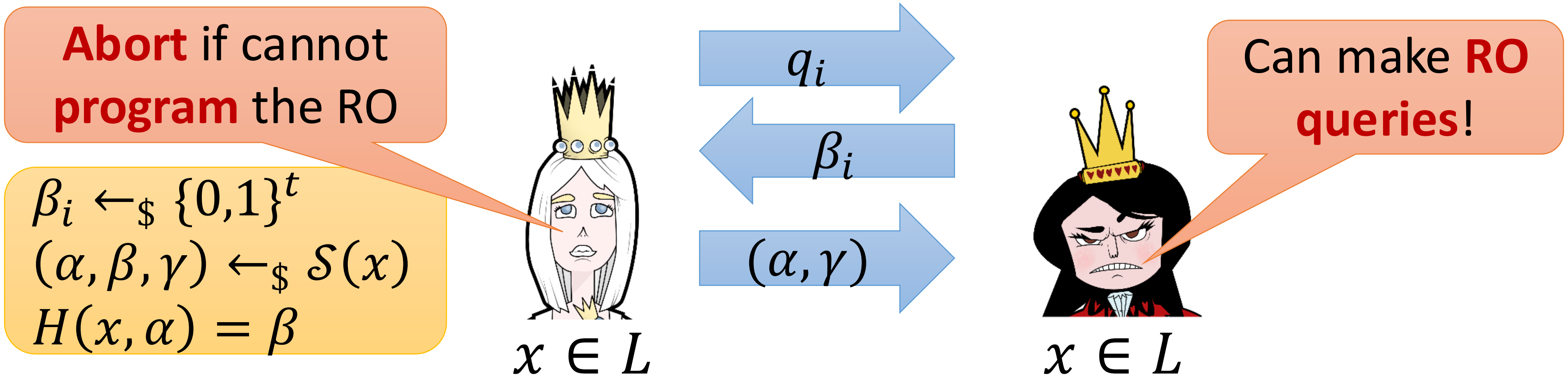
Analysis in the ROM

- By the **forking lemma**, we get that w.p. $\epsilon/2p$ over the choice of $(\mathbf{q}_1, \dots, \mathbf{q}_{i^*})$, $\mathcal{P}^*_{\mathrm{FS}}$ wins w.p. $\epsilon/2p$ conditioned on $\alpha' = \alpha$
- Hence:

$$\mathbb{P}[\mathcal{P}^* \text{ wins}] \geq \left(\frac{\epsilon}{2p}\right)^2$$

  - Since this is **non-negligible**, then soundness follows
- It remains to prove **zero-knowledge**
  - But we did not yet defined what zero-knowledge in the ROM means
  - Typically, the simulator is allowed to **program the random oracle**

Analysis in the ROM



- Let $\mathcal{S}$ be the **HVZK simulator** for the public-coin protocol
- The **NIZK simulator** $\mathcal{S}_{\mathrm{FS}}$:
  - Answer RO query $q_i = (\alpha_i, x_i)$ with random β_i
  - Upon input $x \in L$, run $(\alpha, \beta, \gamma) \leftarrow_\$ \mathcal{S}(x)$ and program $H(x, \alpha) = \beta$
  - Abort if (x, α) was previously queried to the RO
- **Non-triviality:** Need that α is **unpredictable**!

On Adaptive Soundness

- Our definition of soundness for NIZKs is **non-adaptive**
  - In particular, the choice of $x \notin L$ **cannot depend on the CRS**
  - One can show that the Fiat-Shamir transform actually achieves **adaptive soundness**
- Note that the FS-collapse defines $\beta = H(x, \alpha)$, i.e. we hash both the **statement** x and the **commitment** α
  - Sometimes, a variant where $\beta = H(\alpha)$ is also used
  - However, this might not be adaptively sound leading to **actual attacks** in some applications
  - D. Bernhard, O. Pereira, B. Warinschi. "How not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios." ASIACRYPT 2012

Generalization to Multi-Round Protocols

- The FS transform can be generalized to **constant-round** public-coin arguments
  - The prover $\mathcal{P}_{\mathrm{FS}}$ hashes the **current view** $(x, \alpha_1, \dots, \alpha_{i-1})$ in order to obtain the i-th message β_i from the verifier $\mathcal{V}$
  - A non-interactive proof now consists of $\zeta = (\alpha_1, \dots, \alpha_n)$
- This is also known to be **tight**
  - There exists a **non-constant-round** public-coin argument for which the FS-collapse is **not sound** (even in the ROM)
  - Consider any constant-round public-coin argument with constant soundness, and **amplify** soundness by **sequential repetition**
  - This yields negligible soundness in non-constant rounds
  - But the reduction does not yield negligible soundness anymore

Fiat-Shamir without Random Oracles?

- Natural question: Can we instantiate the random oracle using an **explicit hash family**?
  - Understand **which properties** of a random oracle are necessary for proving security of the Fiat-Shamir transform in the CRS model
- Unfortunately, this is **not** possible for **all** 3-round public-coin proofs/arguments
  - S. Goldwasser, Y. T. Kalai. "On the (in)security of the Fiat-Shamir paradigm." FOCS 2003
  - N. Bitansky, D. Dachman-Soled, S. Garg, A. Jain, Y. T. Kalai, A. Lopez-Alt, D. Wichs. "Why Fiat-Shamir for Proofs Lacks a Proof." TCC 2013
  - Still **possible** for some **specific** class of protocols

Correlation Intractability

- Let $\mathcal{H} = \{h: \{0,1\}^s \to \{0,1\}^t\}$ be a family of hash functions
  - Consider any relation $R \subseteq \{0,1\}^s \times \{0,1\}^t$
- We say that $\mathcal{H}$ is R-**correlation-intractable** if for all PPT $\mathcal{A}$:

$$\mathbb{P}[(x, h(x)) \in R : h \leftarrow_\$ \mathcal{H}; x \leftarrow_\$ \mathcal{A}(h)] \in \mathrm{negl}(\lambda)$$

- A relation R is said to be ρ-**sparse**, if $\forall x \in \{0,1\}^s$:

$$\mathbb{P}[(x, y) \in R : y \leftarrow_\$ \{0,1\}^t] \leq \rho(\lambda)$$

  - Moreover, the relation R is **sparse** if $\rho(\lambda) \in \mathrm{negl}(\lambda)$

Fiat-Shamir via Correlation Intractability

Theorem: Assuming $\pi = (\mathcal{P}, \mathcal{V})$ is a 3-round public-coin **proof** for L with **soundness** and **HVZK**, its FS-collapse $(\mathcal{P}_{\mathrm{FS}}, \mathcal{V}_{\mathrm{FS}})$ using a **CI** hash family $\mathcal{H}$ is a **NIZK argument** for L

- Consider the relation:

$$R_{L,\pi} = \{((\alpha, x), \beta) : \exists \gamma \text{ s.t. } x \notin L \wedge \mathcal{V}(x, (\alpha, \beta, \gamma)) = 1\}$$

  - It is not hard to show that **statistical soundness** (with negligible soundness error) implies that R_π is **sparse**
  - But a cheating $\mathcal{P}_{\mathrm{FS}}^*$ finds α^* s.t. $((x, \alpha^*), h(x, \alpha^*)) \in R_{L,\pi}$, **violating CI**

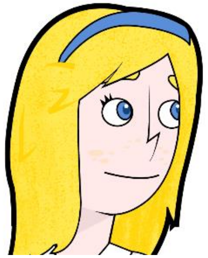
Fiat-Shamir via Correlation Intractability

- Zero-knowledge additionally requires that $\mathcal{H}$ is **programmable**
  - Call $\mathcal{H}$ 1-**universal** if for all $x \in \{0,1\}^s, y \in \{0,1\}^t$, the probability over the choice of $h \in \mathcal{H}$ that $h(x) = y$ equals 2^{-t}
  - $\mathcal{H}$ is **programmable** if it is 1-**universal** and further there exists an **efficient** algorithm $\mathbf{Samp}(1^\lambda, x, y)$ that samples from the conditional distribution $h \leftarrow_\$ \mathcal{H}$ such that $h(x) = y$
- We can assume programmability wlog.
  - Sample $h \leftarrow_\$ \mathcal{H}$ and a random string $u \leftarrow_\$ \{0,1\}^t$
  - Output $h(x) \oplus u$
  - Algorithm $\mathbf{Samp}(1^\lambda, x, y)$ picks $h \leftarrow_\$ \mathcal{H}$ and outputs $(h, h(x) \oplus y)$

Fiat-Shamir via Correlation Intractability

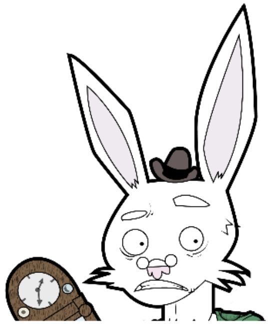
- Assuming **obfuscation**:
  - Y. T. Kalai, G. N. Rothblum, R. D. Rothblum. "From Obfuscation to the security of Fiat-Shamir for Proofs." CRYPTO 17
- Assuming **optimal KDM-secure** encryption:
  - R. Canetti, Y. Chen, L. Reyzin, R. D. Rothblum. "Fiat-Shamir and CI from Strong KDM-Secure Encryption" EUROCRYPT 18
- Assuming **circularly secure** FHE:
  - R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, R. D. Rothblum, D. Wichs. "Fiat-Shamir: From Theory to Practice." STOC 19
- Assuming **(plain) LWE**:
  - C.Peikert, S. Shiehian. "Noninteractive Zero Knowledge from (Plain) Learning With Errors." CRYPTO 19

Questions?



Cryptography Course

Prof. Daniele Venturi
Dipartimento di Informatica



SAPIENZA
UNIVERSITÀ DI ROMA

Academic Year 2024/2025

References

- [Ajt96] Miklós Ajtai: *Generating hard instances of lattice problems (extended abstract).* STOC 1996
- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, Amit Sahai: *Fast cryptographic primitives and circular-secure encryption based on hard learning problems*. CRYPTO 2009
- [GGM84] Oded Goldreich, Shafi Goldwasser, Silvio Micali: How to construct random functions (extended abstract). FOCS 1984
- [Mic01] Daniele Micciancio: *Improving lattice based cryptosystems using the Hermite normal form.* CaLC 2001
- [NR95] Moni Naor, Omer Reingold: Synthesizers and their application to the parallel construction of psuedo-random functions. FOCS 1995
- [NR97] Moni Naor, Omer Reingold: Number-theoretic constructions of efficient pseudo-random functions. FOCS 1997
- [Pei10] Chris Peikert: *An efficient and parallel Gaussian sampler for lattices*. CRYPTO 2010
- [Reg05] Oded Regev: *On lattices, learning with errors, random linear codes, and cryptography*. STOC 2005
- [Sho94] Peter W. Shor: *Algorithms for quantum computation: discrete logarithms and factoring*. FOCS 1994
- [NRR00] Moni Naor, Omer Reingold, Alon Rosen: Pseudo-random functions and factoring (extended abstract). STOC 2000
- [BPR12] Abhishek Banerjee, Chris Peikert, Alon Rosen: *Pseudorandom functions and lattices*. EUROCRYPT 2012

SAPIENZA
UNIVERSITÀ DI ROMA

References

- [AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, Daniel Wichs: *Learning with rounding, revisited - New reduction, properties and applications*. CRYPTO 2013
- [Bab86] László Babai: *On Lovász' lattice reduction and the nearest lattice point problem*. Comb. 6(1) 1986
- [Ajt99] Miklós Ajtai: *Generating hard instances of the short basis problem*. ICALP 1999
- [GPV08] Craig Gentry, Chris Peikert, Vinod Vaikuntanathan: *Trapdoors for hard lattices and new cryptographic constructions*. STOC 2008
- [P10] Chris Peikert: *An Efficient and Parallel Gaussian Sampler for Lattices*. CRYPTO 2010
- [AP09] Joël Alwen, Chris Peikert: *Generating shorter bases for hard random lattices*. STACS 2009
- [MP12] Daniele Micciancio, Chris Peikert: *Trapdoors for lattices: simpler, tighter, faster, smaller*. EUROCRYPT 2012
- [Kle01] Philip N. Klein: *Finding the closest lattice vector when it's unusually close*. SODA 2000
- [CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, Chris Peikert: *Bonsai trees, or how to delegate a lattice basis*. EUROCRYPT 2010