Efficient Verifiable Mixnets from Lattices, Revisited

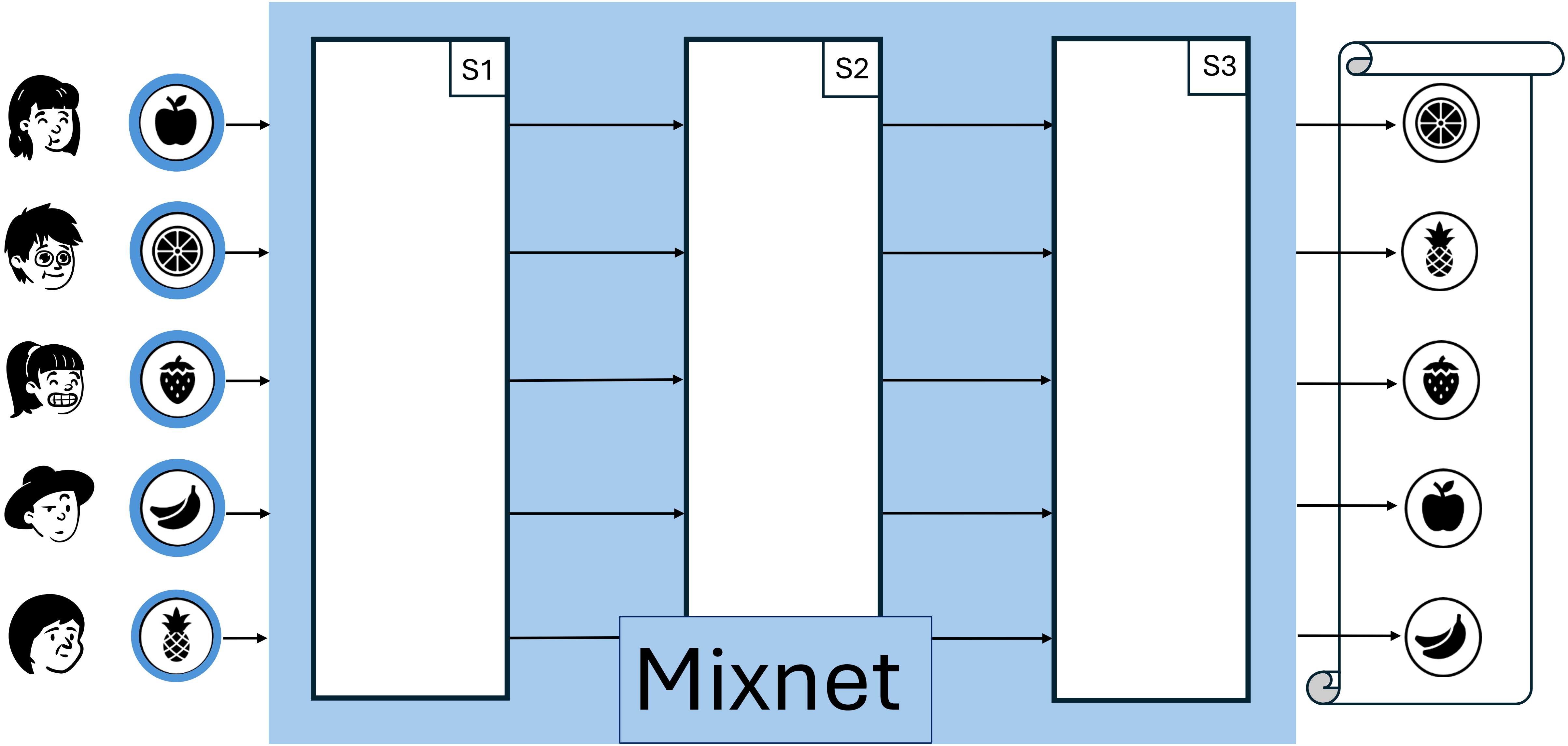
Jonathan Bootle[1] **Vadim Lyubashevsky**[1] **Antonio Merino–Gallardo**[1,2,3]

[1] IBM Research Europe – Zurich

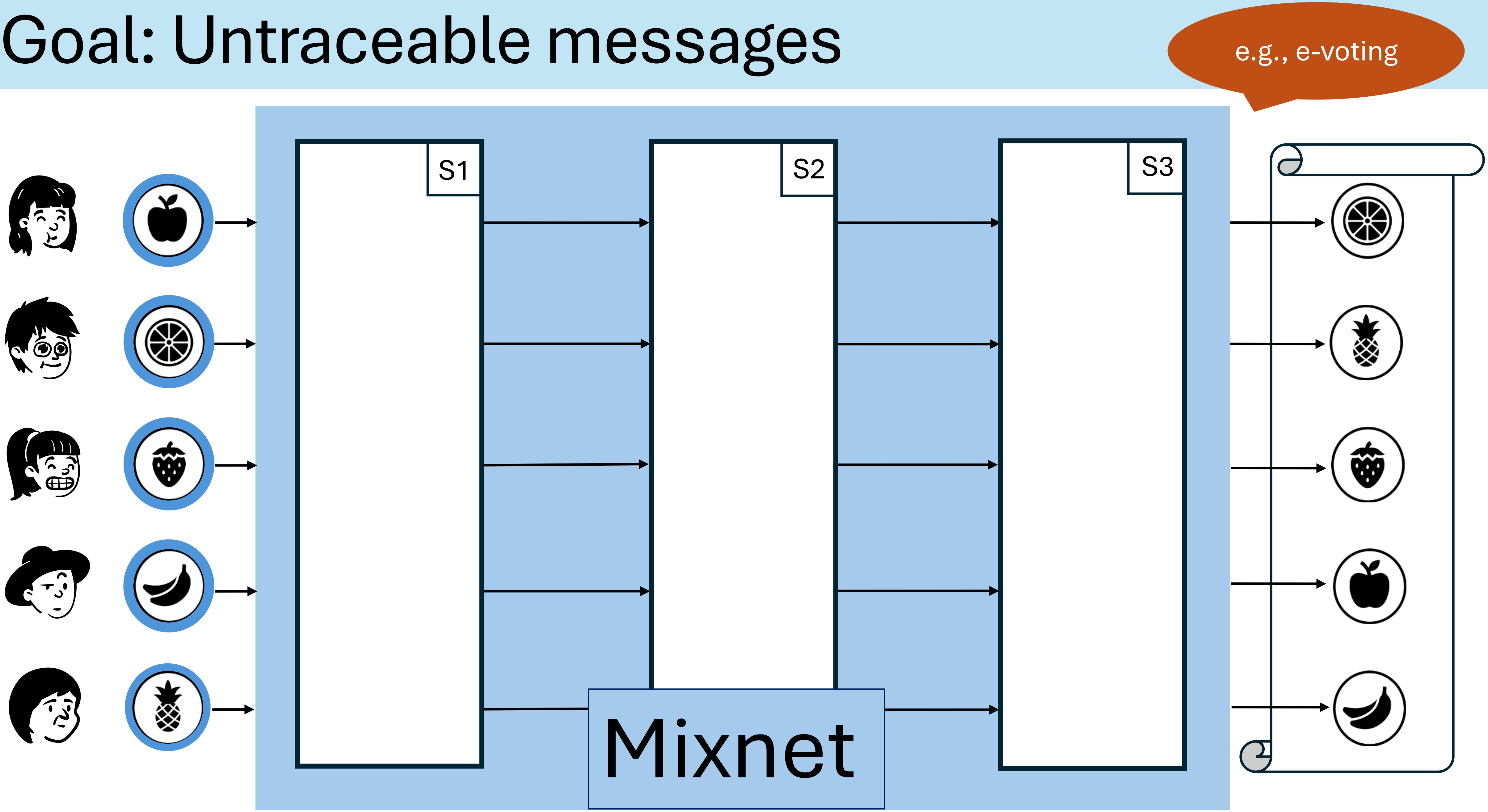
[2] University of Potsdam, Germany

[3] Work done partly while at ETH Zurich

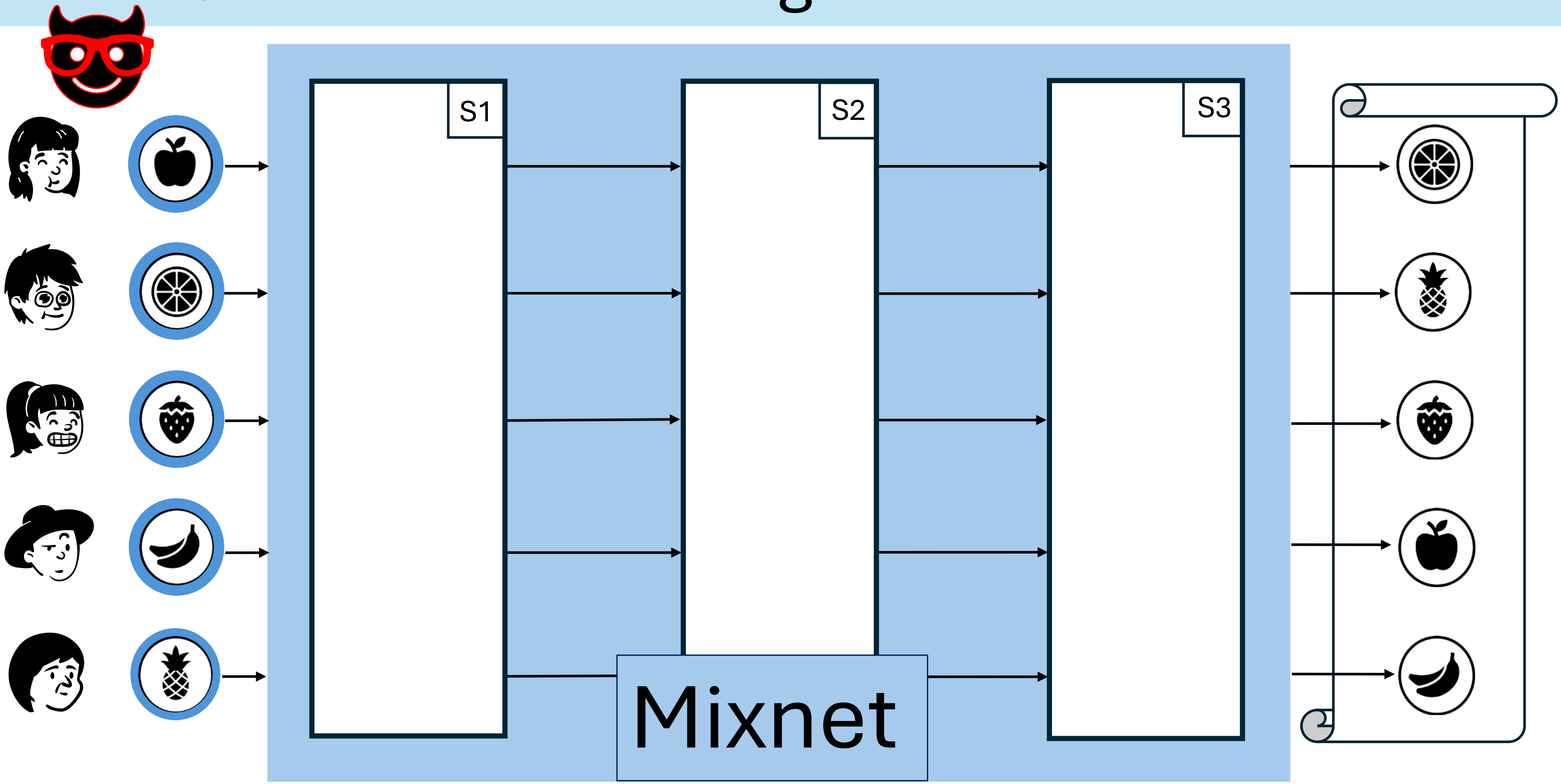
Goal: Untraceable messages



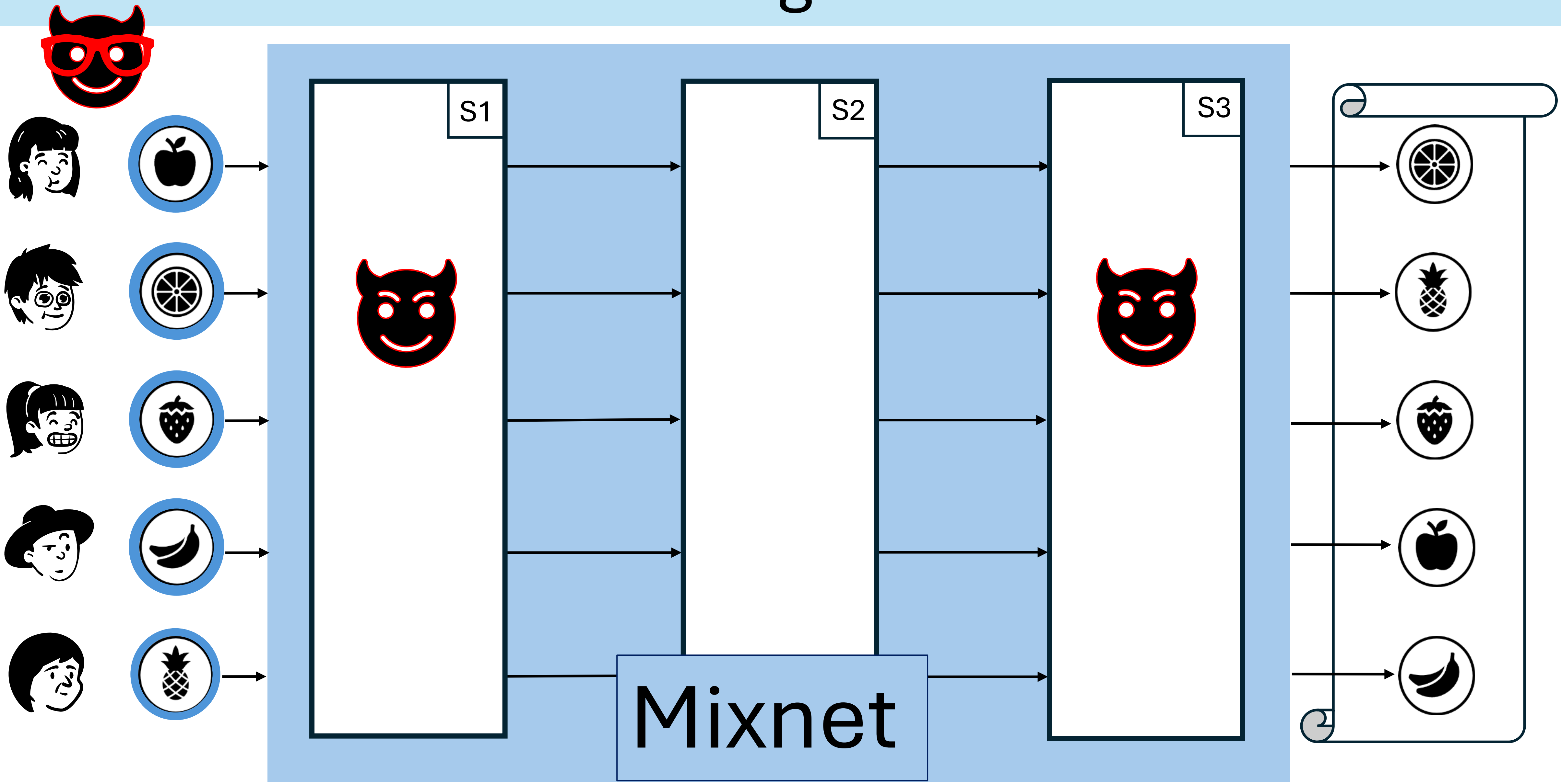
Goal: Untraceable messages



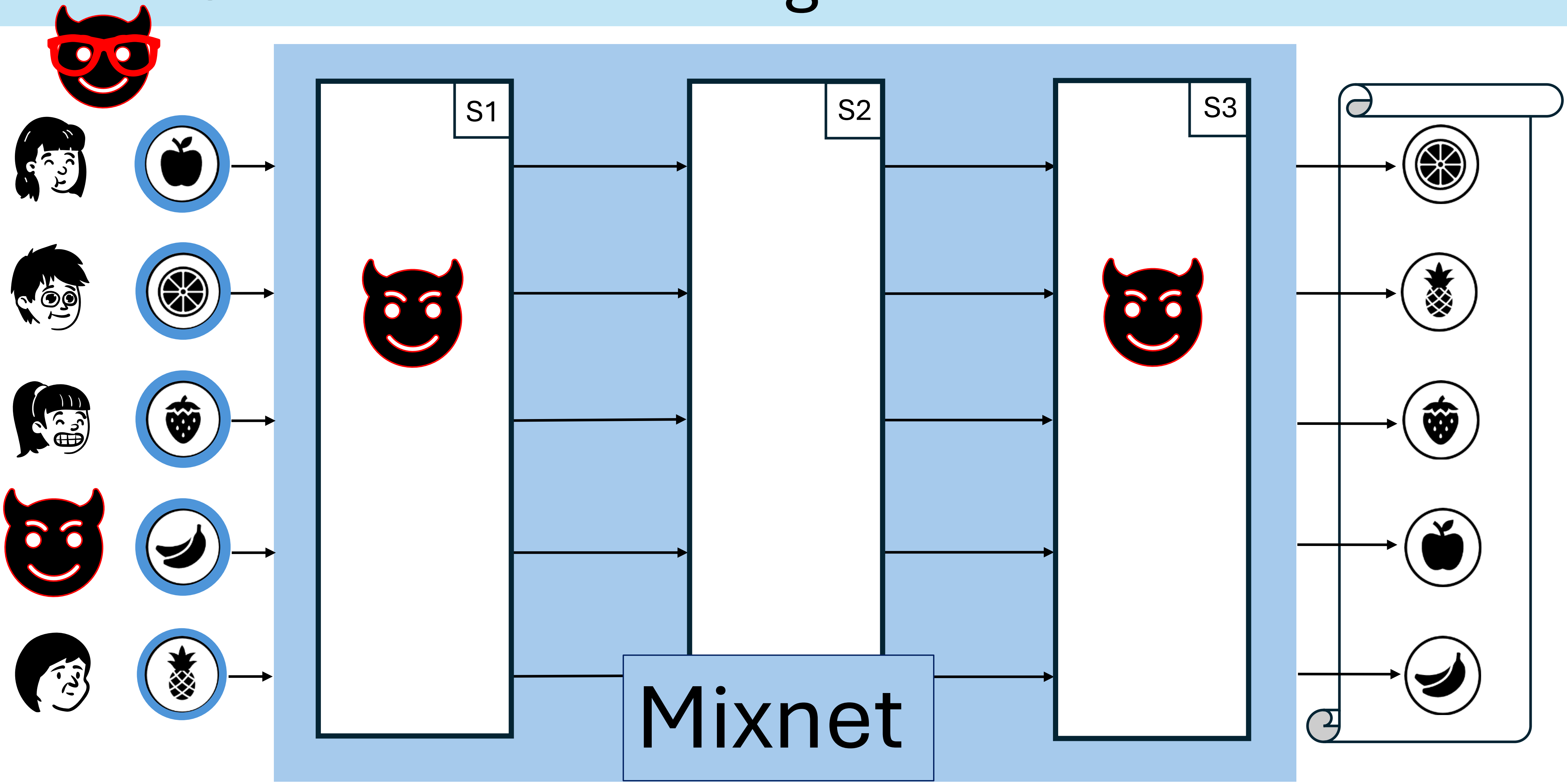
Goal: Untraceable messages



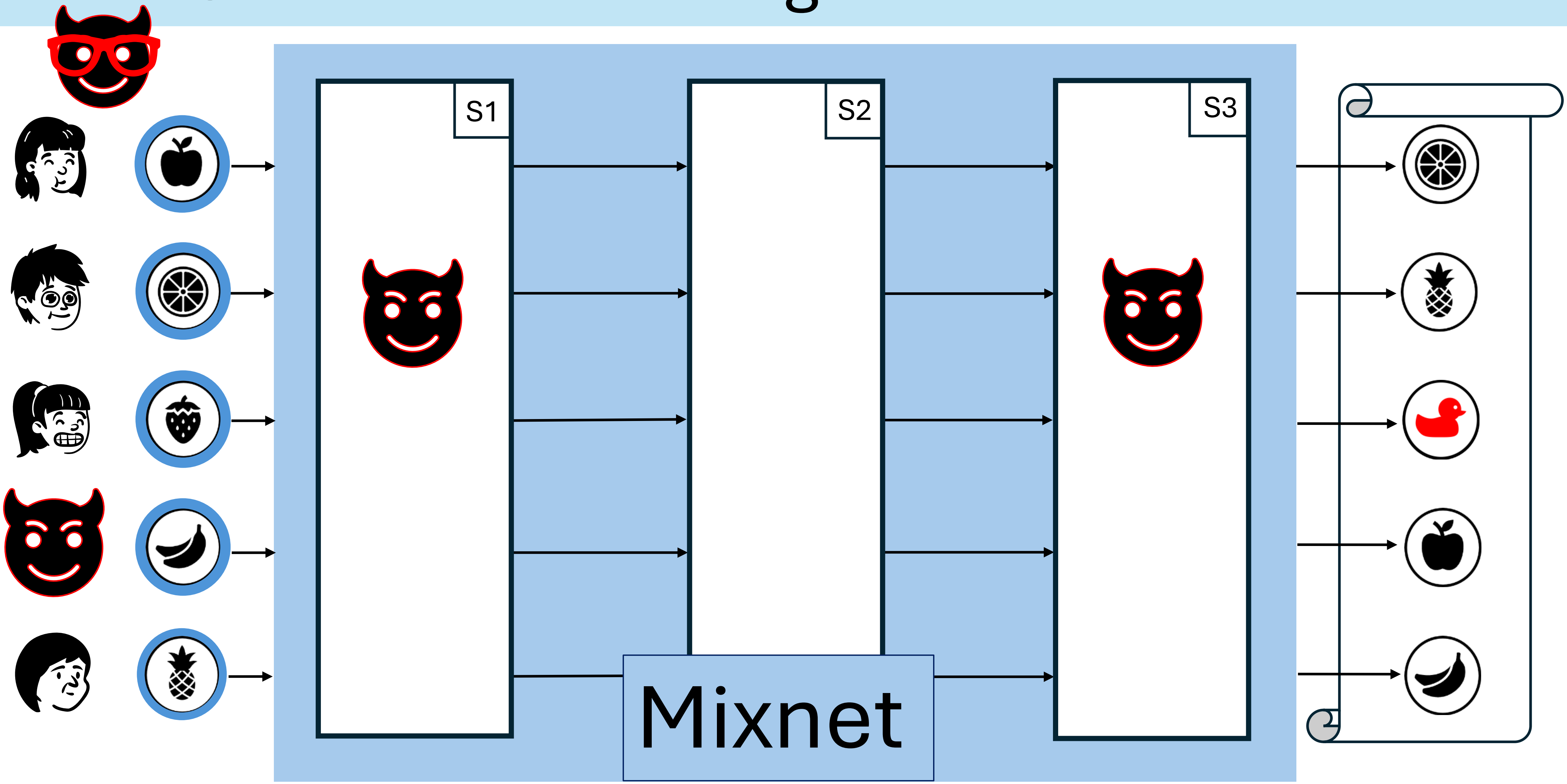
Goal: Untraceable messages



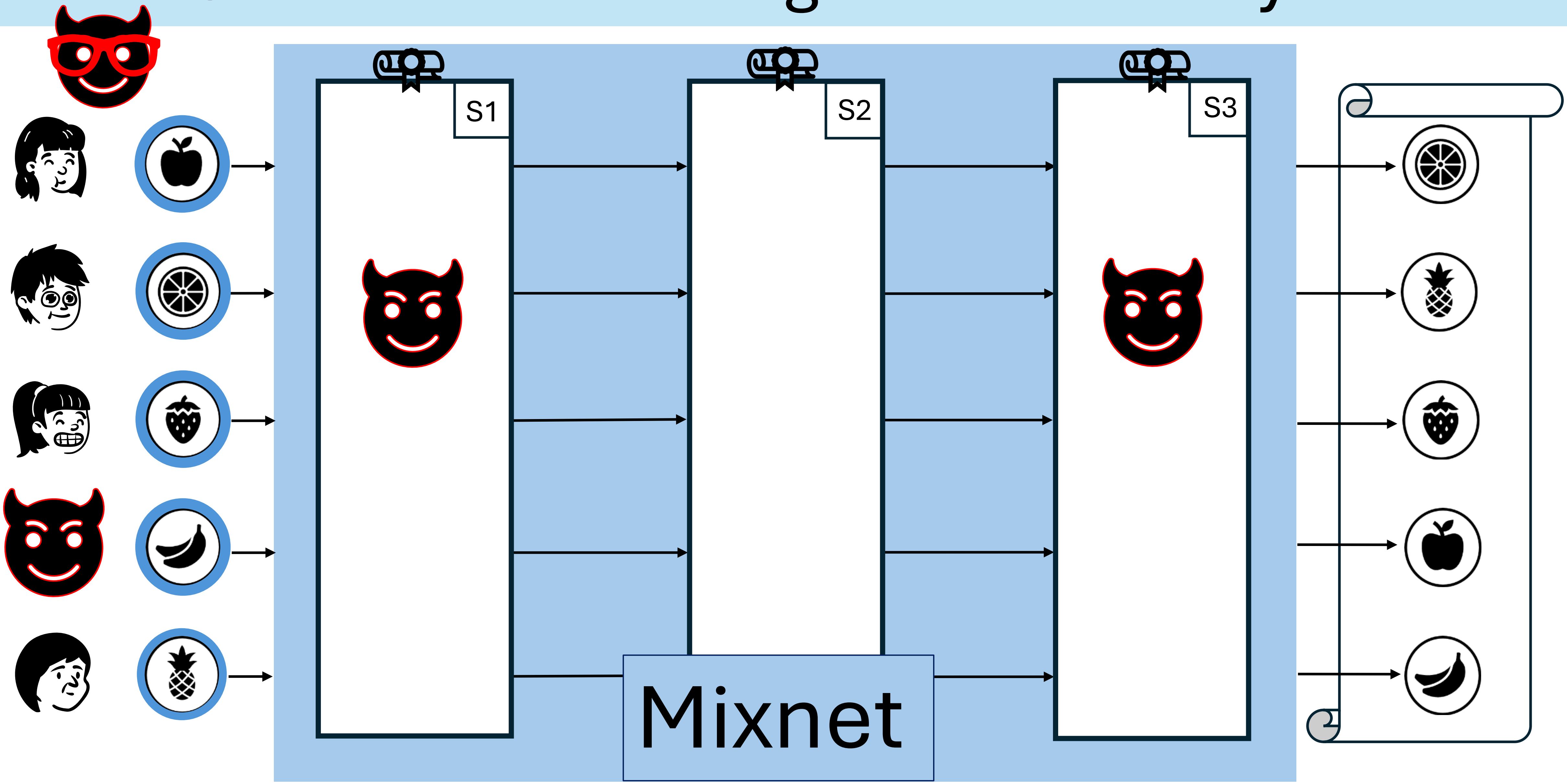
Goal: Untraceable messages



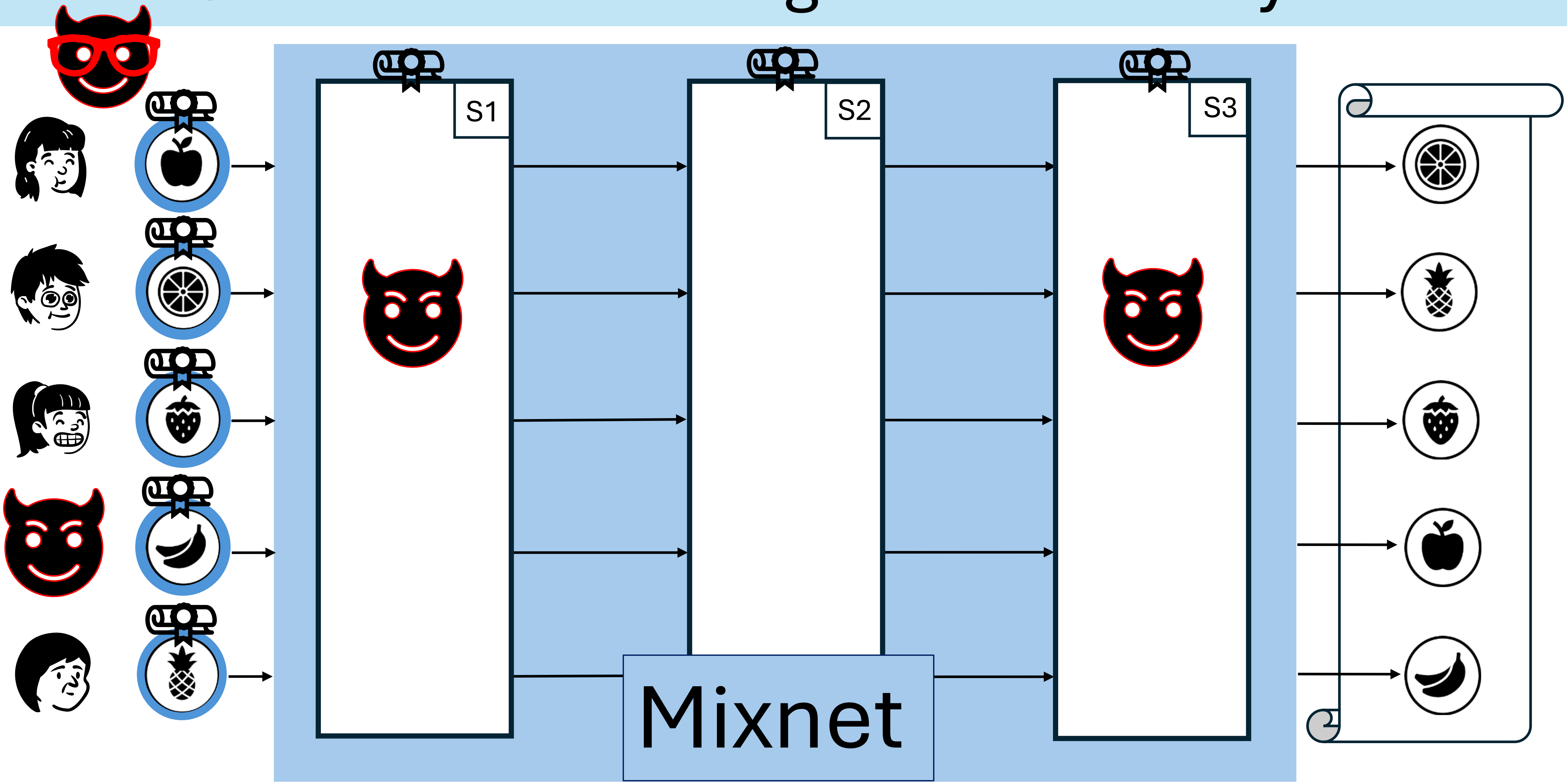
Goal: Untraceable messages



Goal: Untraceable messages + verifiability

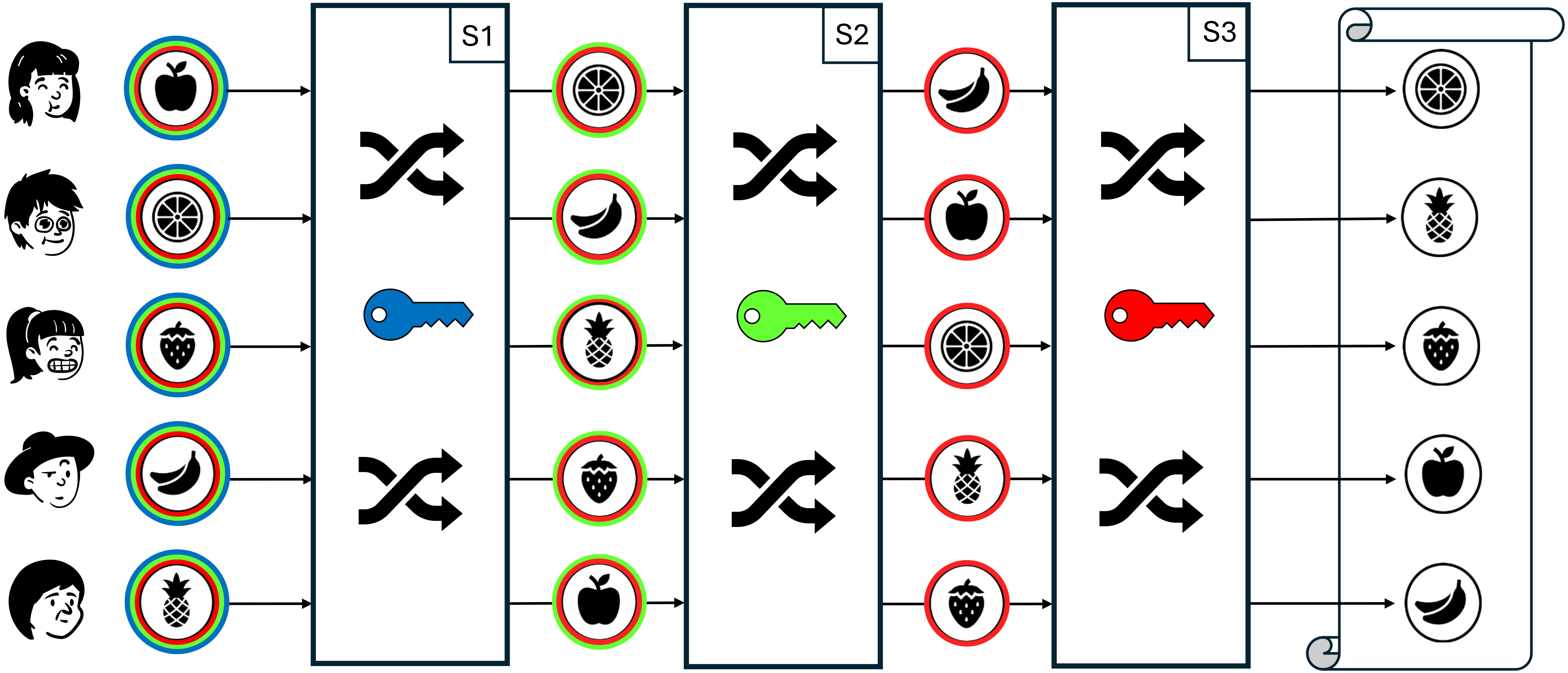


Goal: Untraceable messages + verifiability



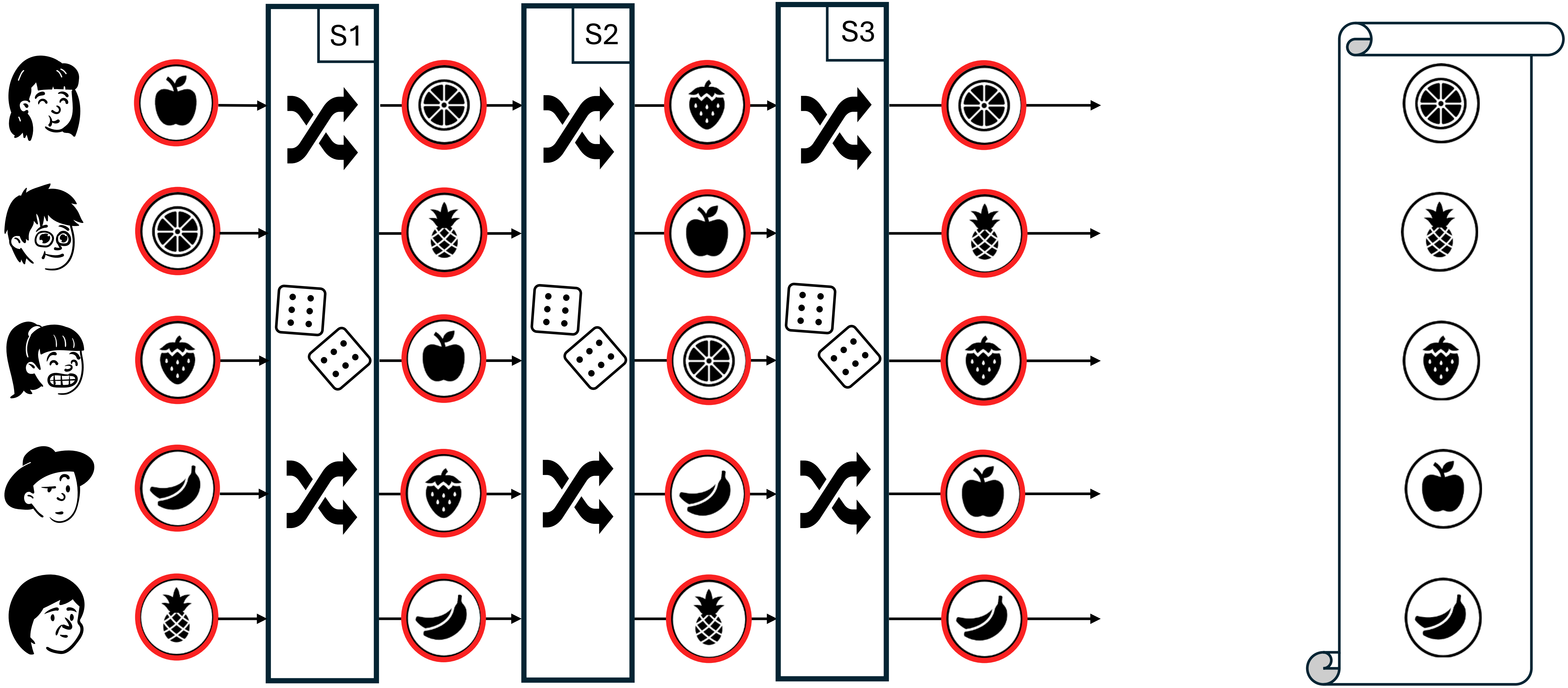
Decryption mixnet

[Cha81]



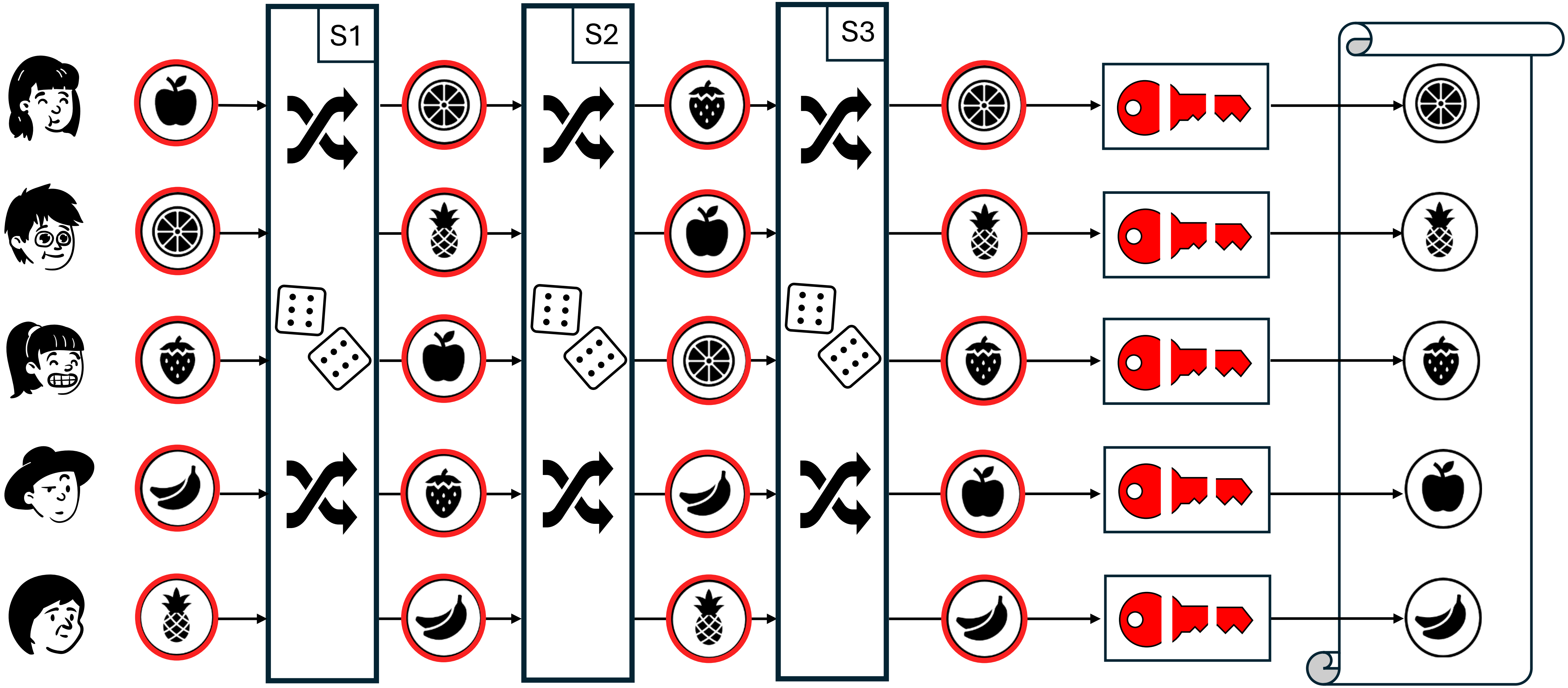
Re-encryption mixnet

[PIK94]



Re-encryption mixnet

[PIK94]



What is the classical approach for constructing verifiable mixnets?

Classical mixnets

Re-encryption mixnet

+

ElGamal encryption

+

Proof of correct shuffle and re-randomization based on DLOG

+

Distributed decryption with proof of correct decryption for ElGamal

~~Classical~~ mixnets
Lattice-based

~~Re-encryption~~ Decryption mixnet

+

~~ElGamal~~ *Kyber* encryption

+

Proof of correct shuffle and ~~re-randomization~~ based on ~~DLOG~~ LWE

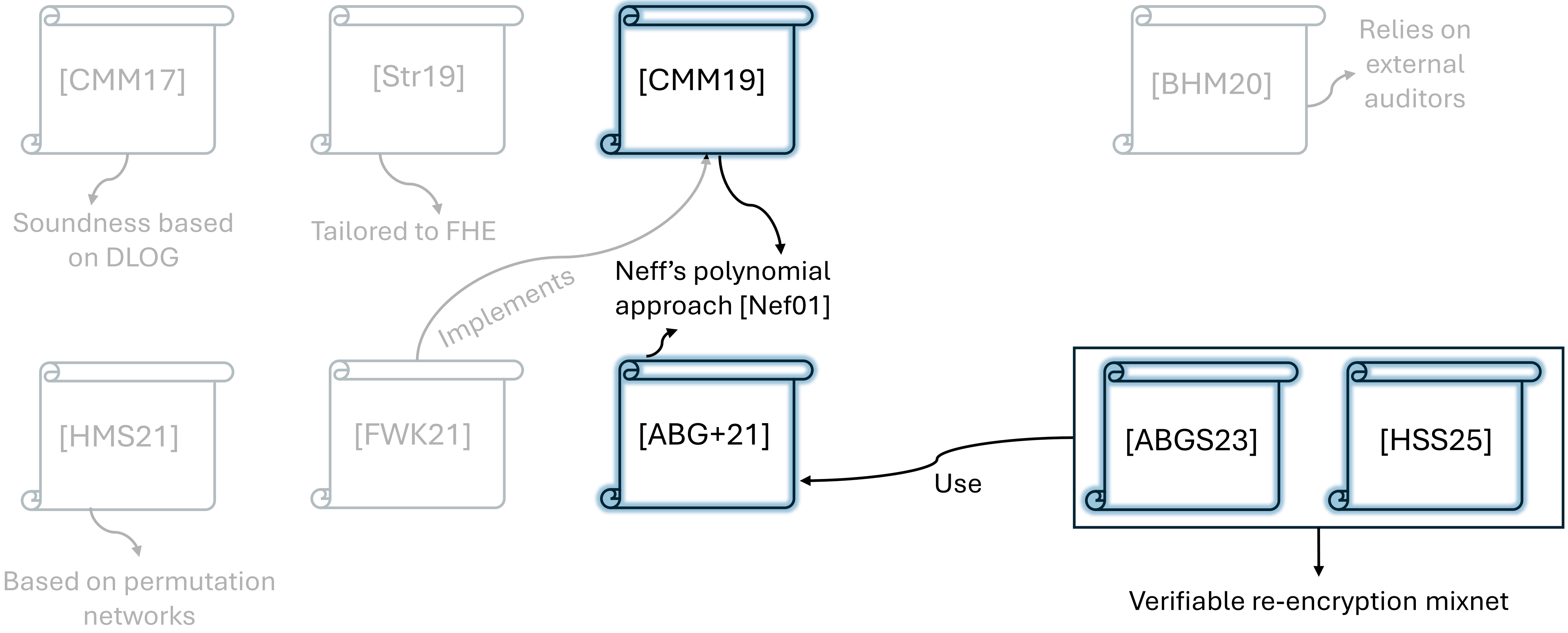
+

~~Distributed~~ decryption with proof of correct decryption for ~~ElGamal~~ Kyber

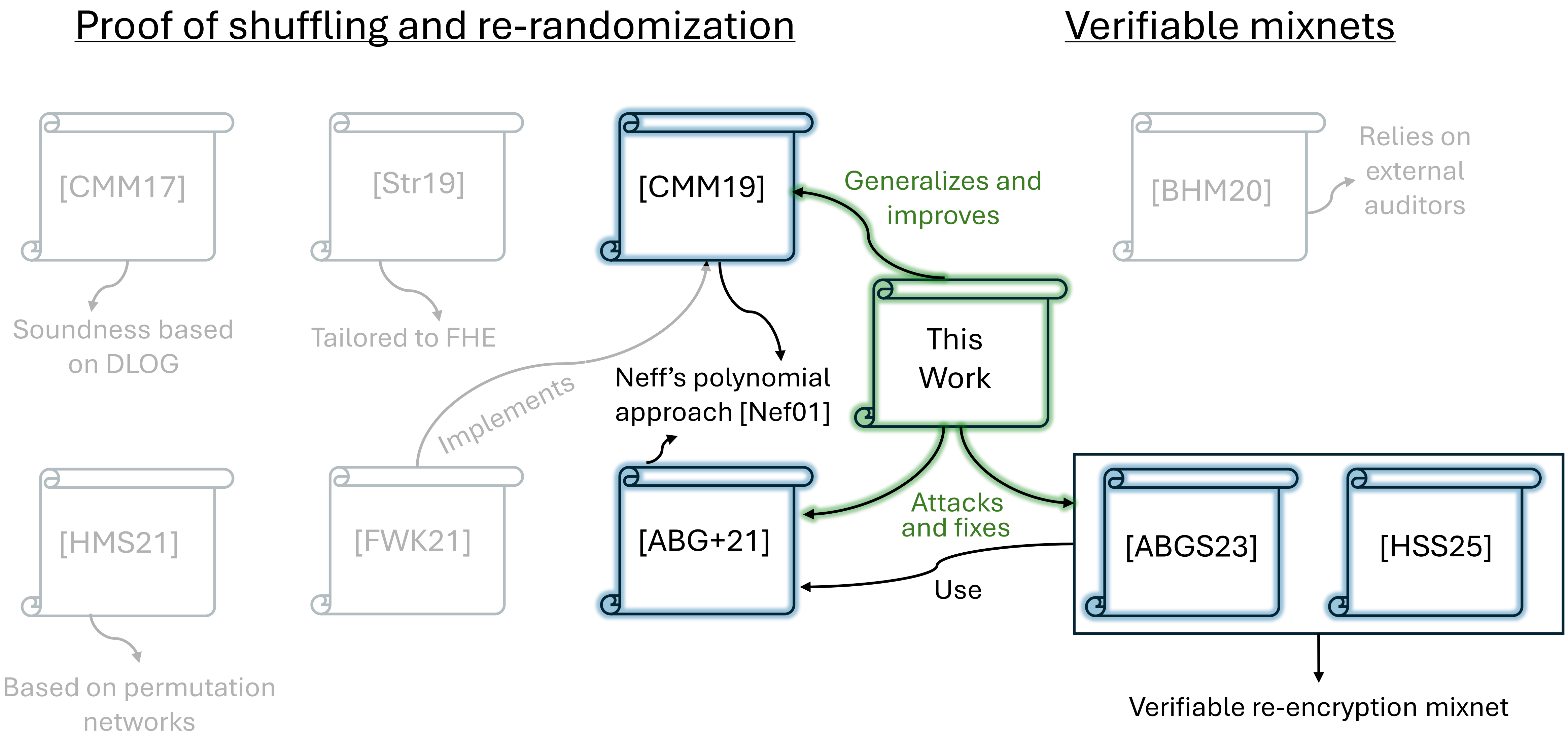
Prior work on lattice-based verifiable mixnets

Proof of shuffling and re-randomization

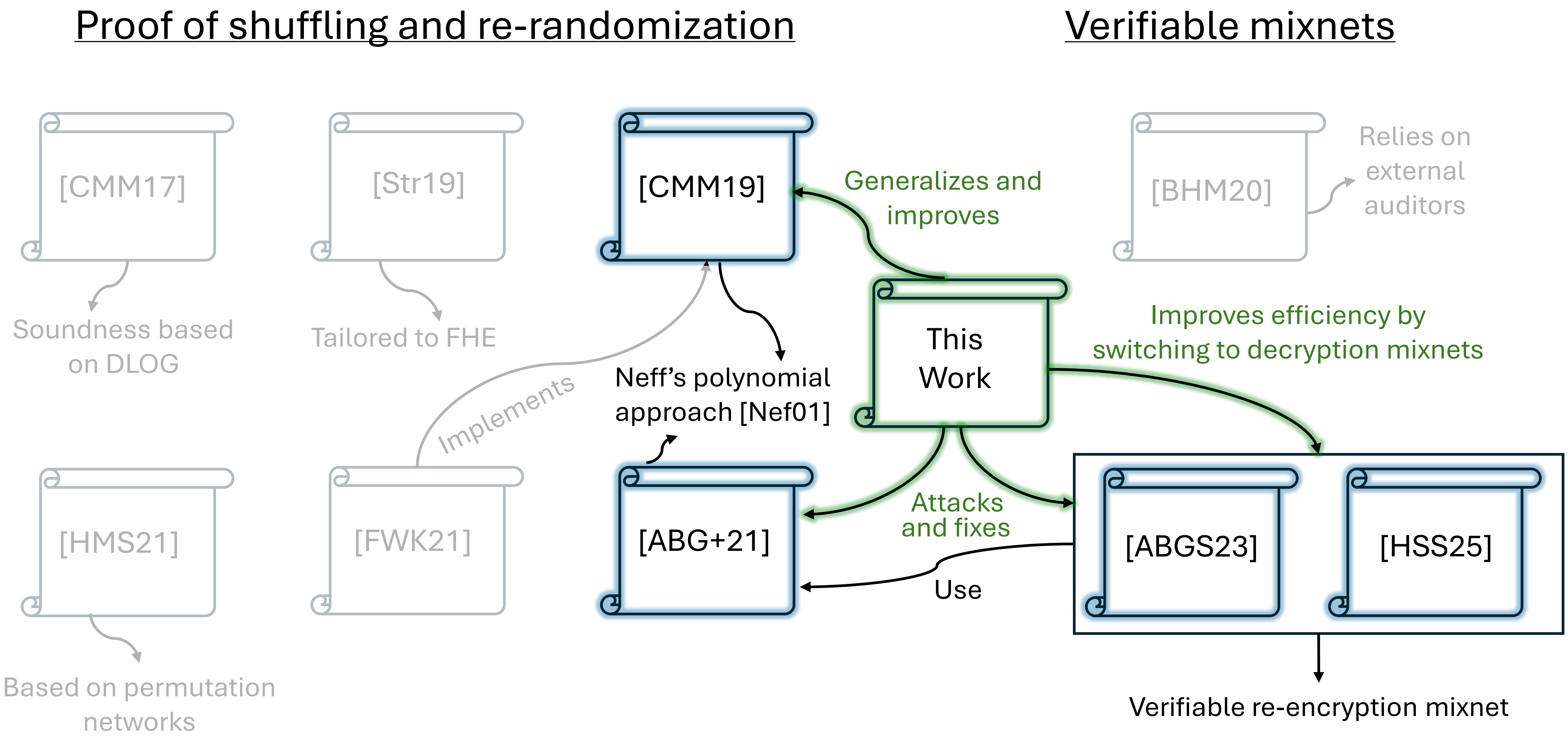
Verifiable mixnets



Contributions



Contributions



Contributions - Comparison

	Modulus q	Ciphertext size	Proof size (per user & server)	Total mixnet size (4 servers, per user)
[ABGS23]	$\approx 2^{78}$	80KB	290KB shuffle + 157KB decryption	2188KB
[HSS25]	$\approx 2^{59}$	15KB	115KB shuffle + 85KB decryption	875KB
This work	3301	6.5KB*	110KB*	467KB

*Average size for a mixnet with 4 layers.

Contributions - Comparison

To be improved with succinct proofs

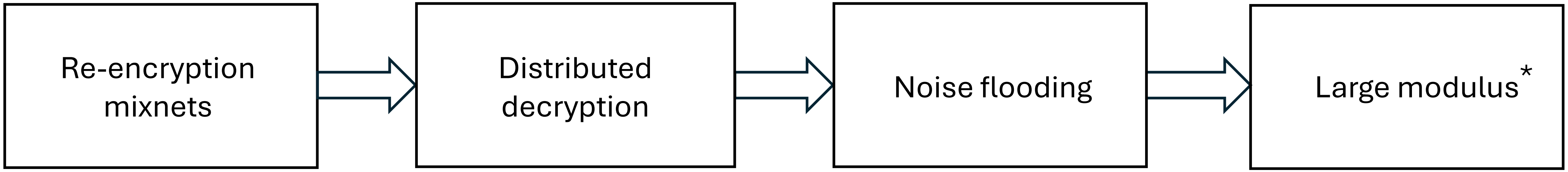
	Modulus q	Ciphertext size	Proof size (per user & server)	Total mixnet size (4 servers, per user)
[ABGS23]	$\approx 2^{78}$	80KB	290KB shuffle + 157KB decryption	2188KB
[HSS25]	$\approx 2^{59}$	15KB	115KB shuffle + 85KB decryption	875KB
This work	3301	6.5KB*	110KB*	467KB

*Average size for a mixnet with 4 layers.

The issue with lattice-based re-encryption mixnets

Re-encryption mixnets ⇒ Distributed decryption ⇒ Noise flooding ⇒ Large modulus

The issue with lattice-based re-encryption mixnets



*[CSS+22, BS23, MS23] allow for polynomial modulus but with limitations:

- [CSS+22, BS23] are tailored to FHE.
- [MS23] assumes honestly distributed ciphertext noise.

Constructing a lattice-based decryption mixnet

Lattice hardness assumptions

Learning **W**ith **E**rrors (LWE) problem

$$\left\{ \left(A \right), \left(A \right)\left(s \right) + \left(e \right) \right\} \approx \left\{ \left(A \right), \left(w \right) \right\}$$

Uniform

Short

Learning **W**ith **R**ounding (LWR) problem

$$\left\{ \left(A \right), \left\lceil \left(A \right)\left(s \right) \right\rfloor \right\} \approx \left\{ \left(A \right), \left\lceil \left(w \right) \right\rfloor \right\}$$

For efficiency, we work over $\mathbb{Z}_q[t]/(t^d + 1)$

Lattice-based decryption mixnet

How to perform the layered encryption? 

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #1

$$\mathrm{Enc}_{pk_1}(\mathrm{Enc}_{pk_2}(m))$$

Lattice-based decryption mixnet

How to perform the layered encryption? 

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #1

$$\mathrm{Enc}_{pk_1}(\mathrm{Enc}_{pk_2}(m))$$

Really large ciphertext expansion!

Lattice-based decryption mixnet

How to perform the layered encryption? 

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #2

$$\mathrm{Enc}_{pk_1}(k_1) \,\|\, \left(\mathrm{AES}_{k_1}\left(\mathrm{Enc}_{pk_2}(k_2) \,\|\, \mathrm{AES}_{k_2}(m)\right)\right)$$

- Linear ciphertext expansion

Lattice-based decryption mixnet

How to perform the layered encryption?

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #2

Not ZKP-friendly: different algebras

$$\mathrm{Enc}_{pk_1}(k_1) \;\Big|\Big|\; \Big(\mathrm{AES}_{k_1}\Big(\mathrm{Enc}_{pk_2}(k_2) \;\Big|\Big|\; \mathrm{AES}_{k_2}(m)\Big)\Big)$$

- Linear ciphertext expansion ✓

Lattice-based decryption mixnet

How to perform the layered encryption? 

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #3

$$\mathrm{Enc}_{pk_1}(k_1) \;\Big|\Big|\; \Big(\lceil \mathrm{A}_1 \cdot k_1 \rfloor + \Big(\mathrm{Enc}_{pk_2}(k_2) \Big|\Big| \big(\lceil \mathrm{A}_2 \cdot k_2 \rfloor + m \big) \Big) \Big)$$

- Linear ciphertext expansion ✓
- Friendly to lattice proofs ✓

Lattice-based decryption mixnet

How to perform the layered encryption? 

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #3

$$\overbrace{\mathrm{Enc}_{pk_1}(k_1)}^{\text{9KB}} \;||\; \left(\lceil \mathrm{A}_1 \cdot k_1 \rfloor + \left(\overbrace{\mathrm{Enc}_{pk_2}(k_2)}^{\text{9KB}} \,||\, (\lceil \mathrm{A}_2 \cdot k_2 \rfloor + m)\right)\right)$$

- Linear ciphertext expansion ✓
- Friendly to lattice proofs ✓

Need k_i large and e.g. ternary for LWR to be hard over Kyber ring (12-bit modulus)

<u>Goal</u>: make k_i as small as possible

Lattice-based decryption mixnet

How to perform the layered encryption?

Define $\mathrm{Enc} = \mathrm{KyberCPA.Enc}$

Attempt #4 ✓

$$\overbrace{\mathrm{Enc}_{pk_1}(k_1)}^{2.3\text{KB}} \; || \; \Big((A_1 \cdot s_1 + e_1) + \Big(\overbrace{\mathrm{Enc}_{pk_2}(k_2)}^{2.3\text{KB}} \, || \, \big((A_2 \cdot s_2 + e_2) + m \big) \Big) \Big)$$

Derive from $[\mathrm{A}_1 \cdot k_1]$

Derive from $[\mathrm{A}_2 \cdot k_2]$

<u>Solution</u>: LWR over small ring with modulus e.g. 64

How to prove a shuffle?

Proof of shuffle in fields

[Neff01]

Lemma 4.1 *Let $\mathbb{F}$ be a field and $N \in \mathbb{N}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \mathbb{F}^N$. If*

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \tag{4.1}$$

over $\mathbb{F}[X]$, then

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N).$$

Proof of shuffle in rings?

[ABG+21]

$R_q = \mathbb{Z}_q[t]/(t^d + 1)$ $\qquad\qquad\qquad\qquad$ R_q^N

Lemma 4.~~1~~2 *Let ~~$\mathbb{F}$ be a field~~ and $N \in \mathbb{N}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \cancel{\mathbb{F}^N}$. If*

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \tag{4.1}$$

$R_q^N[X]$

over $\cancel{\mathbb{F}[X]}$, then

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N).$$

Proof of shuffle in rings?

[ABG+21]

$R_q = \mathbb{Z}_q[t]/(t^d + 1)$

R_q^N

Lemma 4.~~1~~2 *Let ~~$\mathbb{F}$ be a field~~ and $N \in \mathbb{N}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \cancel{\mathbb{F}^N}$. If*

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \tag{4.1}$$

$R_q^N[X]$

over $\cancel{\mathbb{F}[X]}$, then

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N).$$

- R_q is not a field for any choice of q. There is no unique factorization.
- Credits to Katerina Sotiraki for the observation that this approach does not work.

Proof of shuffle in rings?

[ABG+21]

$R_q = \mathbb{Z}_q[t]/(t^d + 1)$

R_q^N

Lemma 4.~~1~~2 *Let ~~$\mathbb{F}$ be a field~~ and $N \in \mathbb{N}$. Let $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \cancel{\mathbb{F}^N}$ R_q^N. If*

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \tag{4.1}$$

$R_q^N[X]$

over $\cancel{\mathbb{F}[X]}$, *then*

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N).$$

amerigal / shuffle-attack

- R_q is not a field for any choice of q. There is no unique factorization.
- Credits to Katerina Sotiraki for the observation that this approach does not work.

Proof of shuffle in rings – This work

- $\mathfrak{R} := \mathfrak{R}_1 \times \cdots \times \mathfrak{R}_k$, with $\mathfrak{R}_i$ integral domains
- $D \subset \mathfrak{R}$ with invertible differences
- $g: \{1, \ldots, N\} \rightarrow D$ injective
- $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \mathfrak{R}^N$

Proof of shuffle in rings – This work

- $\mathfrak{R} \coloneqq \mathfrak{R}_1 \times \cdots \times \mathfrak{R}_k$, with $\mathfrak{R}_i$ integral domains
- $D \subset \mathfrak{R}$ with invertible differences
- $g: \{1, \dots, N\} \to D$ injective
- $(a_1, \dots, a_N), (b_1, \dots, b_N) \in \mathfrak{R}^N$

Lemma 4.2 If there exist $(\sigma_1, \dots, \sigma_N) \in D^N$ such that

$$\prod_{i=1}^{N} (a_i + g(i) \cdot X_1 - X_2) = \prod_{i=1}^{N} (b_i + \sigma_i \cdot X_1 - X_2)$$

over $\mathfrak{R}[X_1, X_2]$, then

$$(a_1, \dots, a_N) \sim_P (b_1, \dots, b_N).$$

Proof of shuffle in rings – This work

- $\mathfrak{R} := \mathfrak{R}_1 \times \cdots \times \mathfrak{R}_k$, with $\mathfrak{R}_i$ integral domains
- $D \subset \mathfrak{R}$ with invertible differences
- $g: \{1, \ldots, N\} \to D$ injective
- $(a_1, \ldots, a_N), (b_1, \ldots, b_N) \in \mathfrak{R}^N$

Lemma 4.2 If there exist $(\sigma_1, \ldots, \sigma_N) \in D^N$ such that

$$\prod_{i=1}^{N} (a_i + g(i) \cdot X_1 - X_2) = \prod_{i=1}^{N} (b_i + \sigma_i \cdot X_1 - X_2)$$

over $\mathfrak{R}[X_1, X_2]$, then

$$(a_1, \ldots, a_N) \sim_P (b_1, \ldots, b_N).$$

- Generalization of [CMM19], whose product expression comes from [BG12].

We choose different D and g.

We use the product directly on the messages.

Future work and open questions

- Instantiate mixnets with succinct zero-knowledge proofs.
- Upgrade to IND-CCA security via the Naor-Yung paradigm.
- Analyze if recent techniques to achieve distributed decryption with polynomial modulus can be adapted to mixnets.

Thank you!

References

[ABG+21] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, Tjerand Silde, and Thor Tunge. Lattice-based proof of shuffle and applications to electronic voting. In Kenneth G. Paterson, editor, *CT-RSA 2021*, volume 12704 of *LNCS*, pages 227–251. Springer, Cham, May 2021.

[ABGS23] Diego F. Aranha, Carsten Baum, Kristian Gjøsteen, and Tjerand Silde. Verifiable mix-nets and distributed decryption for voting from lattice-based assumptions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1467–1481, New York, NY, USA, 2023. Association for Computing Machinery.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Berlin, Heidelberg, April 2012.

[BHM20] Xavier Boyen, Thomas Haines, and Johannes Müller. A verifiable and practical lattice-based decryption mix net with external auditing. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 336–356. Springer, Cham, September 2020.

[BS23] Katharina Boudgoust and Peter Scholl. Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part I*, volume 14438 of *LNCS*, pages 371–404. Springer, Singapore, December 2023.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, feb 1981.

[CMM17] Nuria Costa, Ramiro Martínez, and Paz Morillo. Proof of a shuffle for lattice-based cryptography. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevičius, editors, *NordSec 2017*, volume 10674 of *LNCS*, pages 280–296, Cham, 2017. Springer.

[CMM19] Núria Costa, Ramiro Martínez, and Paz Morillo. Lattice-based proof of a shuffle. In Andrea Bracciali, Jeremy Clark, Federico Pintore, Peter B. Rønne, and Massimiliano Sala, editors, *FC 2019 Workshops*, volume 11599 of *LNCS*, pages 330–346. Springer, Cham, February 2019.

[CSS+22] Siddhartha Chowdhury, Sayani Sinha, Animesh Singh, Shubham Mishra, Chandan Chaudhary, Sikhar Patranabis, Pratyay Mukherjee, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Efficient threshold FHE with application to real-time systems. Cryptology ePrint Archive, Report 2022/1625, 2022. `https://eprint.iacr.org/2022/1625`.

[FWK21] Valeh Farzaliyev, Jan Willemson, and Jaan Kristjan Kaasik. Improved lattice-based mix-nets for electronic voting. In Jong Hwan Park and Seung-Hyun Seo, editors, *ICISC 21*, volume 13218 of *LNCS*, pages 119–136. Springer, Cham, December 2021.

[HM20] Thomas Haines and Johannes Müller. SoK: Techniques for verifiable mix nets. In Limin Jia and Ralf Küsters, editors, *CSF 2020 Computer Security Foundations Symposium*, pages 49–64. IEEE Computer Society Press, 2020.

[HMS21] Javier Herranz, Ramiro Martínez, and Manuel Sánchez. Shorter lattice-based zero-knowledge proofs for the correctness of a shuffle. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *FC 2021 Workshops*, volume 12676 of *LNCS*, pages 315–329. Springer, Berlin, Heidelberg, March 2021.

[HSS25] Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. More efficient lattice-based electronic voting from NTRU. *IACR Communications in Cryptology*, 1(4), 2025.

[MS23] Daniele Micciancio and Adam Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. Cryptology ePrint Archive, Paper 2023/1728, 2023.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001.

[PIK94] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 248–259. Springer, Berlin, Heidelberg, May 1994.

[Str19] Martin Strand. A verifiable shuffle for the GSW cryptosystem. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 165–180. Springer, Berlin, Heidelberg, March 2019.

Extra: Proof of shuffle from [ABG+21] - Attack

$a_i, b_i \in R_q = \mathbb{Z}_q[t]/(t^d + 1)$

$$\prod_{i=1}^{N}(a_i - X) = \prod_{i=1}^{N}(b_i - X) \quad \not\Rightarrow \quad (a_1, a_2, \ldots, a_N) \sim_P (b_1, b_2, \ldots, b_N)$$

- Suppose $R_q \cong \mathbb{Z}_q[t]/p_1 \times \mathbb{Z}_q[t]/p_2$

- Let $a_1, a_2 \in R_q$ and denote

$$\begin{cases} a_1 \equiv (a_{11}, a_{12}) \\ a_2 \equiv (a_{21}, a_{22}) \end{cases}$$

- We construct $b_1, b_2 \in R_q$ as

$$\begin{cases} b_1 \equiv (a_{11}, a_{22}) \\ b_2 \equiv (a_{21}, a_{12}) \end{cases}$$

- Then in general $(a_1, a_2) \nsim_P (b_1, b_2)$ but

$$(a_1 - X)(a_2 - X) = (b_1 - X)(b_2 - X) \quad !!$$

$$b_2 \equiv (a_{21}, a_{12})$$