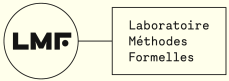
Modularity of mix-server security proofs in the CCSA logic: case of Bayer-Groth protocol

Myrto Arapinis[1] **Margot Catinaud**[2] Caroline Fontaine[2] Guillaume Scerri [2]

[1]The University of Edinburgh
Edinburgh, Scotland

[2]Université Paris-Saclay, CNRS, ENS Paris-Saclay,
Laboratoire Méthodes Formelles (LMF)
Gif-sur-Yvette, France



GT MFS, March 2026, Luz-Saint-Sauveur

Introduction – Overview of the talk

Voting protocol chronology

Voting phase

Mix

Tally

Main security properties looked for

Vote privacy

Verifiability
(universal & individual)

Introduction – Overview of the talk

Voting protocol chronology

Voting phase

Mix

Tally

Main security properties looked for

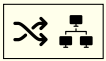
Vote privacy

Verifiability
(universal & individual)

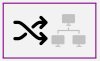
Cryptographic tools

First contribution:
Security proofs of mix-servers

Second contribution:
compose mix-servers
towards mixnets security

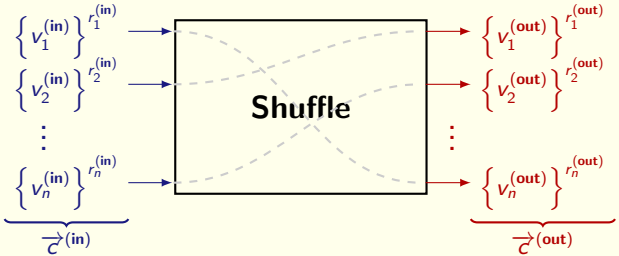


Third contribution:
From game secure mixnets
towards UC secure mixnets



– Mix-servers oriented to e-voting protocols

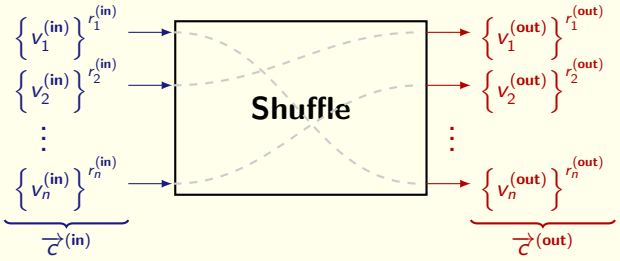
Mix-servers in a nutshell



$$\exists\, \pi \in \mathfrak{S}_n,\ \forall\, i \in [\![1; n]\!],\ v_i^{(\text{out})} = v_{\pi(i)}^{(\text{in})}$$

– Mix-servers oriented to e-voting protocols

Mix-servers in a nutshell

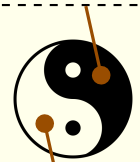


$$\exists \pi \in \mathfrak{S}_n,\ \forall i \in [\![1; n]\!],\ v_i^{(out)} = v_{\pi(i)}^{(in)}$$

Security properties looked for

Permutation secrecy

The attacker **must not guess** the permutation used to shuffle lists

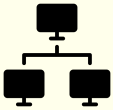


Vote integrity

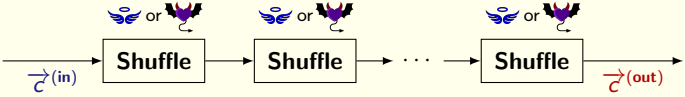
Plaintexts **must** stay **untouched**

– Mixnets

Network of mix-servers



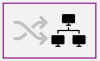
Mix-servers scheduling



Each mix-server can be

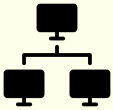
or

Honest Dishonest

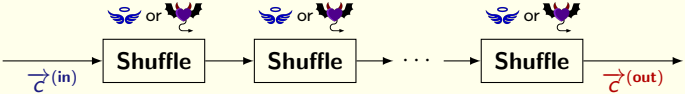


– Mixnets

Network of mix-servers



Mix-servers scheduling



Each mix-server can be

Honest

or

Dishonest

Mixnet security properties

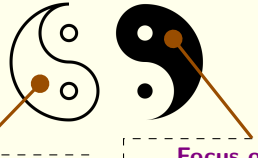
- **Vote confidentiality**

  Ensured provided that *at least* one mix-server is honest

- **Resistance to malicious mix-servers**

  Any dishonest mix-server **should not be able to** inject or dismiss votes

– Achieve both vote integrity and permutation secrecy: An oxymoron at first sight



Focus on vote integrity only

- **Idea:**

  Reveal all shuffle parameters used

- **Pitfall:**

   **knows** everything

  $\Longrightarrow$ permutation secrecy hopeless

Focus on permutation secrecy only

- **Idea:**

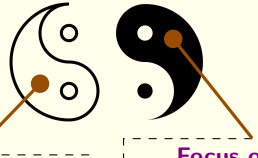
  Keep shuffle parameters (π and $\overrightarrow{r}^{(\mathbf{out})}$) secret

- **Pitfall:**

   **can do** anything they wants

  $\Longrightarrow$ vote integrity hopeless

– Achieve both vote integrity and permutation secrecy: An oxymoron at first sight



Focus on vote integrity only

- **Idea:**

  Reveal all shuffle parameters used

- **Pitfall:**

  knows everything

  $\Longrightarrow$ permutation secrecy hopeless

Focus on permutation secrecy only

- **Idea:**

  Keep shuffle parameters (π and $\overrightarrow{r}^{(\mathbf{out})}$) secret

- **Pitfall:**

  can do anything they wants

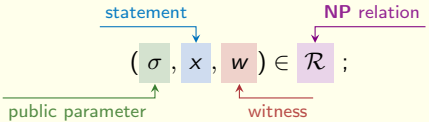
  $\Longrightarrow$ vote integrity hopeless

Solve the apparent unsolvable dilemma: *"prove"* shuffle **without** *leaking* permutation

– (Interactive) Zero-knowledge proofs

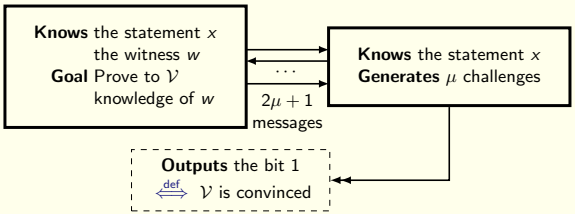
Principle

- **Two agents** a prover $\mathcal{P}$ and a verifier $\mathcal{V}$;
- **Goal:** prove equation



- **Special cases:** NIZK ($\mu = 0$), Σ-protocols ($\mu = 1$).

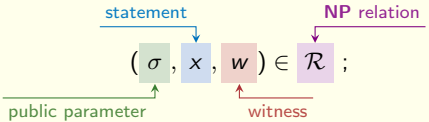
Prover $\mathcal{P}$ **Verifier $\mathcal{V}$**



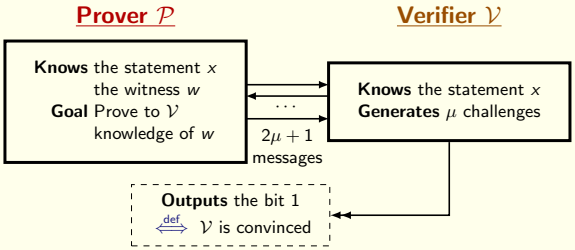
– (Interactive) Zero-knowledge proofs

Principle

- **Two agents** a prover $\mathcal{P}$ and a verifier $\mathcal{V}$;
- **Goal:** prove equation



- **Special cases:** NIZK ($\mu = 0$), Σ-protocols ($\mu = 1$).



Basic properties

Zero-Knowledge

Settings:

$\mathcal{P}$ *honest* prover

$\mathcal{V}$ *dishonest* verifier

Soundness

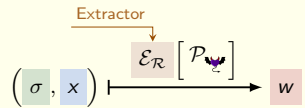
Settings:

$\mathcal{P}$ *dishonest* prover

$\mathcal{V}$ *honest* verifier

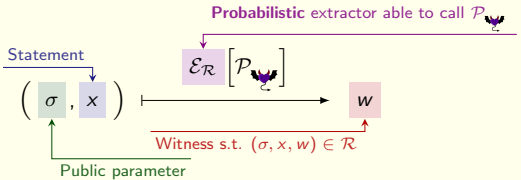
Usual way to prove soundness

Knowledge soundness



– Zoom on knowledge soundness property of ZK proofs and rewinding technique

Knowledge soundness

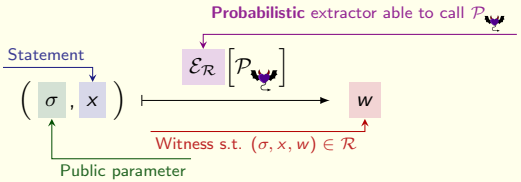


Warning

Zero-Knowledge $\Longrightarrow$ We need a *"trick"*

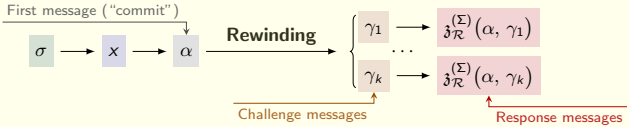
– Zoom on knowledge soundness property of ZK proofs and rewinding technique

Knowledge soundness



Usual "trick" for interactive ZK proofs

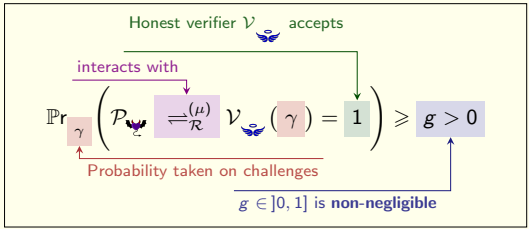
Use the rewinding technique – lightning example of Σ-protocols

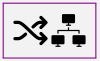


Warning

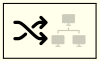
Zero-Knowledge $\Longrightarrow$ We need a *"trick"*

Hypothesis made on $\mathcal{P}$



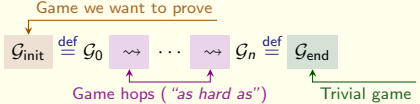


– Contributions going from *mix-servers* to *mixnets*



Mix-server security

Traditional framework used: Cryptographic games



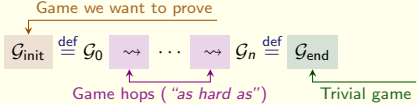
First contribution

Are logical tools developped modular enough?

– Contributions going from *mix-servers* to *mixnets*

Mix-server security

Traditional framework used: Cryptographic games

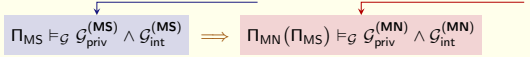


First contribution

Are logical tools developped modular enough?

Mixnet security

Traditional framework used: *Universal Composability* (UC)



Second contribution

mix-server security $\Longrightarrow$ mixnet security

– Contributions going from *mix-servers* to *mixnets*

Mix-server security

Traditional framework used: Cryptographic games

Game we want to prove

$\mathcal{G}_{\text{init}} \overset{\text{def}}{=} \mathcal{G}_0 \rightsquigarrow \cdots \rightsquigarrow \mathcal{G}_n \overset{\text{def}}{=} \mathcal{G}_{\text{end}}$

Game hops (*"as hard as"*) — Trivial game

First contribution

Are logical tools developped modular enough?

Mixnet security

Traditional framework used: *Universal Composability* (UC)

Secure shuffle protocol — Secure mixnet protocol

$\Pi_{\text{MS}} \vDash_{\mathcal{G}} \mathcal{G}^{(\text{MS})}_{\text{priv}} \wedge \mathcal{G}^{(\text{MS})}_{\text{int}} \implies \Pi_{\text{MN}}(\Pi_{\text{MS}}) \vDash_{\mathcal{G}} \mathcal{G}^{(\text{MN})}_{\text{priv}} \wedge \mathcal{G}^{(\text{MN})}_{\text{int}}$

Second contribution

mix-server security $\implies$ mixnet security

A link between frameworks

Game-based secure mixnets — UC secure mixnets

$\Pi_{\text{MN}} \vDash_{\mathcal{G}} \mathcal{G}^{(\text{MN})}_{\text{priv}} \wedge \mathcal{G}^{(\text{MN})}_{\text{int}} \iff \Pi_{\text{MN}} \preccurlyeq_{\text{uc}} \mathcal{F}_{\text{MN}}$

Third contribution

From **game-based secure** mixnets towards **UC secure** mixnets

– *Computationally Complete Symbolic Attacker* model

The CCSA model

	Syntax	Semantics $[\![\cdot]\!]^{\rho}_{\eta}$
Attacker model	Recursive function (frame)	Probabilistic Polynomial-time Turing Machines (PPTMs)
Messages	Terms algebra (t)	Bitstrings ($[\![t]\!]^{\rho}_{\eta} \in \{0,1\}^*$)
Proof techniques	Rewriting / induction	Reductions (complexity-like)

Arguments in favour of this logic

- Best trade-off between **expressivity**[1] and **abstraction**;
- **Mechanisable**[2] in a proof assistant (SQUIRREL);
- Proof can be adapted to obtain **concrete security**[3].

[1] *Baelde D., Koutsos A., Lallemand J.*, **A Higher-Order Indistinguishability Logic for Cryptographic Reasoning**, LICS, 2023
[2] *Baelde D., Delaune S., Jacomme C., Koutsos A., Lallemand J.*, **The Squirrel Prover and its Logic**, ACM SIGLOG News, 2024
[3] *Baelde D., Fontaine C., Koutsos A., Scerri G., Vignon T.*, **A Probabilistic Logic for Concrete Security**, CSF, 2024

– First contribution – State of the art of shuffle protocols

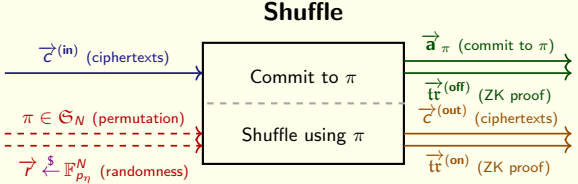
Where are we?

See our paper

hal.science/hal-04937921v1

CCSA proof of Terelius & Wikström shuffle protocol $\left([\text{TW10}], [\text{CFS25}]\right)$

Shuffle protocols basics



Two zero-knowledge proofs

- Commit to a permutation:

$$\overrightarrow{\mathrm{tr}}^{(\text{off})} \vDash_{\mathbb{ZK}} \overrightarrow{\mathbf{a}}_\pi = \mathrm{Com}(\, ck \,,\, \pi \,;\, \mathbf{s} \,) \;;$$

- Commitment-consistent shuffle:

$$\overrightarrow{\mathrm{tr}}^{(\text{on})} \vDash_{\mathbb{ZK}} \forall\, i \in [\![1; N]\!],\; c^{(\text{out})}_{\pi(i)} = \mathbf{reenc}\left(c_i^{(\text{in})} \,;\, r_i \right)$$

– First contribution – State of the art of shuffle protocols

Where are we?

See our paper

hal.science/hal-04937921v1

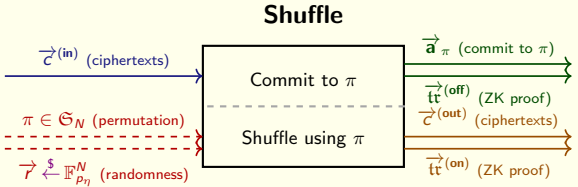
CCSA proof of Terelius & Wikström shuffle protocol ([TW10], [CFS25])

Next target

SwissPost

Bayer & Groth shuffle protocol ([BG12])

Shuffle protocols basics



Two zero-knowledge proofs

- Commit to a permutation:

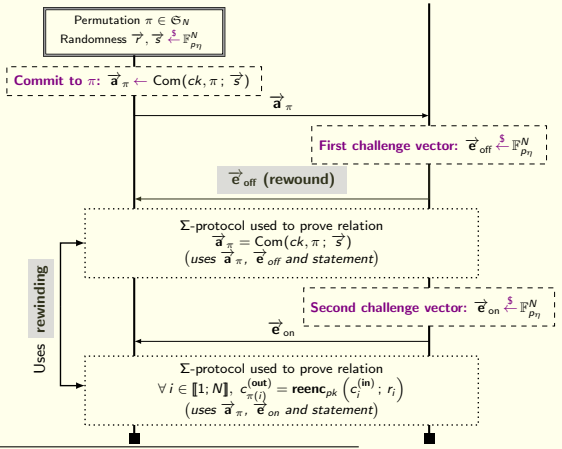
$$\overrightarrow{\text{tt}}^{(\text{off})} \vDash_{\mathbb{ZK}} \overrightarrow{a}_\pi = \text{Com}(\, ck \,,\, \pi \,;\, \mathbf{s}\,) \;;$$

- Commitment-consistent shuffle:

$$\overrightarrow{\text{tt}}^{(\text{on})} \vDash_{\mathbb{ZK}} \forall\, i \in [\![1; N]\!],\; c^{(\text{out})}_{\pi(i)} = \textbf{reenc}\left(c_i^{(\text{in})} \,;\, r_i \right)$$

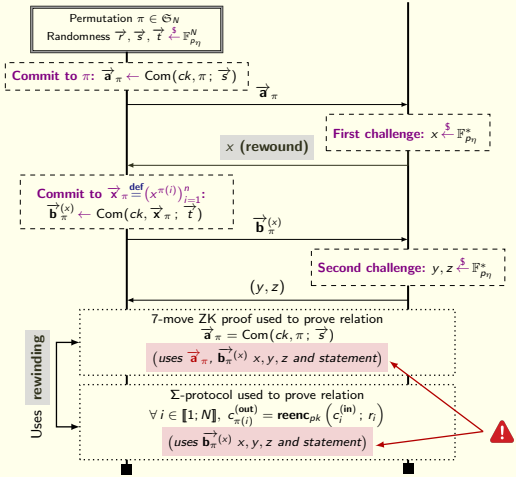
– First contribution – Comparison between TW and BG shuffle protocols

Terelius-Wikström shuffle protocol* ($\mu_{\mathsf{TW}} = 4$)



*Simplified to make them clearer

Bayer-Groth shuffle protocol* ($\mu_{\mathsf{BG}} = 6$)



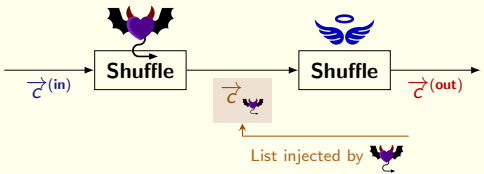
– Second contribution – From **mix-servers** to **mixnet** security: Difficulties

	Mix-server	Mixnet
Vote privacy game $\mathcal{G}_{\text{priv}}$	"Easy" (ZK property)	**Hard** (coming next)
Verifiability game $\mathcal{G}_{\text{int}}$	**Hard** (rewinding)	"Easy" (composition)

– Second contribution – From **mix-servers** to **mixnet** security: Difficulties

	Mix-server	Mixnet
Vote privacy game $\mathcal{G}_{\text{priv}}$	"Easy" (ZK property)	**Hard** (coming next)
Verifiability game $\mathcal{G}_{\text{int}}$	**Hard** (rewinding)	"Easy" (composition)

An attack on confidentiality without resistance to malicious mix-servers



knows ▷ $\vec{c}^{(\text{out})}$

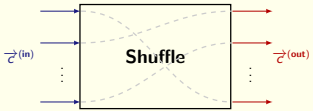
▷ underlying plaintexts of $\vec{c}$

$\Longrightarrow$ **can deduce** permutation

– Second contribution – **Mix-servers** towards **mixnet** security games

Mix-server security cryptographic games

↪ $\mathcal{G}_{\text{int}}^{(\mathbf{MS})}$ – Verifiability game



$\mathcal{G}_{\text{int}}^{(\mathbf{MS})} \overset{\text{means}}{\Longleftrightarrow} \left(\mathbf{valid}_n^{(\mathbf{in})}\ \overrightarrow{c}^{(\mathbf{in})}\ v_{\text{in}}\right) \rightarrow \left(\mathbf{valid}_n^{(\mathbf{out})}\ \overrightarrow{c}^{(\mathbf{out})}\ v_{\text{out}}\right)$

Mixnet security cryptographic games

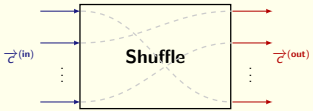
↪ $\mathcal{G}_{\text{int}}^{(\mathbf{MN})}$ – Verifiability game

- ▷ Either for and , integrity game $\mathcal{G}_{\text{int}}^{(\mathbf{MS})}$ holds;
- ▷ Good composition of $\mathbf{valid}_n$ predicates $\Longrightarrow$ $\mathcal{G}_{\text{int}}^{(\mathbf{MN})}$ holds.

– Second contribution – **Mix-servers** towards **mixnet** security games

Mix-server security cryptographic games

↪ $\mathcal{G}_{\text{int}}^{(\mathbf{MS})}$ – Verifiability game



$$\mathcal{G}_{\text{int}}^{(\mathbf{MS})} \overset{\text{means}}{\iff} \left(\mathbf{valid}_n^{(\mathbf{in})}\ \vec{c}^{(\mathbf{in})}\ v_{\text{in}}\right) \to \left(\mathbf{valid}_n^{(\mathbf{out})}\ \vec{c}^{(\mathbf{out})}\ v_{\text{out}}\right)$$

↪ $\mathcal{G}_{\text{priv}}^{(\mathbf{MS})}$ – Privacy game

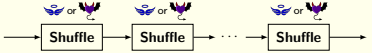
Two **different** permutations, given by

$$\mathbf{shuffle}\ \pi_1\ \left(\mathbf{prepare}\ L\right) \sim \mathbf{shuffle}\ \pi_2\ \left(\mathbf{prepare}\ L\right)$$

Same plaintexts lists, **not** controlled by

Mixnet security cryptographic games

↪ $\mathcal{G}_{\text{int}}^{(\mathbf{MN})}$ – Verifiability game



- ▷ Either for and , integrity game $\mathcal{G}_{\text{int}}^{(\mathbf{MS})}$ holds;
- ▷ Good composition of $\mathbf{valid}_n$ predicates $\implies$ $\mathcal{G}_{\text{int}}^{(\mathbf{MN})}$ holds.

↪ $\mathcal{G}_{\text{priv}}^{(\mathbf{MN})}$ – Privacy game

- ▷ Can be reduced to the case of **only one** honest mix-server

Two plaintexts lists, only differs by their order, chosen by

$$\mathbf{mix}\ \pi\ \left(\mathbf{prepare}\ L\right) \sim \mathbf{mix}\ \pi\ \left(\mathbf{prepare}\ \pi_{\text{?}}(L)\right)$$

Random permutation, **not** controlled by

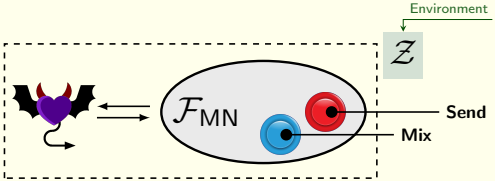
– Third contribution – From games to UC

Which ideal functionality do we talk about?

Why not mix-servers?

More naturally expressed (and easier to do) through games: $\mathcal{G}_{\text{int}}^{(\mathbf{MS})}$ and $\mathcal{G}_{\text{priv}}^{(\mathbf{MS})}$

Traditional way: ideal mixnet



[1] *Wikström D.*, **A Universally Composable Mixnet**, TCC, 2004

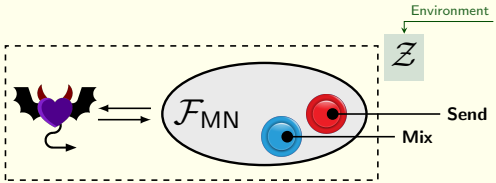
– Third contribution – From games to UC

Which ideal functionality do we talk about?

Why not mix-servers?

More naturally expressed (and easier to do) through games: $\mathcal{G}_{\text{int}}^{(\mathbf{MS})}$ and $\mathcal{G}_{\text{priv}}^{(\mathbf{MS})}$

Traditional way: ideal mixnet



Ideal functionality studied[1]

Ideal functionality – Ideal mixnet $\mathcal{F}_{\text{MN}}$

Parameters: $n, p \in \mathbb{N}^*$; $\theta \in]0, 1]$ mix threshold.
Agents: p mix-servers $(\text{M}_j)_{j=1}^{n}$, n senders $(\text{S}_i)_{i=1}^{n}$, ideal adversary.
Commands: ↪ **Send** – acts on senders; ↪ **Mix** – acts on mix-servers.
Notable facts: ▷ Environment schedules agents; ▷ Includes decryption; ▷ Environment knows votes.

Main objective (still under work)

Theorem – Game-based secure toward UC secure mixnets

$$\Pi_{\text{MN}} \vDash_{\mathcal{G}} \mathcal{G}_{\text{int}}^{(\mathbf{MN})} \wedge \mathcal{G}_{\text{priv}}^{(\mathbf{MN})} \iff \Pi_{\text{MN}} \preccurlyeq_{\text{uc}} \mathcal{F}_{\text{MN}}$$

[1] *Wikström D.*, **A Universally Composable Mixnet**, TCC, 2004

Key take-aways and prospectives

Key take aways

3 main research axes

- ▷ **(Mix-servers)** Security proofs of shuffle protocols;
- ▷ **(Composition)** From mix-servers security to mixnets security;
- ▷ **(Mixnets)** UC realization of ideal functionality $\mathcal{F}_{\mathsf{MN}}$.

Prospectives

Key take-aways and prospectives

Key take aways

3 main research axes

- **(Mix-servers)** Security proofs of shuffle protocols;
- **(Composition)** From mix-servers security to mixnets security;
- **(Mixnets)** UC realization of ideal functionality $\mathcal{F}_{\mathsf{MN}}$.

2 cryptographic/logic frameworks used

- *Universal composability* (UC)
- *Computationally Complete Symbolic Attacker* (CCSA)

2 security properties studied

Vote privacy & Verifiability

Prospectives

Key take-aways and prospectives

Key take aways

3 main research axes

- **(Mix-servers)** Security proofs of shuffle protocols;
- **(Composition)** From mix-servers security to mixnets security;
- **(Mixnets)** UC realization of ideal functionality $\mathcal{F}_{\mathsf{MN}}$.

2 cryptographic/logic frameworks used

- *Universal composability* (UC)
- *Computationally Complete Symbolic Attacker* (CCSA)

2 security properties studied

Vote privacy & Verifiability

Success

- CCSA axiom of **rewinding** technique quite **suitable** (\o/ yay!).

Prospectives

Key take-aways and prospectives

Key take aways

3 main research axes

- **(Mix-servers)** Security proofs of shuffle protocols;
- **(Composition)** From mix-servers security to mixnets security;
- **(Mixnets)** UC realization of ideal functionality $\mathcal{F}_{\mathsf{MN}}$.

2 cryptographic/logic frameworks used

- *Universal composability* (UC)
- *Computationally Complete Symbolic Attacker* (CCSA)

2 security properties studied

Vote privacy & Verifiability

Success

- CCSA axiom of **rewinding** technique quite **suitable** (\o/ yay!).

Prospectives

Zero-knowledge proofs

- From interactive ZK proofs to Non-Interactive ones (through **Fiat-Shamir transform**).

CCSA logic

- Handle **re-programmable Random Oracles**

E-voting protocols

- Look at **interactions** between mixnets and other parts of e-voting protocols

Key take-aways and prospectives

Key take aways

3 main research axes

- **(Mix-servers)** Security proofs of shuffle protocols;
- **(Composition)** From mix-servers security to mixnets security;
- **(Mixnets)** UC realization of ideal functionality $\mathcal{F}_{\mathsf{MN}}$.

2 cryptographic/logic frameworks used

- *Universal composability* (UC)
- *Computationally Complete Symbolic Attacker* (CCSA)

2 security properties studied

Vote privacy & Verifiability

Success

- CCSA axiom of **rewinding** technique quite **suitable** (\o/ yay!).

Prospectives

Zero-knowledge proofs

- From interactive ZK proofs to Non-Interactive ones (through **Fiat-Shamir transform**).

CCSA logic

- Handle **re-programmable Random Oracles**

E-voting protocols

- Look at **interactions** between mixnets and other parts of e-voting protocols

Thank you for your attention!

mcatinaud.gitlab.io

Bibliography

CCSA logic related

[BKL23] *Baelde D., Koutsos A., Lallemand J.*, **A Higher-Order Indistinguishability Logic for Cryptographic Reasoning**, LICS, 2023

[BDJKL24] *Baelde D., Delaune S., Jacomme C., Koutsos A., Lallemand J.*, **The Squirrel Prover and its Logic**, ACM SIGLOG News, 2024

[BFKSV24] *Baelde D., Fontaine C., Koutsos A., Scerri G., Vignon T.*, **A Probabilistic Logic for Concrete Security**, CSF, 2024

Cryptographic papers

[TW10] *Terelius B., Wikström D.*, **Proof of restricted shuffles**, AFRICACRYPT, 2010

[BG12] *Bayer S., Groth J.*, **Efficient zero-knowledge argument for correctness of a shuffle**, EUROCRYPT, 2012

E-voting protocols papers

— *Glondu S.*, **Belenios specification**, Version 3.0

— *Swiss Post*, **Swiss Post Voting System – System specification**, Version 1.5.2, 2025

UC related papers

[W04] *Wikström D.*, **A Universally Composable Mix-Net**, TCC, 2004

[CS25] *Chevalier C., Sageloli E.*, **Diving Deep Into UC: Uncovering and Resolving Issues in Universal Composability**, IACR Cryptol. ePrint Arch, 2025

Our paper

[CFS25] **Catinaud M.**, *Fontaine C., Scerri G.*, **Proving E-voting Mixnets in the CCSA model: Zero-Knowledge proofs and Rewinding**, HAL Arch, 2025

Most of icons come from Flaticon