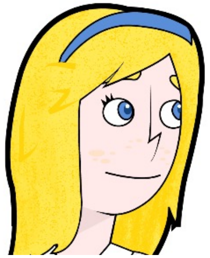
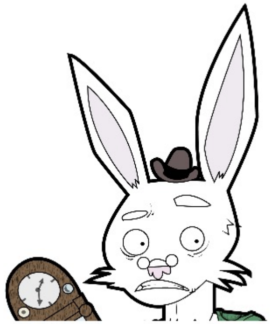
Post-quantum Cryptography

Data Privacy and Security



Prof. Daniele Venturi
Dipartimento di Informatica



SAPIENZA
UNIVERSITÀ DI ROMA

Academic Year 2024-2025

Modern Cryptography

- Cryptography is **everywhere**
  - Credit cards, electronic passports, electronic commerce,electronic voting, cryptocurrencies, …
- **Provable** security: **Reductions** to solving **hard problems**, given an attacker breaking security of cryptographic primitives
  - Requires to **believe** $P \neq NP$ (and in fact, that OWFs exist)
  - Examples: factoring, discrete logarithm, bilinear maps…
- History of **success**
  - Secret-key cryptography, public-key cryptography, identity-based cryptography, attribute-based cryptography, program obfuscation, …

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?

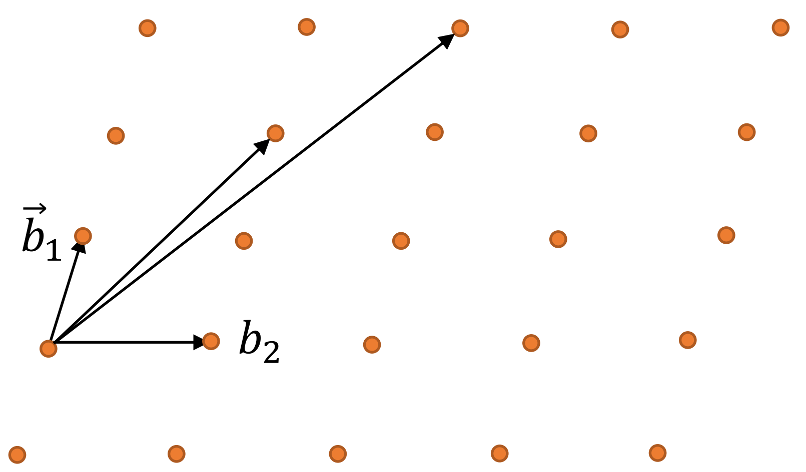
- Big quantum computers won’t be available for **many years**
  - If **ever**…
  - Can’t we just wait?
- Better safe than sorry
  - **Harvesting attacks:** Store today’s keys/ciphertexts to break later
  - **Rewrite history:** Forge signatures for old keys
  - Deploying new cryptography **at scale** 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 $(\vec{b}_1, \dots, \vec{b}_n)$ in $\mathbb{R}^n$ and consider all **integer** combinations

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



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

History

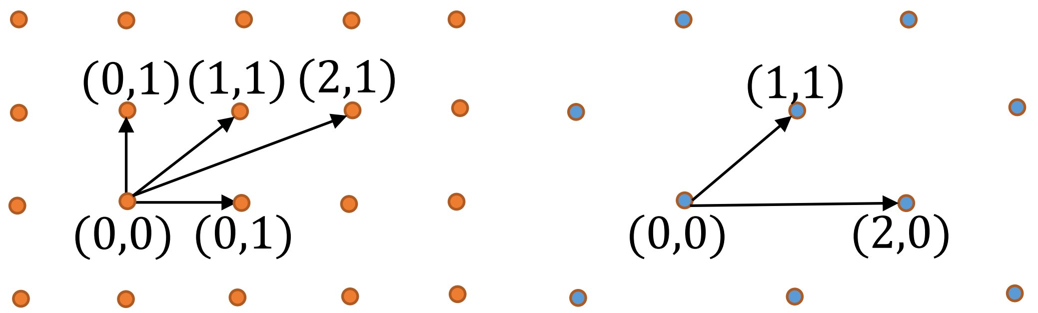
- **Geometric** objects with rich mathematical structure
- Considerable **mathematical interest** starting from Gauss (1801), Hermite (1850), and Minkowski (1896)



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

Equivalent Bases

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

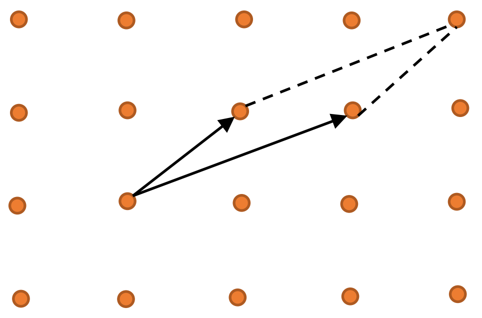


- **Equivalent** bases:
  - Permute vectors (i.e.,$\vec{b}_i \leftrightarrow \vec{b}_j$)
  - Negate vectors (i.e.,$\vec{b}_i \leftarrow (-\vec{b}_i)$)
  - Add integer multiple of another vector (i.e.,$\vec{b}_i \leftarrow \vec{b}_i + k \cdot \vec{b}_j, k \in \mathbb{Z}$)

- **Theorem:** Two bases B_1, B_2 are **equivalent** iff $B_2 = B_1 \cdot U$
  - U **unimodular** (i.e., integer matrix with $\det(U) = \pm 1$)

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**

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

- Useful for measuring arbitrary points **relative to a lattice**
  - Note $x \bmod \mathcal{P}(B) = (a_1 \mathrm{mod} 1)\vec{b}_1 + \cdots + (a_n \mathrm{mod} 1)\vec{b}_n$
  - A point x is in a lattice iff $x \bmod \mathcal{P}(B) = (0, \dots, 0)$

Determinant

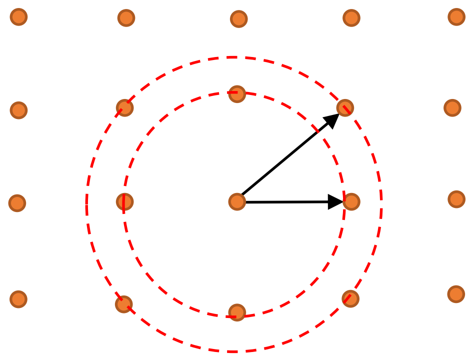
- The **determinant** of a lattice $\mathcal{L}(B)$ is $\det(\mathcal{L}) = |\det(B)|$
- Note that this is well defined, as for every **unilateral** U

$$|\det(B \cdot U)| = |\det(B) \cdot \det(U)| = \det(B)$$

- 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

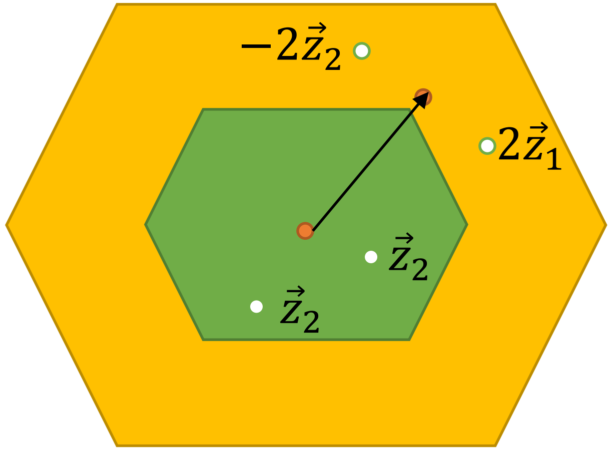
- Let $\lambda_1(\mathcal{L})$ be the length of the **shortest non-zero** vector in a lattice $\mathcal{L}$
  - Usually, in terms of the **Euclidean** norm
  - The shortest vector is **never unique**, as for every $\vec{v} \in \mathcal{L}$ also $-\vec{v} \in \mathcal{L}$
- More generally, $\lambda_k(\mathcal{L})$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



SAPIENZA UNIVERSITÀ DI ROMA

Minkowski's Theorem

- **Lemma (Blichfeld):** For any lattice $\mathcal{L}$ and set $\mathcal{S}$ with $\mathrm{vol}(\mathcal{S}) > \det(\mathcal{L})$, there exist distinct $\vec{z}_1, \vec{z}_2 \in \mathcal{S}$ we have that $\vec{z}_1 - \vec{z}_2 \in \mathcal{L}$
  - The proof is simple and only requires volume arguments (exercise)
- **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 $\mathcal{S}$



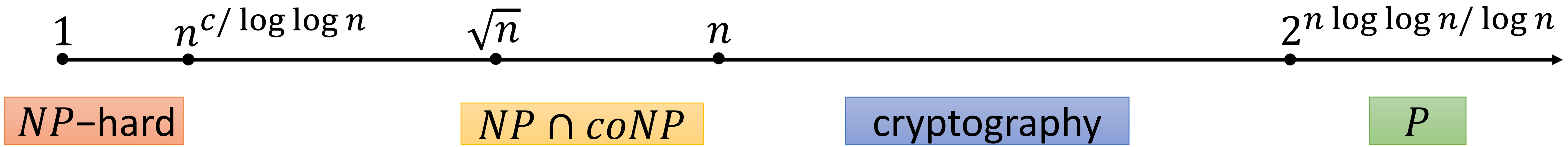
- Let $\vec{z}_1, \vec{z}_2 \in \mathcal{S}/2$; by Blichfeld $\vec{z}_1 - \vec{z}_2 \in \mathcal{L}$
- Now, $2\vec{z}_1, -2\vec{z}_2 \in \mathcal{S}$
- So, their average $\vec{z}_1 - \vec{z}_2 \in \mathcal{S}$
- **Corollary (Minkowski):** For every $\mathcal{L}$, we have that $\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$

Hard Problems

- $\mathbf{SVP}_\gamma$: Given B, find a vector in $\mathcal{L}(B)$ with length $\leq \gamma \cdot \lambda_1(\mathcal{L}(B))$
- $\mathbf{GapSVP}_\gamma$: Given B, **decide** if $\lambda_1(\mathcal{L}(B))$ is ≤ 1 or $\geq \gamma$
- $\mathbf{SIVP}_\gamma$: Given B, find n **linearly independent** vectors in $\mathcal{L}(B)$ with length $\leq \gamma \cdot \lambda_n(\mathcal{L}(B))$
- $\mathbf{CVP}_\gamma$: Given B and $\vec{v}$, 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 $\vec{v}$ is **already close**

SAPIENZA
UNIVERSITÀ DI ROMA

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 $\gamma = \sqrt{n}$

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

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

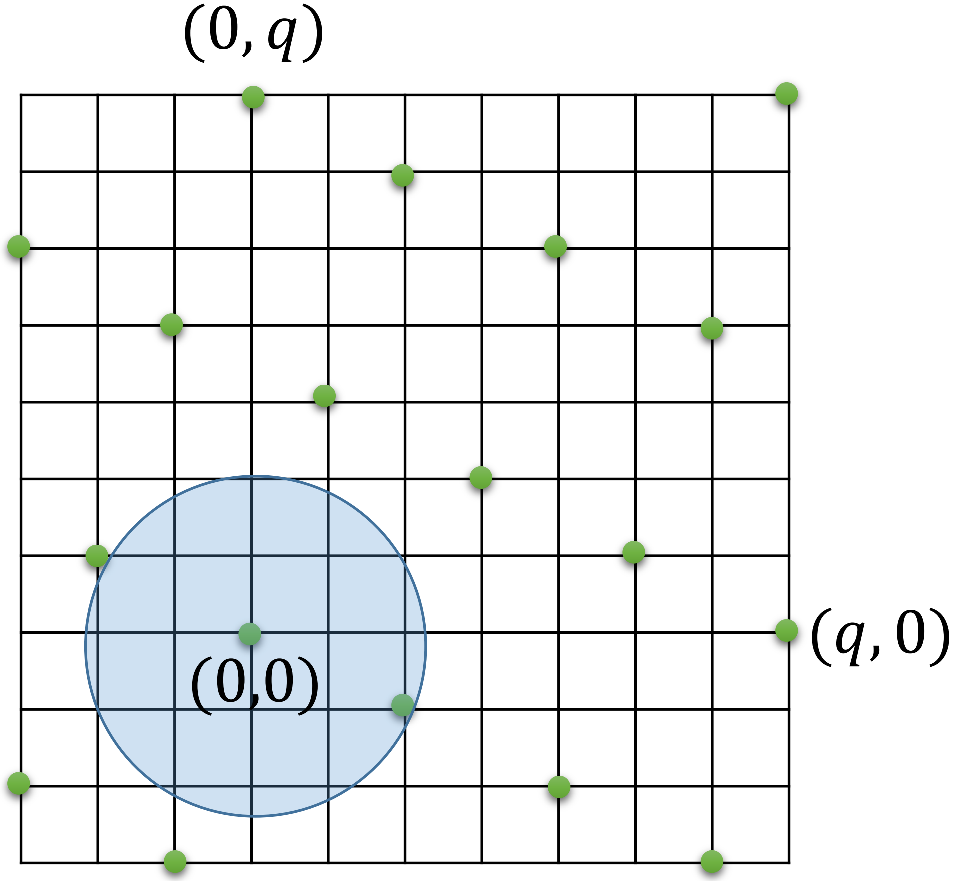
- Observations:
  - Trivial if the size of the z_i's is **not restricted** (Gaussian elimination)
  - Equivalently, find **non-zero short** $\boldsymbol{z} \in \mathbb{Z}^m$ s.t. $\boldsymbol{A} \cdot \boldsymbol{z} = \boldsymbol{0} \in \mathbb{Z}_q^n$

SIS as a Lattice Problem

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

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

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)

SAPIENZA UNIVERSITÀ DI ROMA

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

$$\boldsymbol{s}^{t} \cdot \boldsymbol{A} + \boldsymbol{e}^{t} = \boldsymbol{b}^{t}$$

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

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

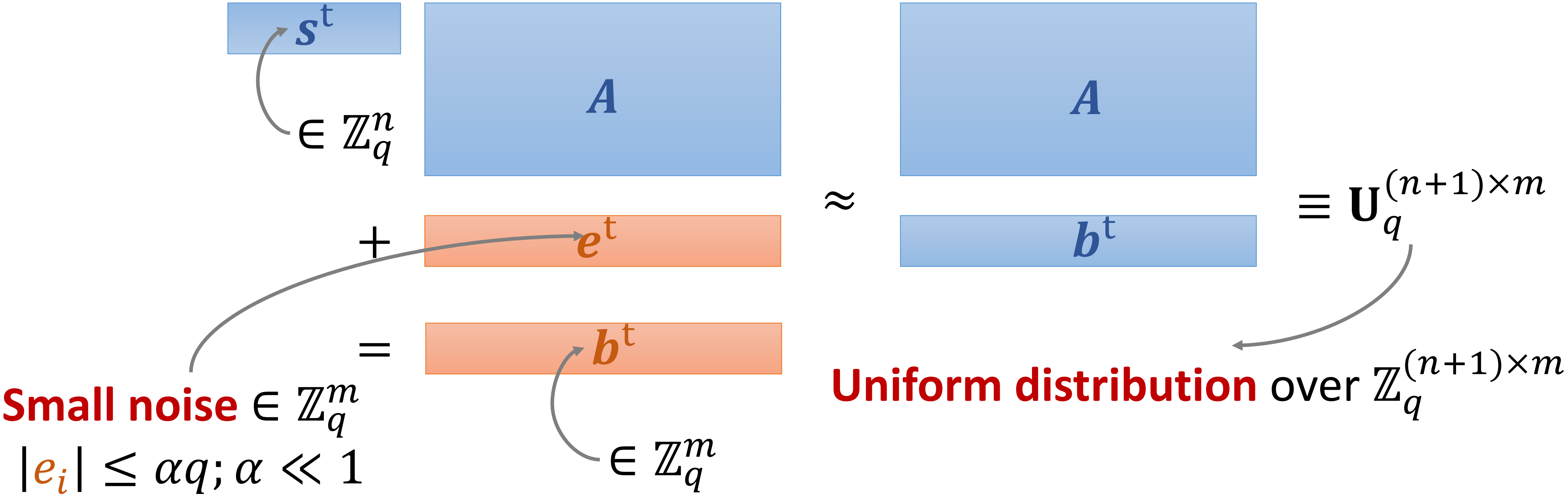
Small noise $\boldsymbol{e} \in \mathbb{Z}_q^m$

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

- 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

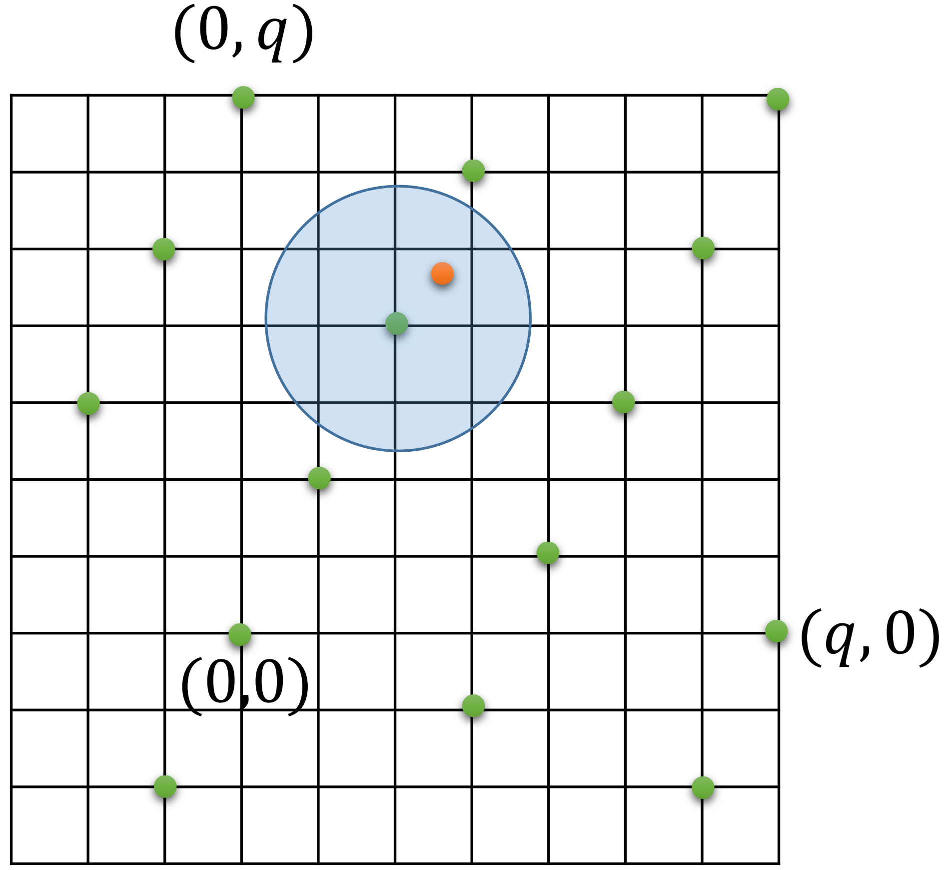


LWE as a Lattice Problem

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

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

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 $\mathbb{P}[|e| > \alpha q : e \leftarrow \chi] \leq \mathrm{negl}(n)$

SAPIENZA UNIVERSITÀ DI ROMA

Simple Properties

- Check a **candidate** solution $\boldsymbol{t} \in \mathbb{Z}_q^n$
  - Test if all $b - \langle \boldsymbol{t}, \boldsymbol{a} \rangle$ are small
  - If $\boldsymbol{t} \neq \boldsymbol{s}$, then $b - \langle \boldsymbol{t}, \boldsymbol{a} \rangle = \langle \boldsymbol{s} - \boldsymbol{t}, \boldsymbol{a} \rangle + e$ 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

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

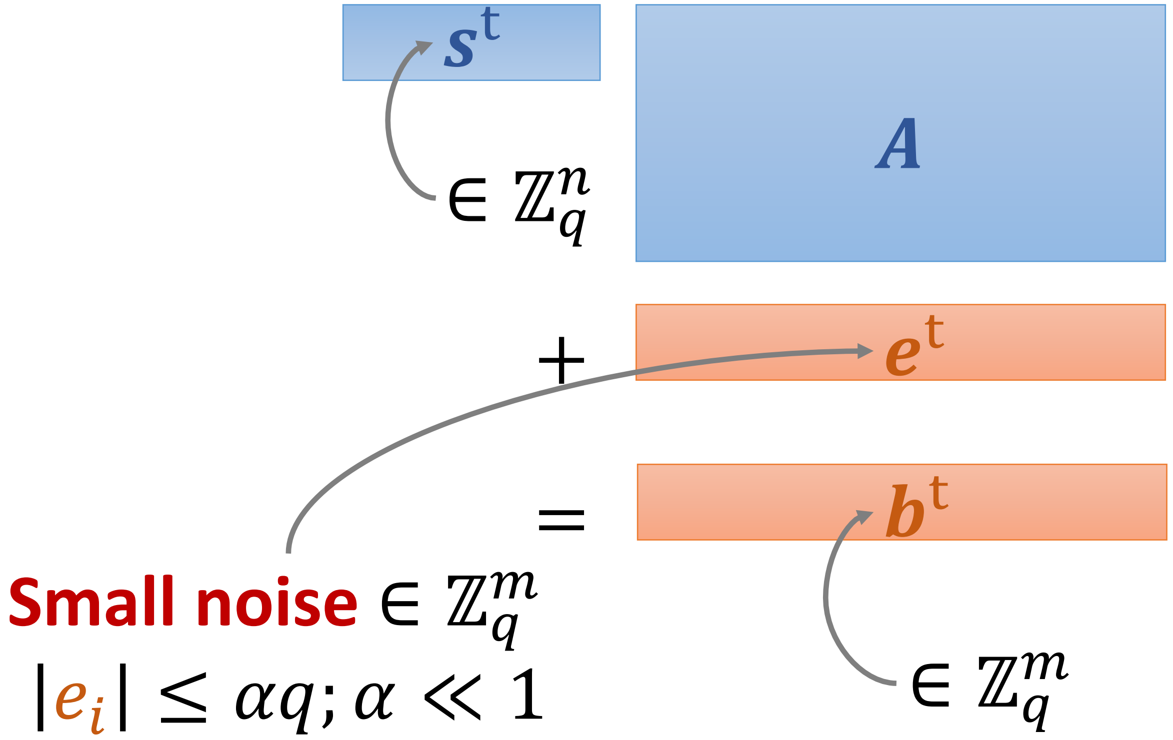
  - This maps **uniform** $(\boldsymbol{a}, b)$ to **uniform** $(\boldsymbol{a}', b')$, and thus works for **decision** LWE too

LWE vs SIS

- SIS has **many** valid solutions, whereas LWE only has **one**
- $\mathbf{LWE} \leq \mathbf{SIS}$
  - 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 $\boldsymbol{b_i} \in \mathbb{Z}_q$ 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
- **<u>Search LWE:</u>** Find secret vector of **polynomials** $\boldsymbol{s}$ in R_q^k given

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

- 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
- **<u>Decision LWE:</u>** 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 $R = \mathcal{O}_K$

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

- Can we **dequantize** the worst-case/average-case reduction?
  - The **classical** GapSVP <= LWE reduction is of little use: for the relevant factors, 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?

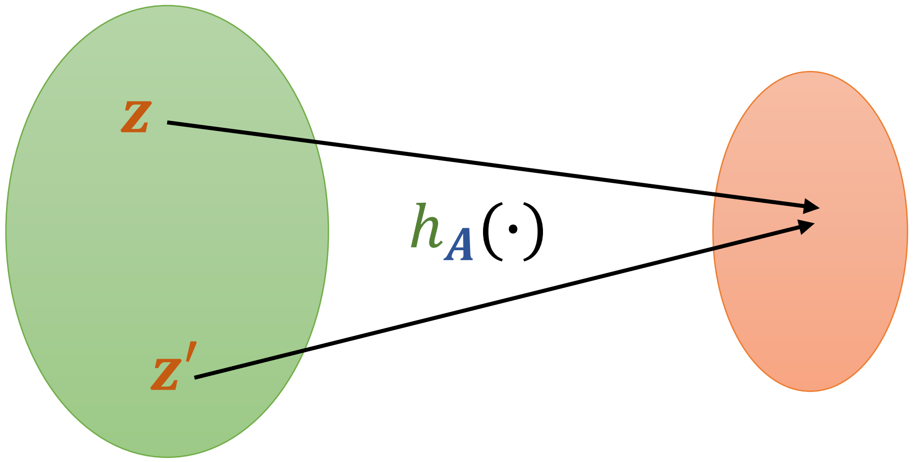
- **Provable** security
  - If scheme is not secure, one can solve hard mathematical problems
  - Not always happens in current implementations (e.g., RSA)
- **Worst-case** security
  - If scheme not secure, one can break **every** instance of lattice problems
  - Factoring and discrete log only guarantee **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 $\boldsymbol{A} = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_m)$, define $h_{\boldsymbol{A}}: \{0,1\}^m \rightarrow \mathbb{Z}_q^n$

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

  - 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 $\boldsymbol{a}_1 \cdot (z_1 - z_1') + \cdots + \boldsymbol{a}_m \cdot (z_m - z_m') = \boldsymbol{0}$, 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 H be a family of **universal hash functions** with domain D and image I. Then, for $x \leftarrow_\$ D$, $h \leftarrow_\$ H$, and $u \leftarrow_\$ I$: $\mathbb{SD}\Big(\big(h, h(x)\big); \big(h, u\big)\Big) \leq 1/2 \cdot \sqrt{|I|/|D|}$
- Note that the function $h_A(\vec{r}) = [A \times \vec{r}]_q$ is **universal**
  - As $\forall \vec{r}_1 \neq \vec{r}_2$: $\mathbb{P}_A[h_A(\vec{r}_1) = h_A(\vec{r}_2)] = \mathbb{P}_A\big[A \times (\vec{r}_1 - \vec{r}_2) = \vec{0}\big] = q^{-n}$
- Hence, for $\vec{r} \leftarrow_\$ \{0,1\}^m$, $A \leftarrow_\$ \mathbb{Z}_q^{n \times m}$, and $\vec{u} \leftarrow_\$ \mathbb{Z}_q^n$, whenever $m = 2 + n \log q + 2n$

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

SAPIENZA UNIVERSITÀ DI ROMA

NIST Standards

Falcon

Lattice Trapdoors

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

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

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

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

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

- Task: **Invert** $f_{\mathbf{A}}$
  - Find the **unique** $\mathbf{s}$ (or $\mathbf{e}$) such that $f_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{s}^{\mathrm{t}} \cdot \mathbf{A} + \mathbf{e}^{\mathrm{t}} \bmod q$
  - Given $\mathbf{u} = f_{\mathbf{A}}(\mathbf{x}') = \mathbf{A} \cdot \mathbf{x}' \bmod q$, **sample random** $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathbf{u})$ with probability proportional to $\exp(-\|\mathbf{x}\|^2/s^2)$
- How? Via a **strong trapdoor** for $\mathbf{A}$ (a **short basis** of $\mathcal{L}^{\perp}(\mathbf{A})$)
  - 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 $f_{\boldsymbol{G}}^{-1}$ [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:** $s_1(\boldsymbol{R}) \approx (\sqrt{\text{rows}} + \sqrt{\text{cols}}) \cdot r$ 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 $\|\tilde{\boldsymbol{S}}_{\boldsymbol{A}}\| \leq (s_1(\boldsymbol{R}) + 1) \cdot \|\tilde{\boldsymbol{S}}\|$

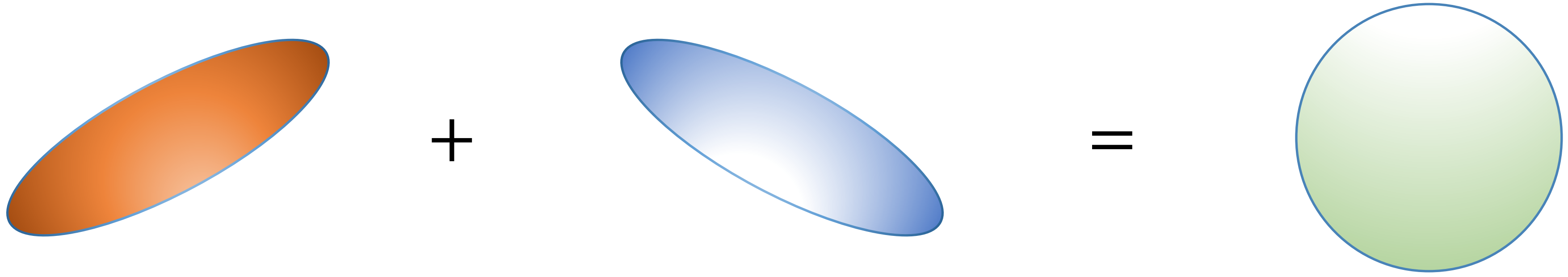
Step 3: Reduce $f_{\boldsymbol{A}}^{-1}$ to $f_{\boldsymbol{G}}^{-1}$

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

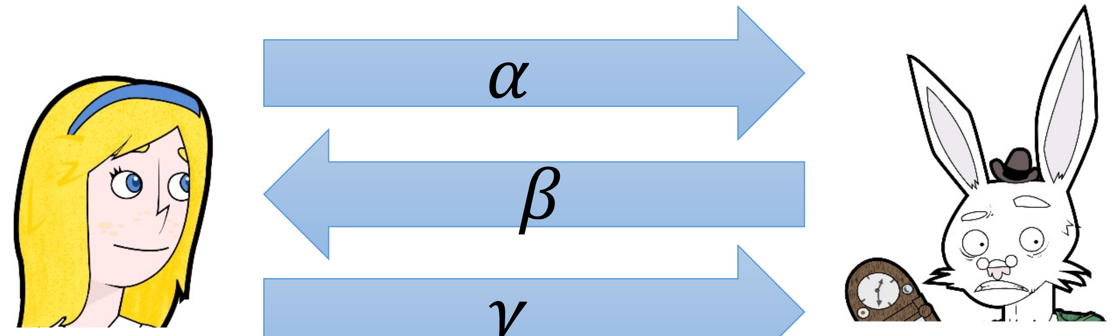
$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{A} \cdot \boldsymbol{p} + \boldsymbol{A} \cdot \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{I} \end{bmatrix} \cdot \boldsymbol{z} = \boldsymbol{A} \cdot \boldsymbol{p} + \boldsymbol{G} \cdot \boldsymbol{z} = \boldsymbol{u}$$

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

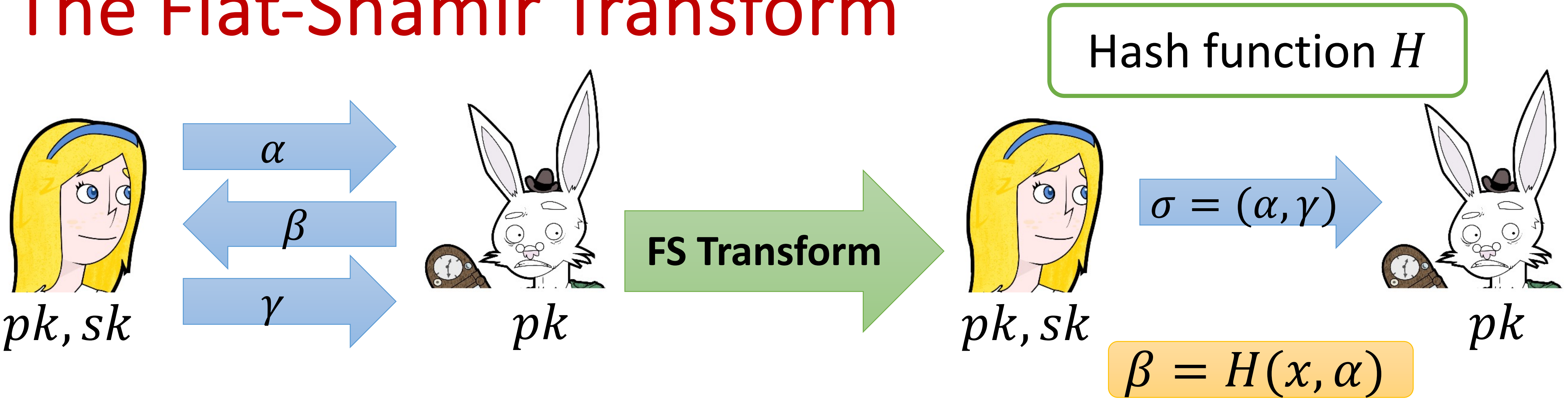


pk, sk $\qquad$ pk

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

- **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



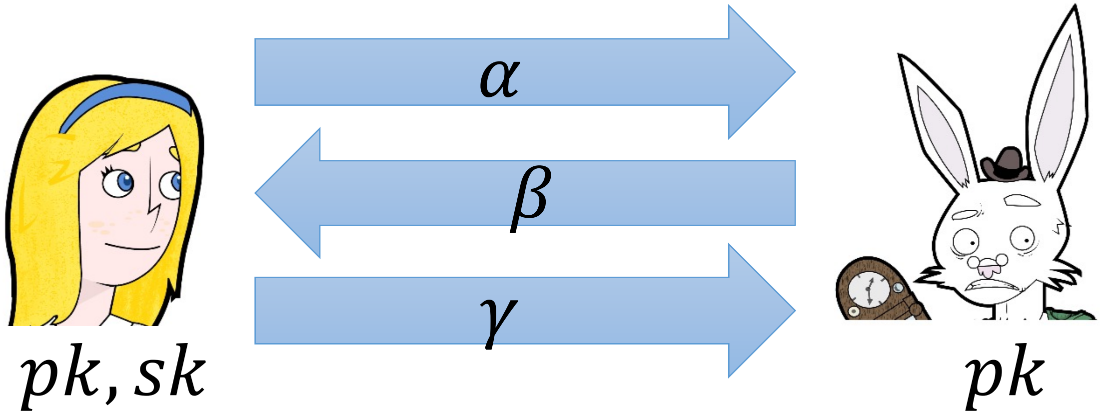
- Given a **canonical** ID scheme, we can derive a **signature scheme** as follows:
  - Alice obtains $\sigma = (\alpha, \gamma)$ from the **prover**, using the **secret key** sk and choosing $\beta = H(x, \alpha)$
  - Bob checks that (α, β, γ) is a **valid transcript**, with $\beta = H(x, \alpha)$

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:

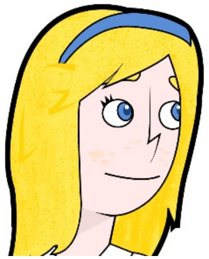
$$\big(pk, sk, \mathcal{S}(pk)\big) \equiv (pk, sk, \langle \mathcal{P}(pk, sk), \mathcal{V}(pk)\rangle)$$

Canonical ID Scheme from Discrete Log

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

$x = g^w$

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

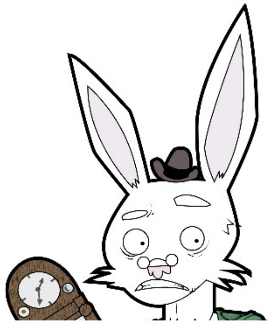


x, w

$\alpha = g^a$ →

← β

γ →



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

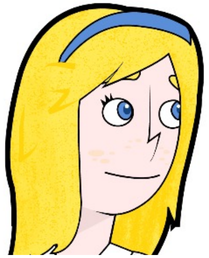
Check $g^\gamma = x^\beta \cdot \alpha$

- **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

$\text{params} = q$

$u \leftarrow_\$ \mathbb{Z}_q^m$
$\gamma = \beta \cdot s + u$

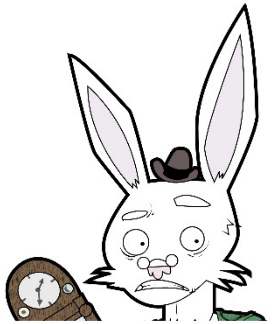


$(A, t), s$

$\alpha = A \cdot u$

β

γ



$A \cdot s = t$

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

Check $A \cdot \gamma = \beta \cdot t + \alpha$

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

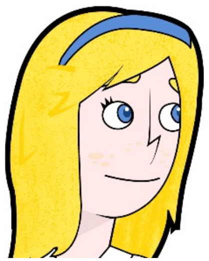
Many Problems…

- 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)

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

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

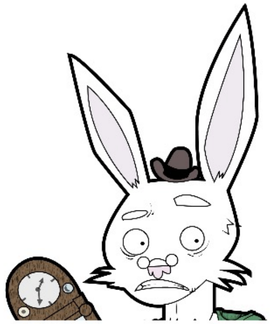


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

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

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

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



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

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 $\boldsymbol{s} = (\boldsymbol{\gamma}_j - \boldsymbol{\gamma}'_j) \cdot (\beta_j - \beta'_j)^{-1}$
  - 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 $\beta = 1$

$$
\begin{array}{cccccccccc|l}
0 & ? & 1 & ? & 1 & 0 & ? & 0 & ? & ? & \beta \cdot \boldsymbol{s} = \boldsymbol{s} \text{ has coefficients in } \{0,1\} \\
 & & & & + & & & & & & \\
0 & ? & 1 & ? & 1 & 0 & ? & 0 & ? & ? & \boldsymbol{u} \text{ has coefficients in } \{0,1\} \\
 & & & & = & & & & & & \\
0 & 1 & 2 & 1 & 2 & 0 & ? & 0 & ? & ? & \gamma \text{ coefficients}
\end{array}
$$

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 $a \le i < b$, we have $\mathbb{P}[s + u = i] = 1/b$
  - Moreover, there are $b - a$ such j'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 $(1 - 1/b)^{mk} \approx 1/e$

SAPIENZA UNIVERSITÀ DI ROMA

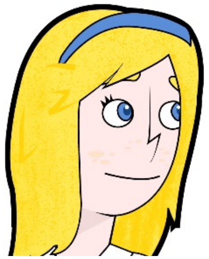
Modified Protocol (Take 2)

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

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

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

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

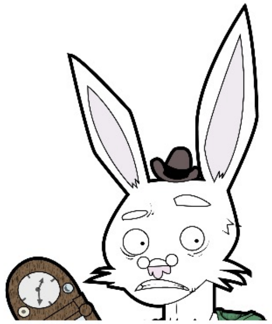


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

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

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

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



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

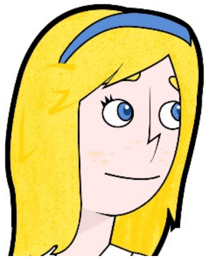
Check $\boldsymbol{A} \cdot \boldsymbol{\gamma}_j = \beta_j \cdot \boldsymbol{t} + \boldsymbol{\alpha}_j$

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

Modified Protocol (Take 2)

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

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

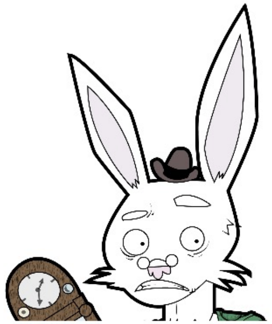


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

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

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

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



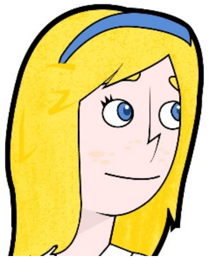
$\beta_j \leftarrow_\$ \{0,1\}$

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

- **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)

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

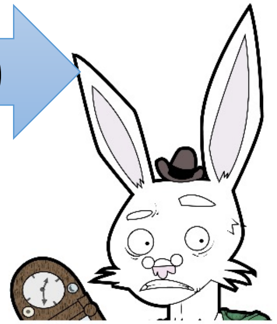


$(\boldsymbol{A}, \boldsymbol{t}), \boldsymbol{s}$

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

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

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



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

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

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 $\{0, \dots, mk\}$)
- 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

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

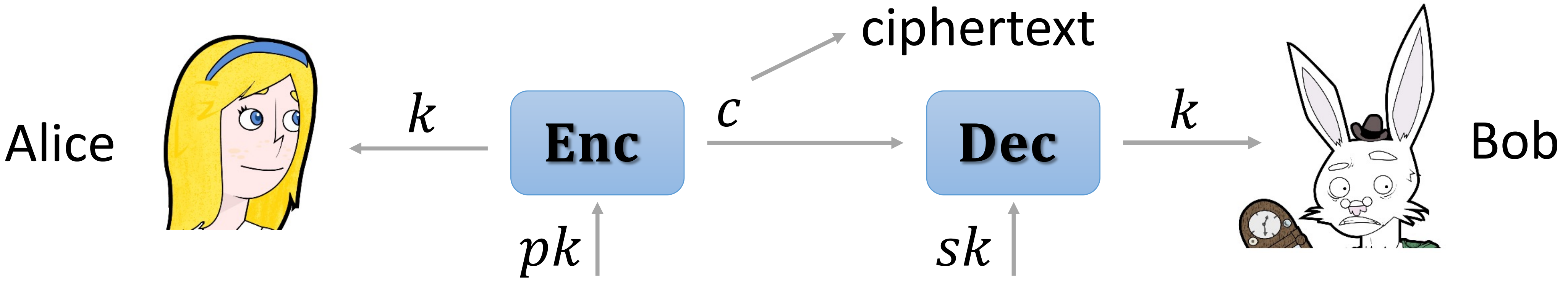
- Public key
  - Primal: pk is **pseudorandom** with **unique** sk
  - Dual: pk is **statistically random** with **many possible** sk
- Ciphertext
  - Primal: A fresh LWE sample with **many possible** coins
  - Dual: Multiple LWE samples with **unique** coins
- Security
  - Primal: Encrypting with **uniform** 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** $(\boldsymbol{c}_0, c_1)$

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

$\xleftarrow{pk, c^*}$

$\xrightarrow{m, c}$

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

$\xrightarrow{m'}$

Challenger

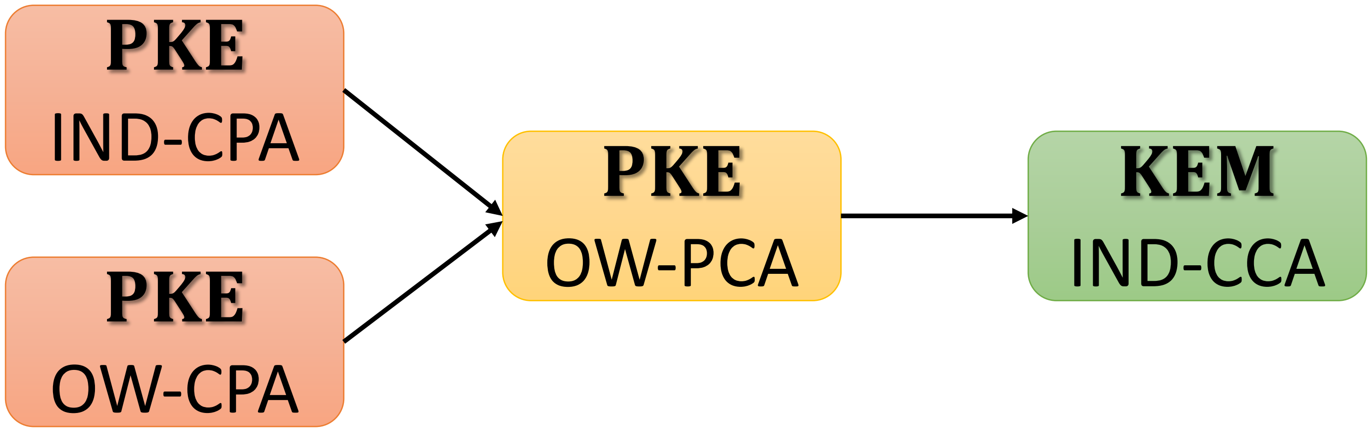
pk, sk

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

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

- **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

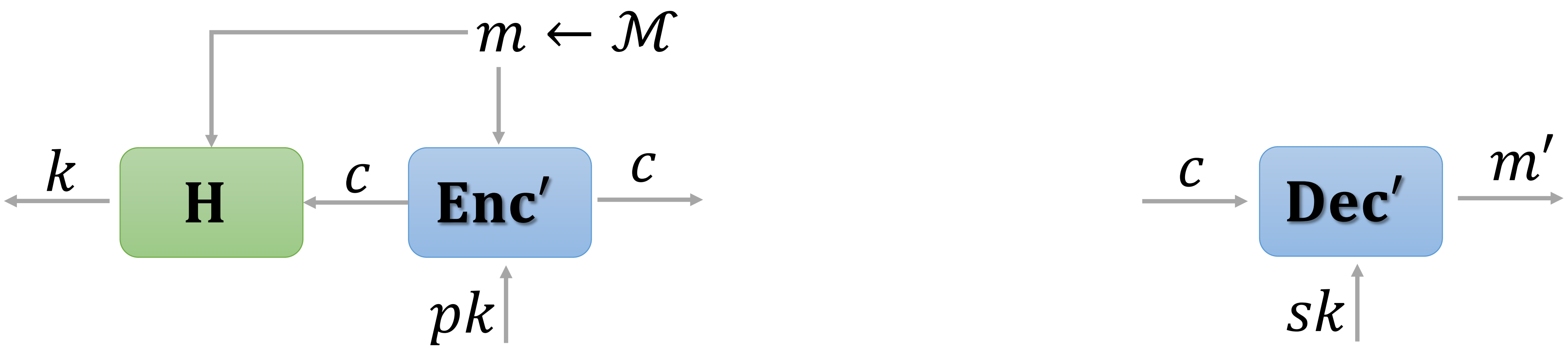


- We can view FO as the **concatenation** of **two transforms** $\mathbf{U} \circ \mathbf{T}$
  - The first transformation takes care of **derandomization** and allows to go from **IND-CPA** to **OW-PCA**
  - The second transformation takes care of **hashing** and allows to go from **OW-PCA** to **IND-CCA**

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**

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



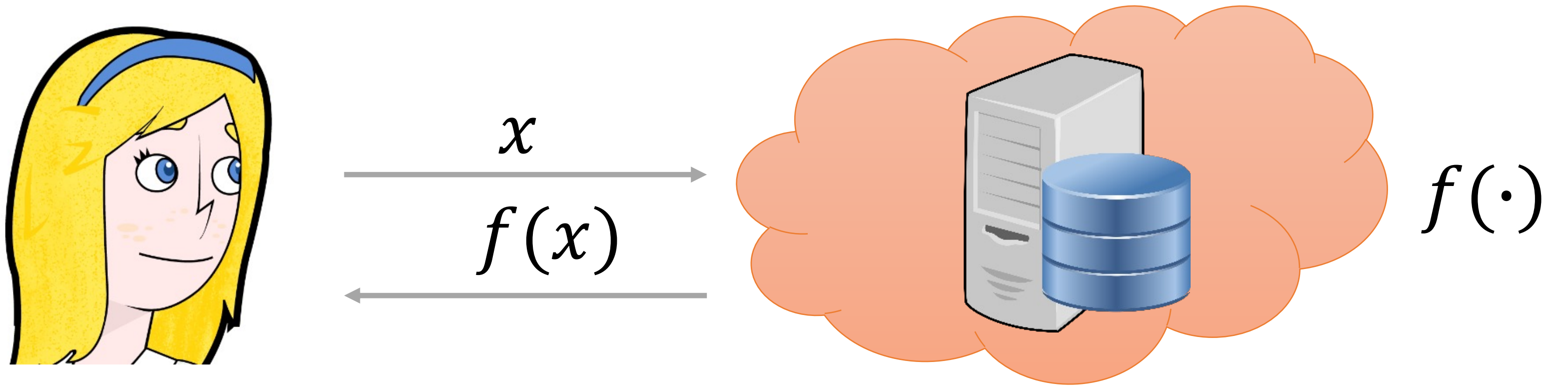
- Encapsulation outputs $k = \mathbf{H}(c, m)$ and c
- Decapsulation obtains $m' = \mathbf{Dec}(sk, c)$ and outputs m'
  - Here, m' could be $\perp$ (**explicit rejection**)
- **Theorem [HKK17]:** Assuming $(\mathbf{Enc'}, \mathbf{Dec'})$ is **OW-PCA**, $(\mathbf{Encaps}, \mathbf{Decaps})$ is **IND-CCA**

Advanced Cryptographic Applications

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"

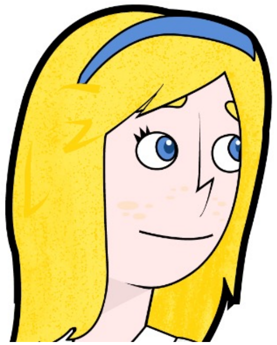
Motivation: Outsourcing of Computation



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

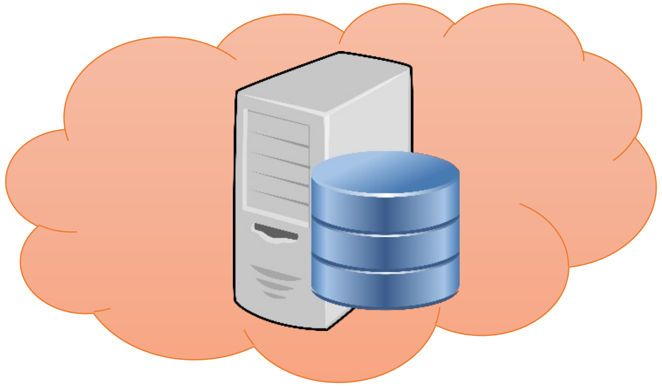
Outsourcing of Computation - Privately

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



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

$\leftarrow$ y

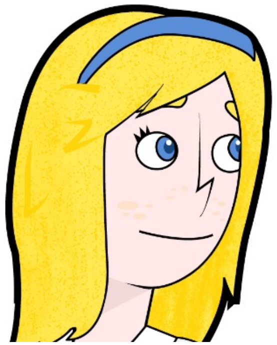


$f(\cdot)$

<u>**Wish:**</u> Homomorphic **evaluation** function:

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

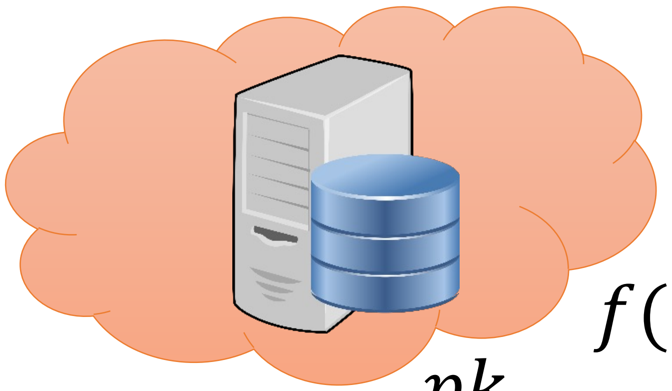
Fully-Homomorphic Encryption (FHE)



pk, sk

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

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



pk

$f(\cdot)$

Correctness:

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

Privacy:

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

FHE = Correctness $\forall$ efficient f = Correctness for universal set

Levelled FHE: **Bounded** depth f

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

A Paradox (and its Resolution)

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

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

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

AH! So $x_1 = 0$

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

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

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

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 $a \bmod q$

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

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

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

$$C_{\text{mult}} = C_1 \times C_2:$$
$$\vec{s} \times (C_1 \times C_2) = (x_1 \cdot \vec{s} + \vec{e}_1) \times C_2$$
$$= x_1 \cdot (x_2 \cdot \vec{s} + \vec{e}_2) + \vec{e}_1 \times C_2$$
$$= x_1 \cdot x_2 \cdot \vec{s} + (x_1 \cdot \vec{e}_2 + \vec{e}_1 \times C_2)$$

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:

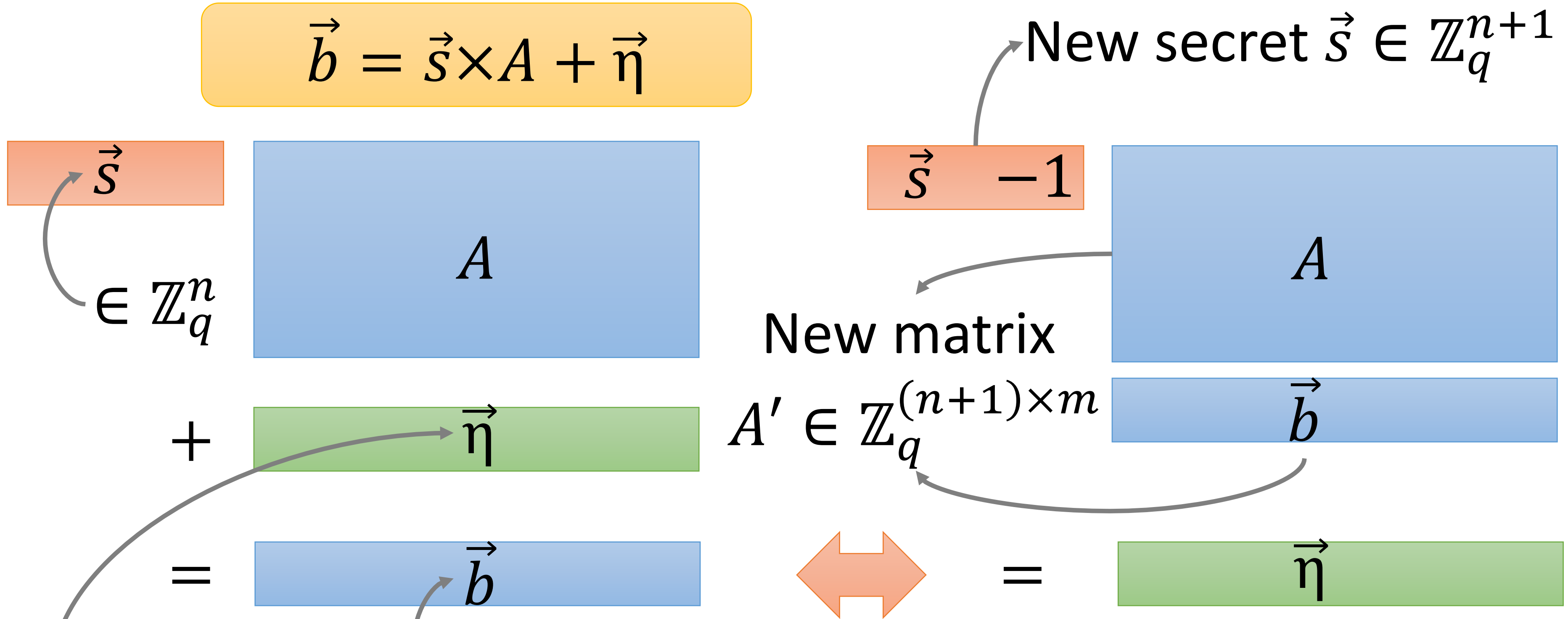
$$C = G \times G^{-1}(C) = \begin{bmatrix} 2^{N-1} & \dots & 2 & 1 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 & 2^{N-1} & \dots & 2 & 1 \end{bmatrix} \times \text{bits}(C)$$

$$k \cdot N = k\lceil \log q \rceil$$

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

LWE – Rearranging Notation

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



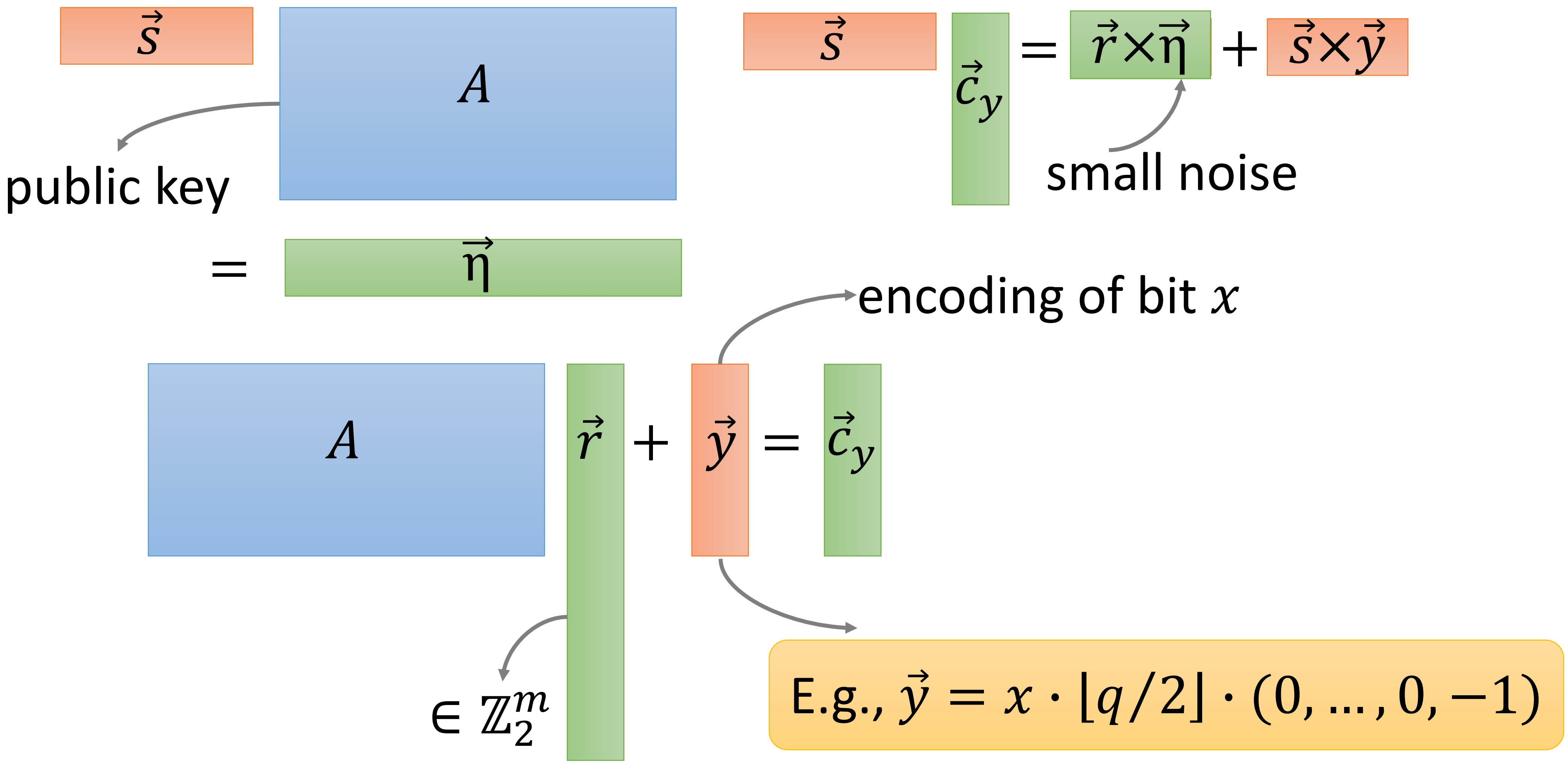
Small noise $\in \mathbb{Z}_q^m$

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

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

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

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

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

The GSW Scheme – Homomorphism

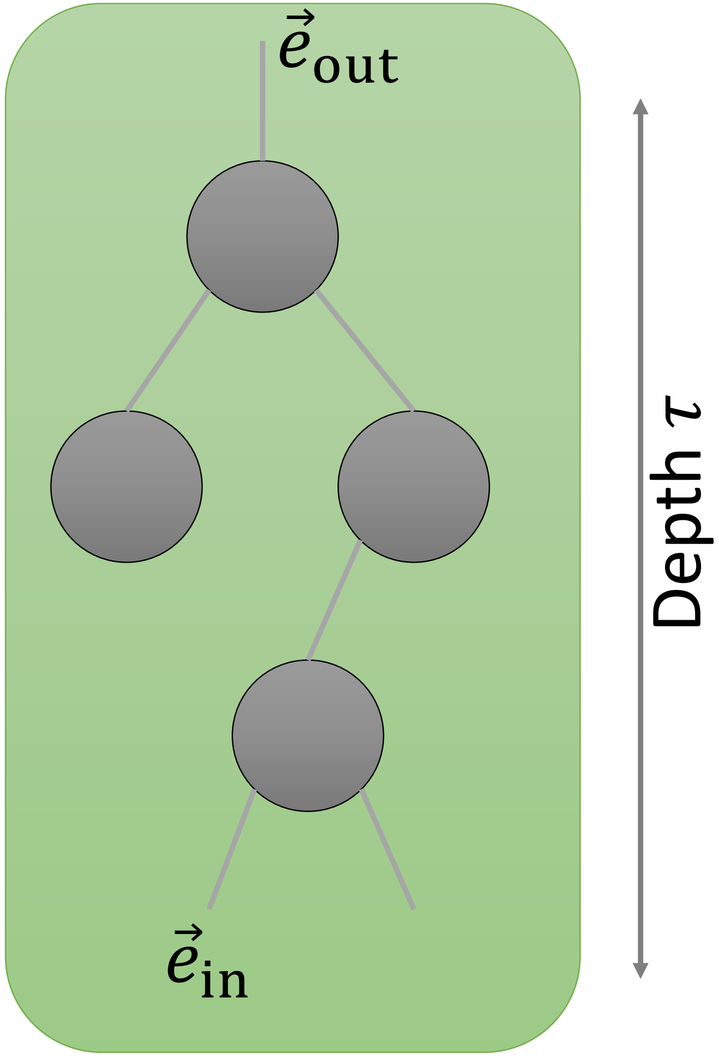
Invariant: $\vec{s} \times C = \vec{e} + x \cdot \vec{s} \times G$

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

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

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

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

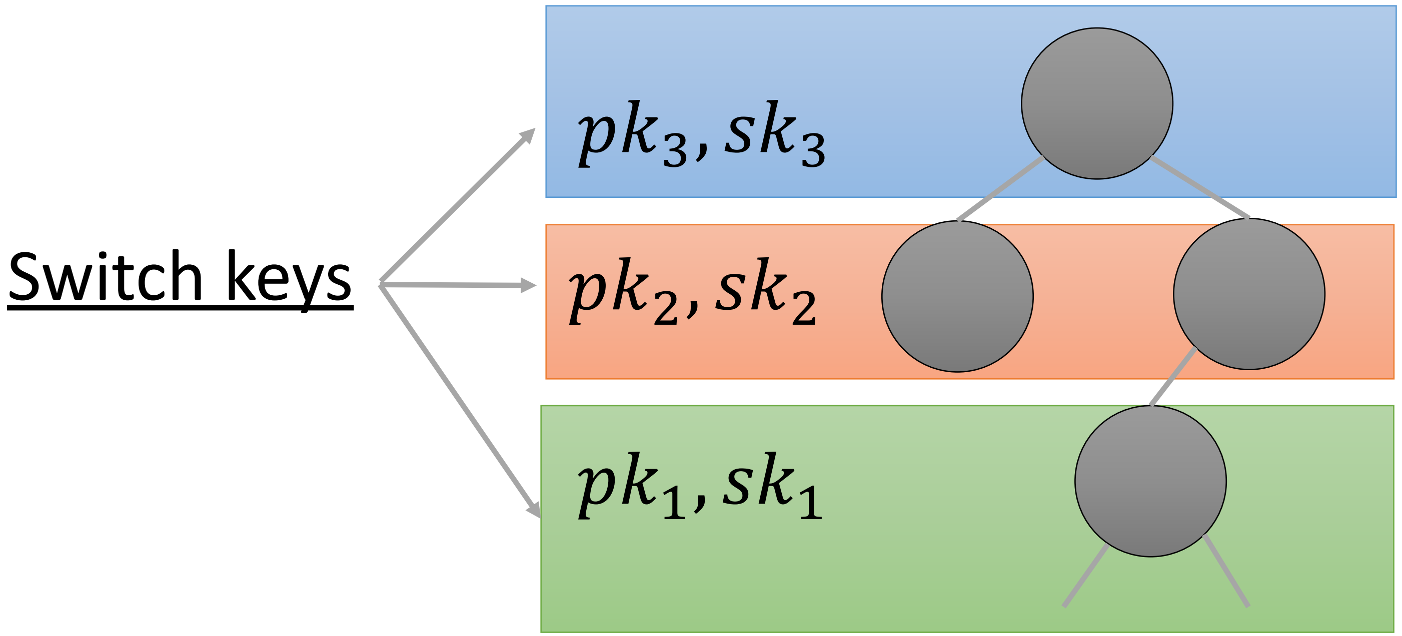
- Similar as in the proof of Regev PKE
- Using LWE we move to a **mental experiment** with $A \leftarrow_{\$} \mathbb{Z}_q^{n\times m}$
- Hence, by the **leftover hash lemma**, with $m = \Theta(n \log q)$, the statistical distance between $(A, A\times\vec{r})$ and uniform is negligible
  - By a **hybrid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

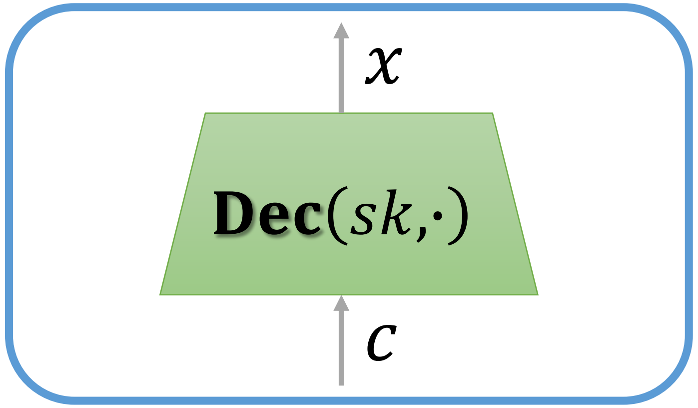
- **Correctness** requires $n \cdot m \cdot (N+1)^{\tau+1} < q/4$
- **Security** requires $m = \Theta(n \log q)$, e.g. $m \geq 1 + 2n(2 + \log q)$
- **Hardness** of LWE requires $q \leq 2^{n^\epsilon}$ for $\epsilon < 1$
  - Substituting we get $q > (2n \log q)^{\tau+3}$
  - And thus $n^\epsilon > (\tau+3)(\log n + \log\log q + 1)$ 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

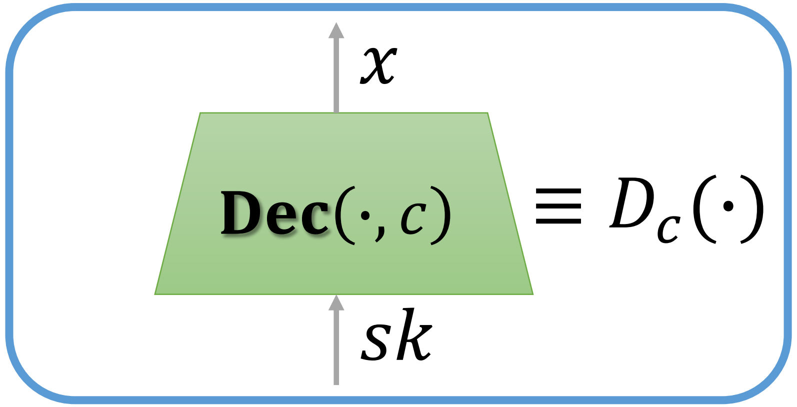
- The only way to increase the homomorphic capacity of GSW is to pick **larger parameters**
- This dependence can be **broken** using a trick by Gentry
- Main idea: Do a few operations, then **switch keys**



How to Switch Keys



<u>Decryption circuit</u>



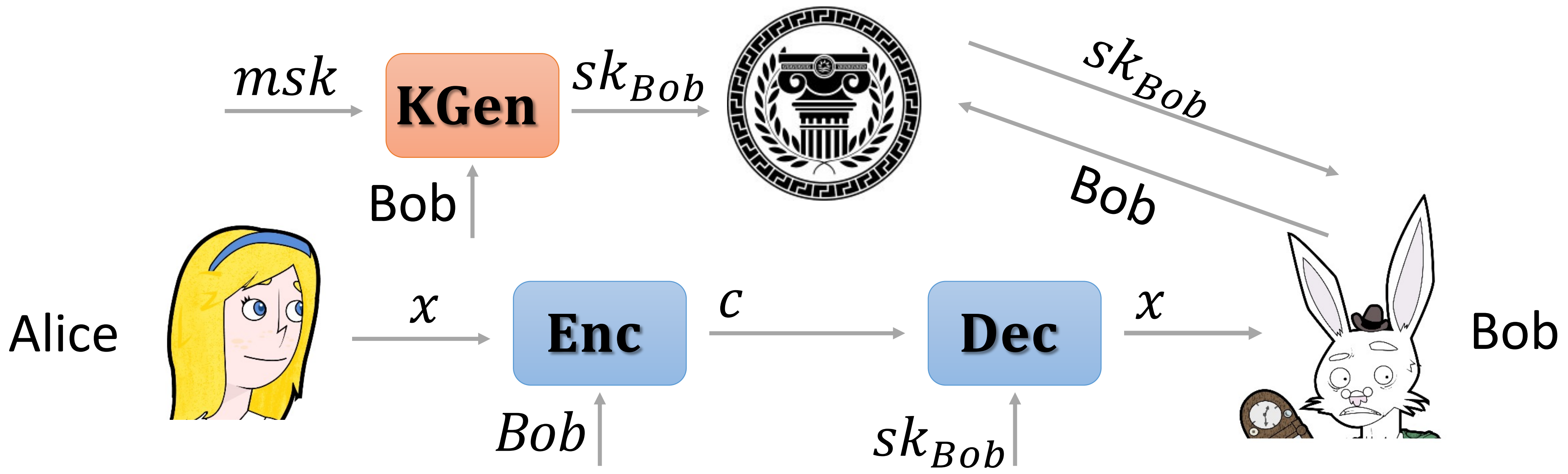
<u>Dual view</u>

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

Circular Security

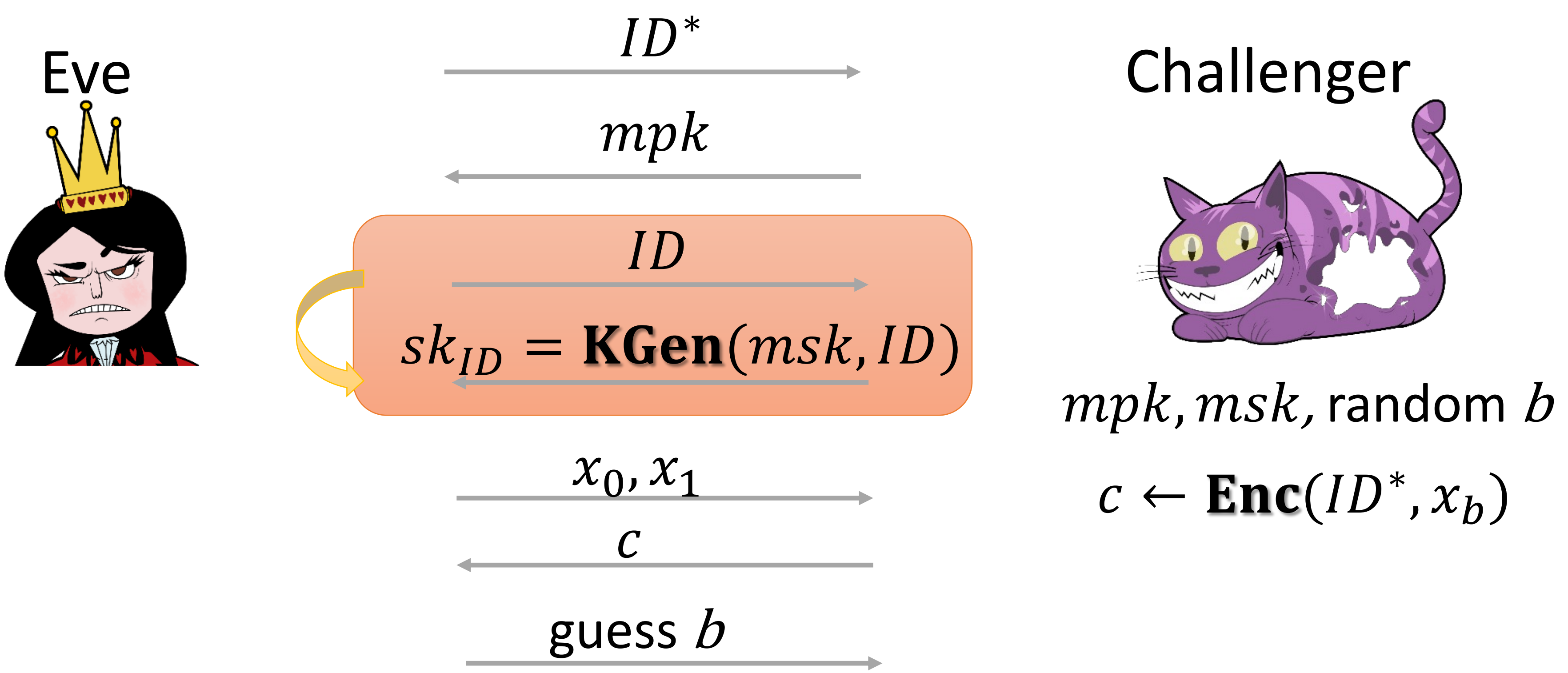
- The above scheme is **compact**, but **not fully homomorphic**, as we need a pair of keys **for each level** in the circuit
- A natural idea is to use a **single pair** (pk, sk) and include in pk' a ciphertext $\vec{c}^* \leftarrow_\$ \mathbf{Enc}(pk, sk)$
  - Correctness still holds for this variant, 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

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



- 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

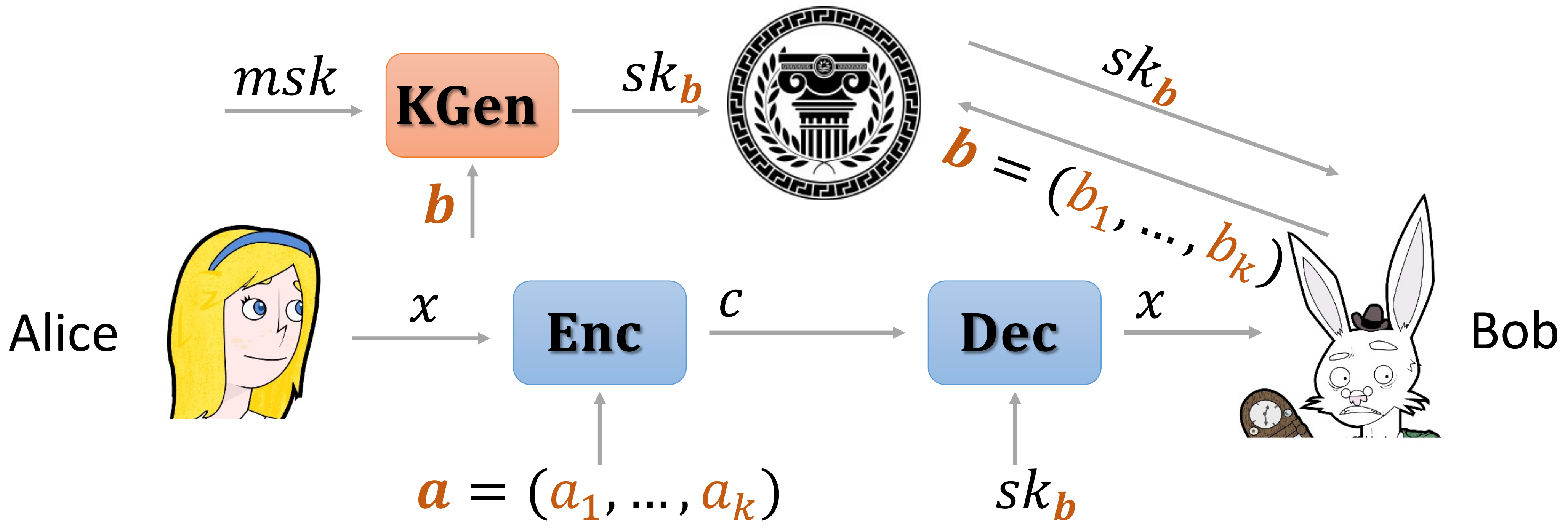
A More Efficient Construction [ABB10]

- **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]

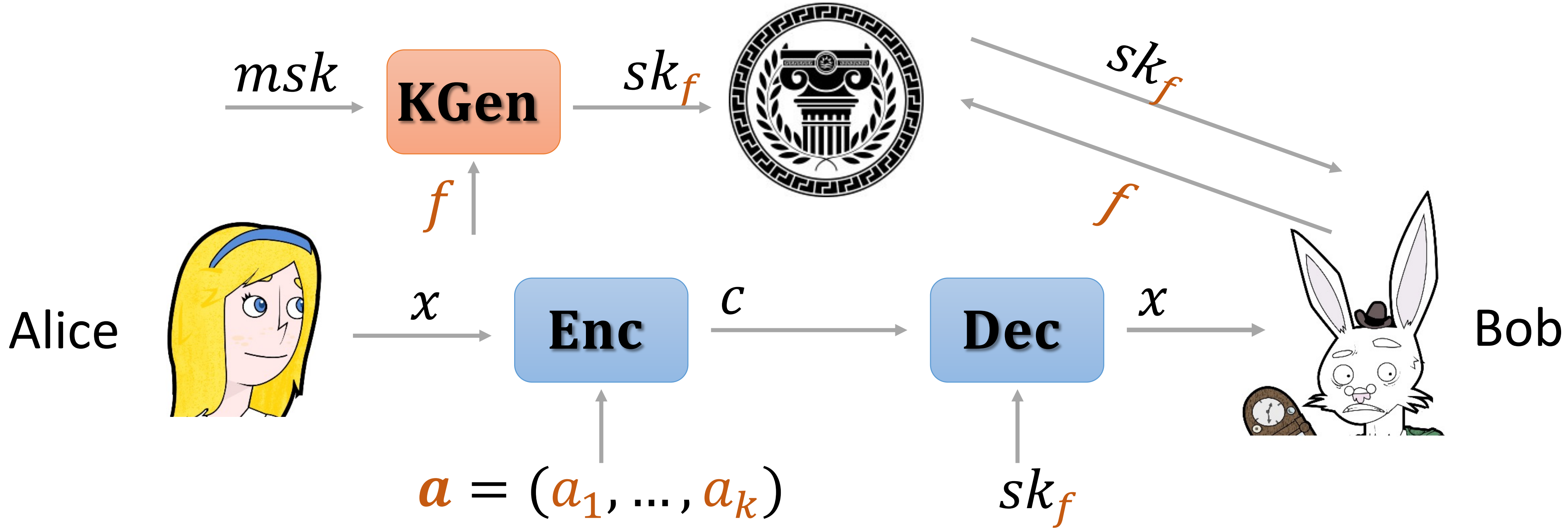


- Decryption reveals x **if and only if** $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = 0$
  - Here, we can also be interested in **attributes privacy**
- Can be used to obtain **predicate encryption** for polynomial evaluation, CNFs/DNFs of bounded degree, and **fuzzy** IBE

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 $c' - \boldsymbol{c}_0^{\mathrm{t}} \cdot \boldsymbol{s}_{12} - \boldsymbol{c}_{12}^{\mathrm{t}} \cdot \boldsymbol{s}_{12} \approx x \cdot q/2$