CRS-Updatable Asymmetric Quasi-Adaptive NIZK Arguments

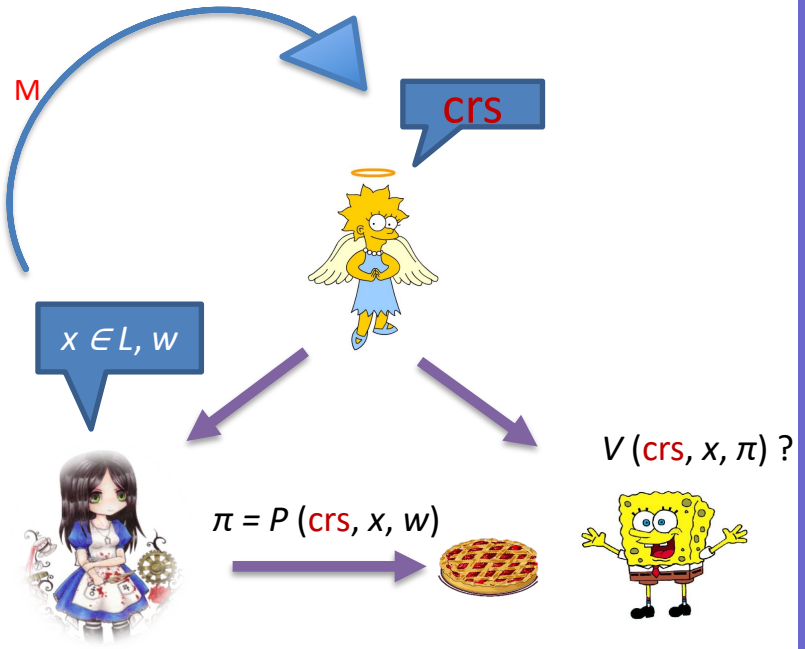
Behzad Abdolmaleki, and Daniel Slamanig

Max Planck Institute for Security and Privacy, Germany
AIT Austrian Institute of Technology, Vienna, Austria

INDOCRYPT 2022

Motivation

- **Quasi-Adaptive NIZK** (QA-NIZK) where the CRS depends to the Language parameter M .
- Such a dependency of the CRS allows one to construct very efficient QA-NIZKs (for linear language) based on standard assumptions.
- QA-NIZK has applications in constructing efficient cryptographic primitives (commitment schemes, IBE, signature schemes, SNARKs compilers, …)



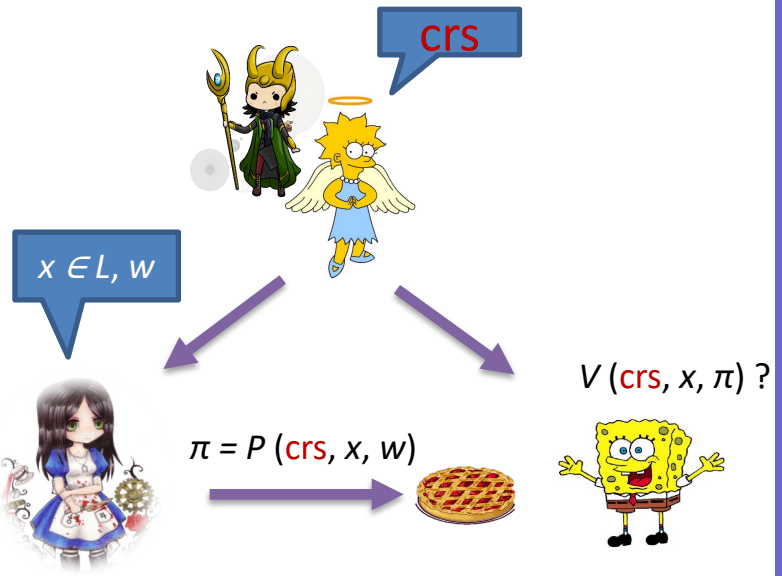
QA-NIZK in CRS model

Motivation

- **Quasi-Adaptive NIZK** (QA-NIZK) where the CRS depends to the Language parameter.

- Such a dependency of the CRS allows one to construct very efficient QA-NIZKs (for linear language) based on standard assumptions.

- QA-NIZK has applications in constructing efficient cryptographic primitives (commitment schemes, IBE, signature schemes, SNARKs compilers, …)



NIZK in CRS model

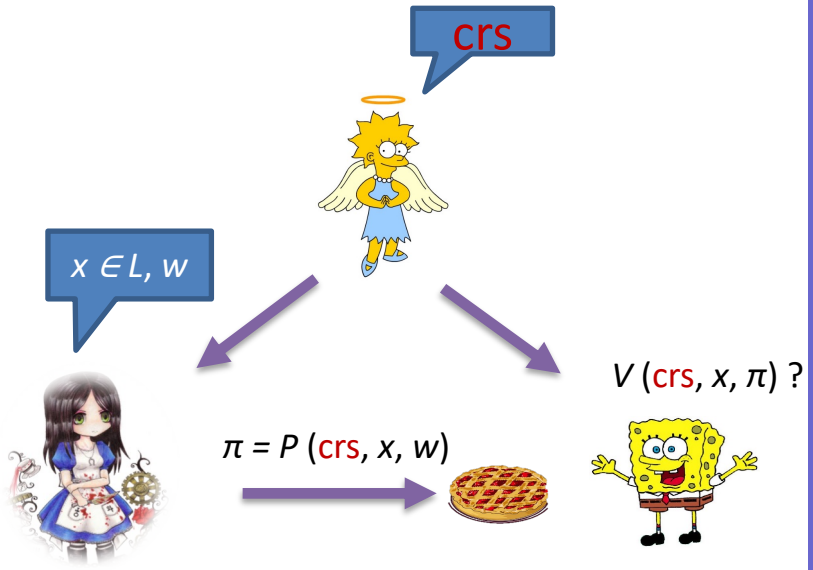
Challenge:

- Such constructions need **a trusted party** to generate the CRS.
  - Is the security guaranteed if the parties do not trust the CRS generator?

Preliminaries:

NIZK in the CRS model

Definition: A NIZK argument system allows the prover to convince the verifier of the validity of some statements and must satisfy the following properties:



NIZK in CRS model

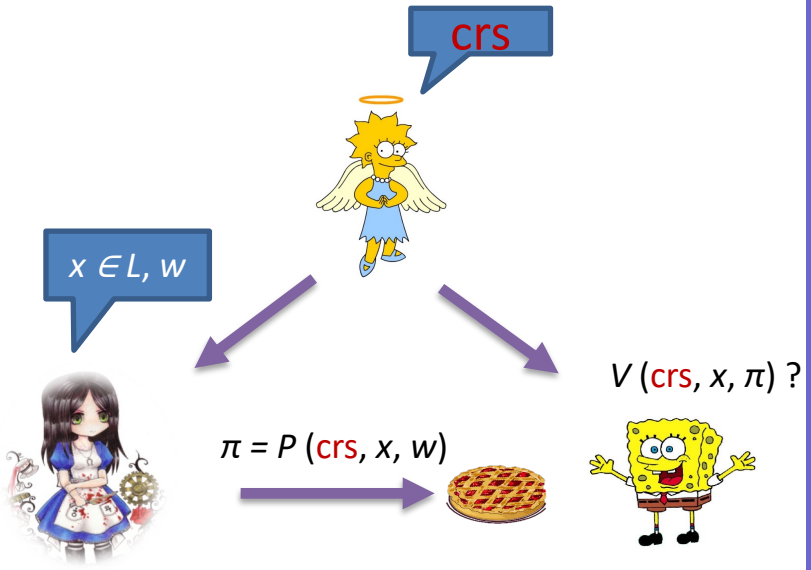
NIZK in the CRS model

Definition: A NIZK argument system allows the prover to convince the verifier of the validity of some statements and must satisfy the following properties:

Completeness: $x \in L \Rightarrow V \text{ accepts } \pi$

Soundness: $x \notin L \Rightarrow V \text{ rejects } \pi$

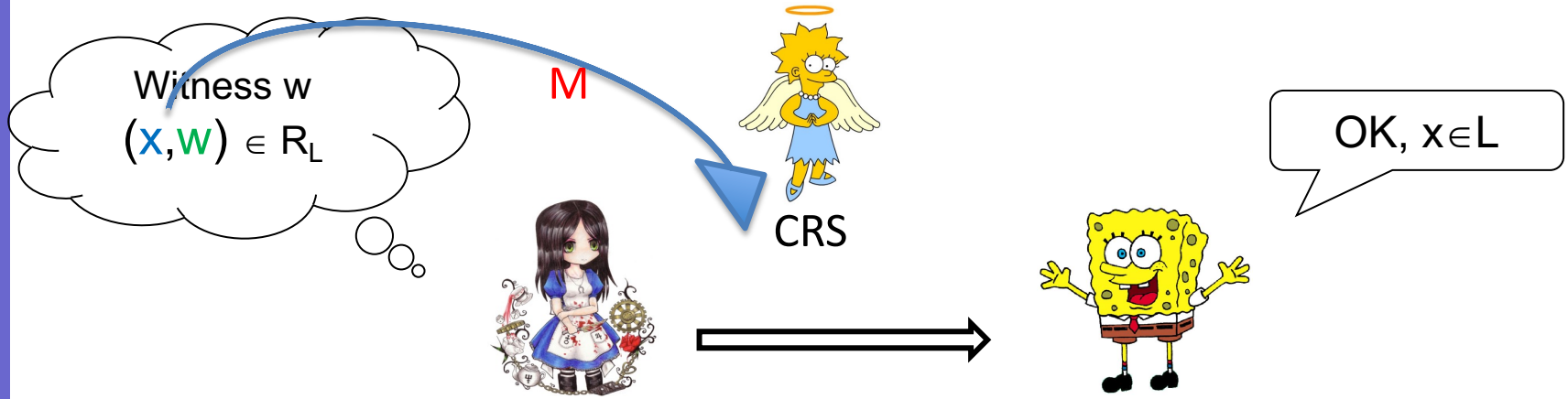
Zero-Knowledge: $\pi \text{ leaks nothing beyond } x \in L$



NIZK in CRS model

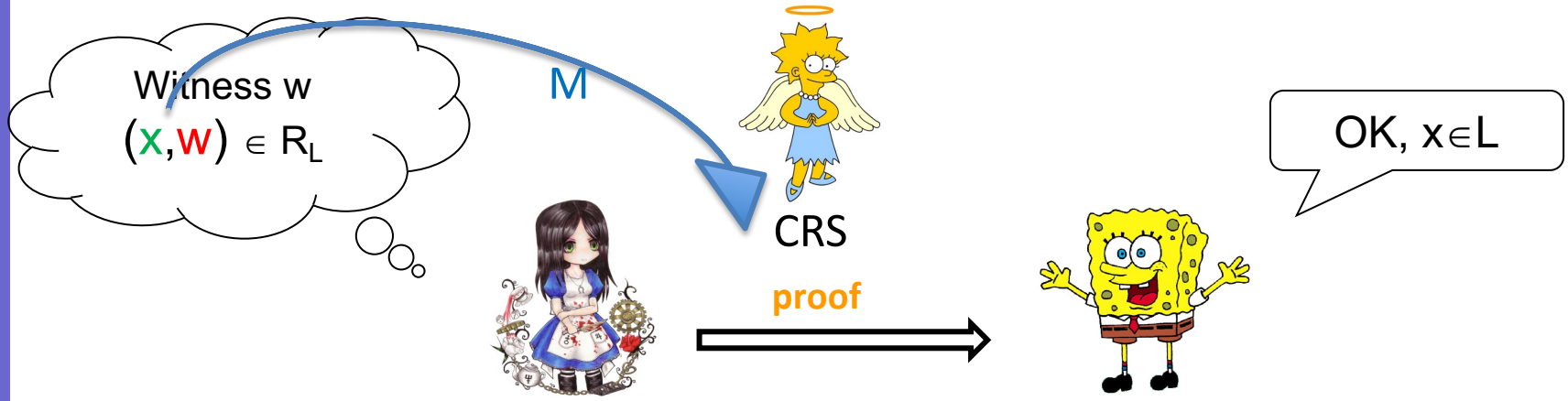
- Quasi-Adaptive NIZK
-Asymmetric Quasi-Adaptive NIZK

Quasi-Adaptive NIZK (QA-NIZK)



- Language parameter M is chosen before the CRS
- M cannot adaptively depend on the CRS
- Usually, M = some public key

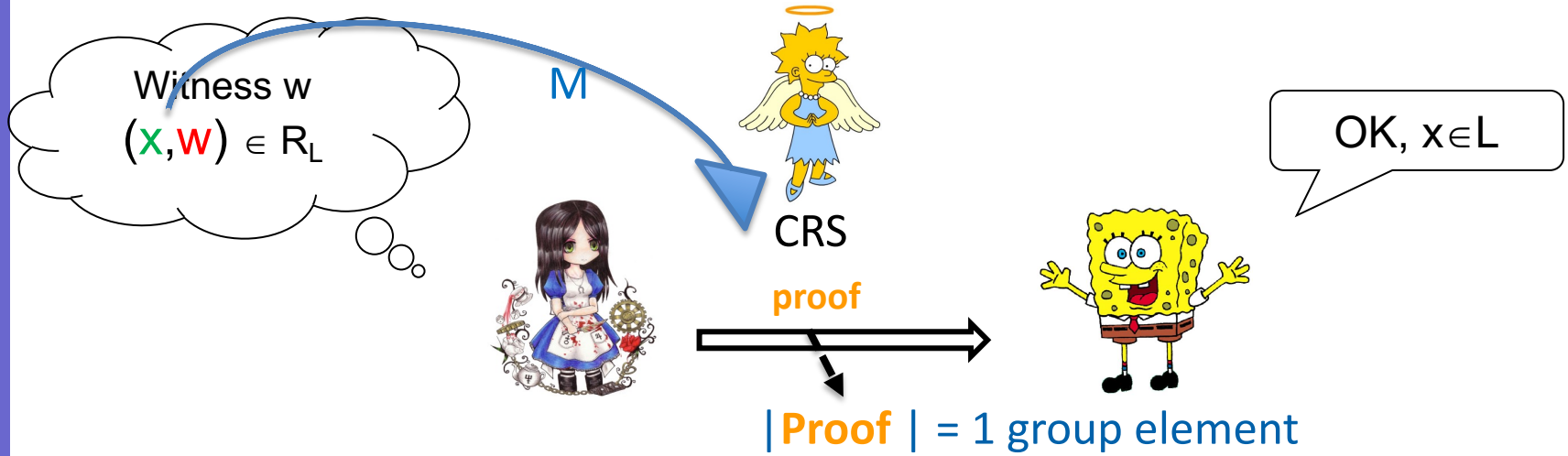
Quasi-Adaptive NIZK (QA-NIZK)



- Language parameter M is chosen before the CRS
- M cannot adaptively depend on the CRS
- Usually, M = some public key

•**Applications** in constructing efficient cryptographic primitives (commitment schemes, IBE, signature schemes, SNARKs compilers, …)

Quasi-Adaptive NIZK (QA-NIZK)



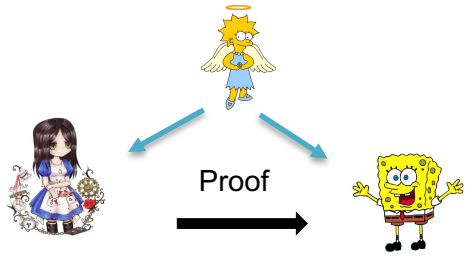
- Language parameter M is chosen before the CRS
- ρ cannot adaptively depend on the CRS
- Usually, M = some public key

•**Applications** in constructing efficient cryptographic primitives (commitment schemes, IBE, signature schemes, SNARKs compilers, …)

[Roy-Julta Asiacrypt2013] ⇨ [**Roy-Jutla** Crypto2014] ⇨ **[Kiltz-Wee Eurocrypt2015]**

Quasi-Adaptive NIZK (QA-NIZK)

Let G_1, G_2, G_T be additive groups of order p
Denote $[a]_i = ag_i$ where g_i is generator of G_i and $a \in \mathbb{Z}_p$
Assume $\cdot : G_1 \times G_2 \to G_T$ is a bilinear map
$[A]_1[B]_2 = [AB]_T$ for compatible matrices A, B

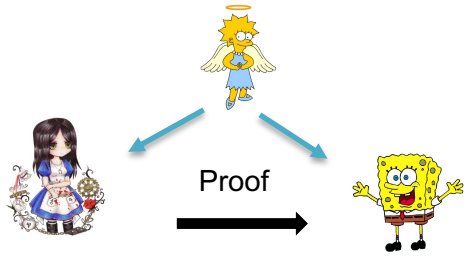


- [Kiltz-Wee Eurocrypt 2015]
  - most efficient known QA-NIZK for SUBSPACE language
- **Task of QA-NIZK for SUBSPACE:**
  - Fix language parameter $[\mathrm{M}]_1 \in G^{n \times m}$
  - Prove in zero knowledge that $[\vec{y}]_1 = [\mathrm{M}]_1 \vec{w}$ for some $\vec{w} \in \mathbb{Z}_p^m$

$$L = \left\{ [\vec{y}]_1 \in G_1^n \mid \exists \vec{w} \in \mathbb{Z}_p^m \ s.t \ [\vec{y}]_1 = [\mathrm{M}]_1 \vec{w} \right\}$$

Asymmetric Quasi-Adaptive NIZK (QA-NIZK)

Let G_1, G_2, G_T be additive groups of order p
Denote $[a]_i = ag_i$ where g_i is generator of G_i and $a \in \mathbb{Z}_p$
Assume $\cdot : G_1 \times G_2 \to G_T$ is a bilinear map
$[A]_1[B]_2 = [AB]_T$ for compatible matrices A, B

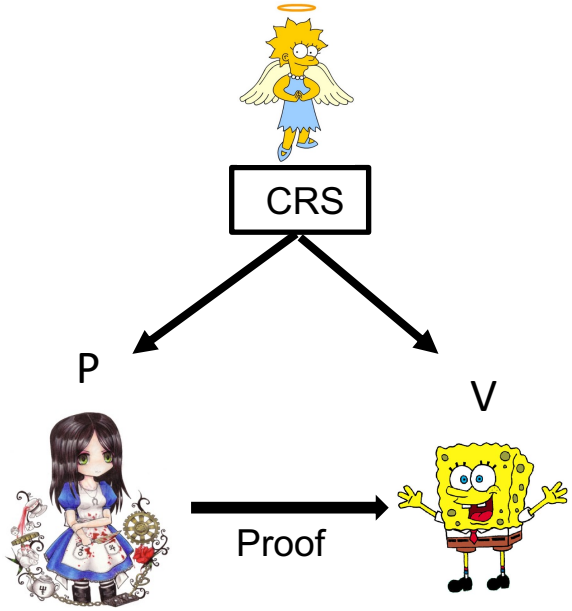


- [González et al. ASIACRYPT 2015]
  - most efficient known asymmetric QA-NIZK for SUBSPACE language

- **Task of Asymmetric QA-NIZK for SUBSPACE:**
  - Fix language parameter $[\mathrm{M}]_1 \in G_1^{n \times m}$ and $[\mathrm{N}]_2 \in G_2^{n \times m}$
  - Prove in zero knowledge that:

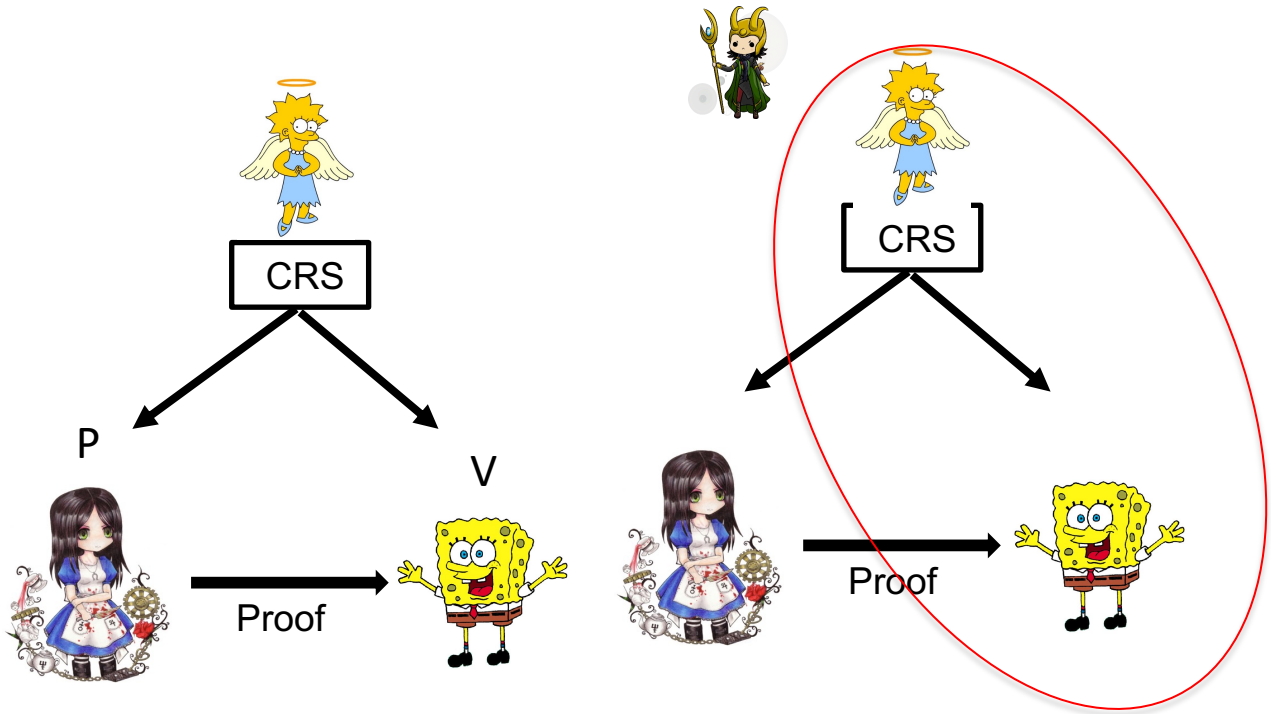
$$L = \left\{ [\vec{y}]_1, [\vec{x}]_2 \in G_1^n \times G_2^n \mid \exists \vec{w} \in \mathbb{Z}_p^m \ s.t \ [\vec{y}]_1 = [\mathrm{M}]_1 \vec{w} \wedge [\vec{x}]_2 = [\mathrm{N}]_2 \vec{w} \right\}$$

NIZKs in Different Subversion Model



The CRS Model
(i.e., [EC:Groth16])

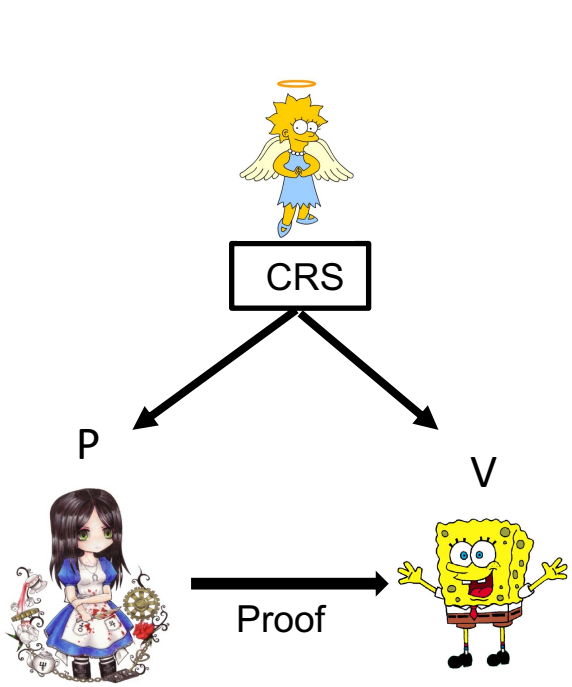
NIZKs in Different Subversion Model

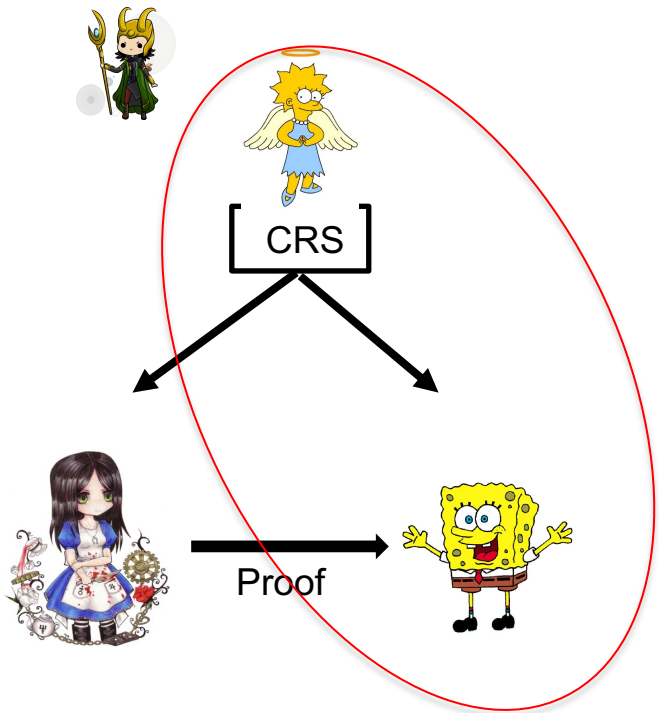


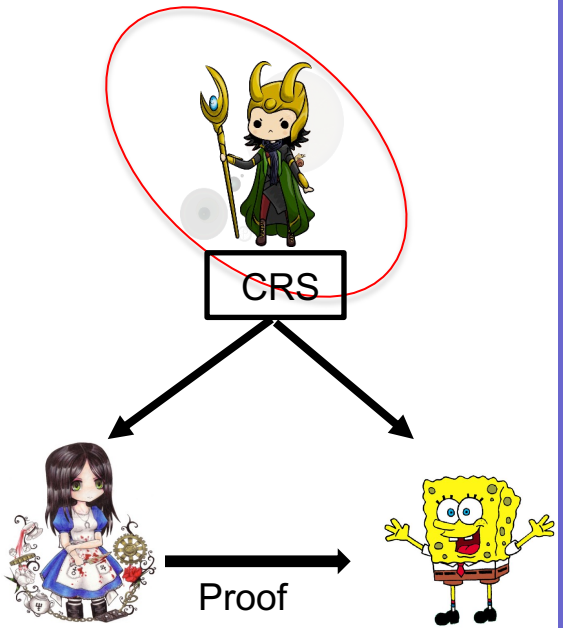
The CRS Model
(i.e., [EC:Groth16])

Subversion Zero-Knowledge model
(i.e., [AC:ABLZ17] [PKC:Fuc18])

NIZKs in Different Subversion Model





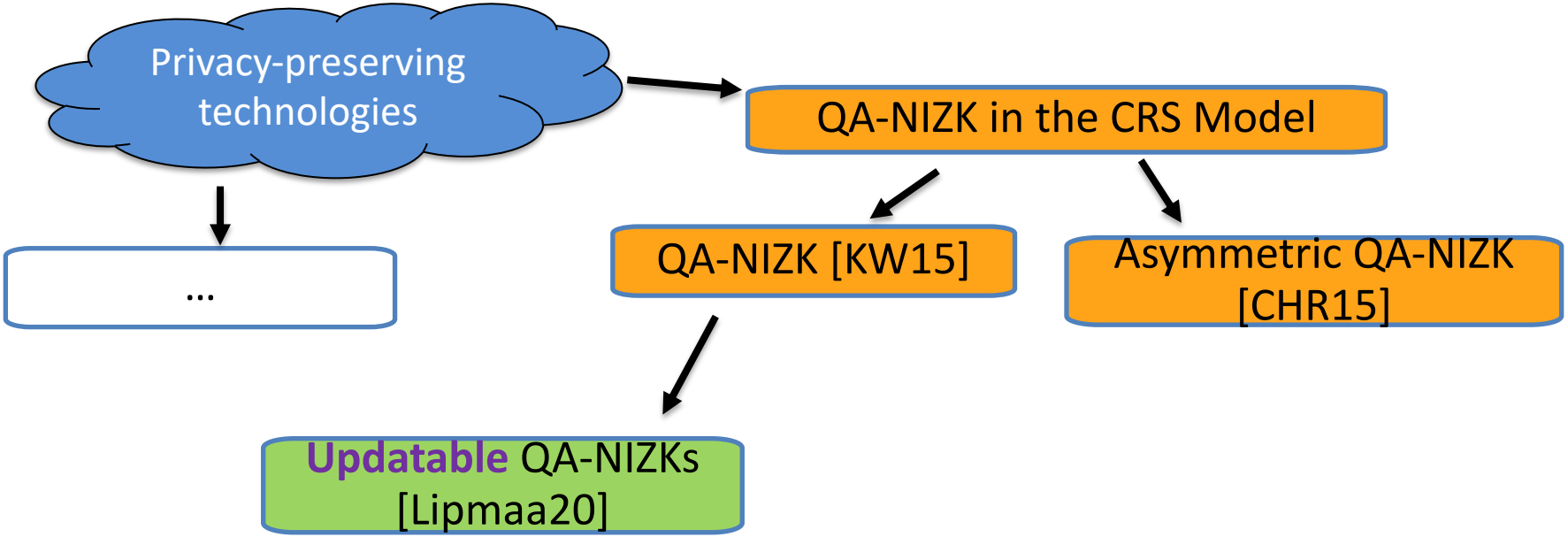


The CRS Model
(i.e., [EC:Groth16])

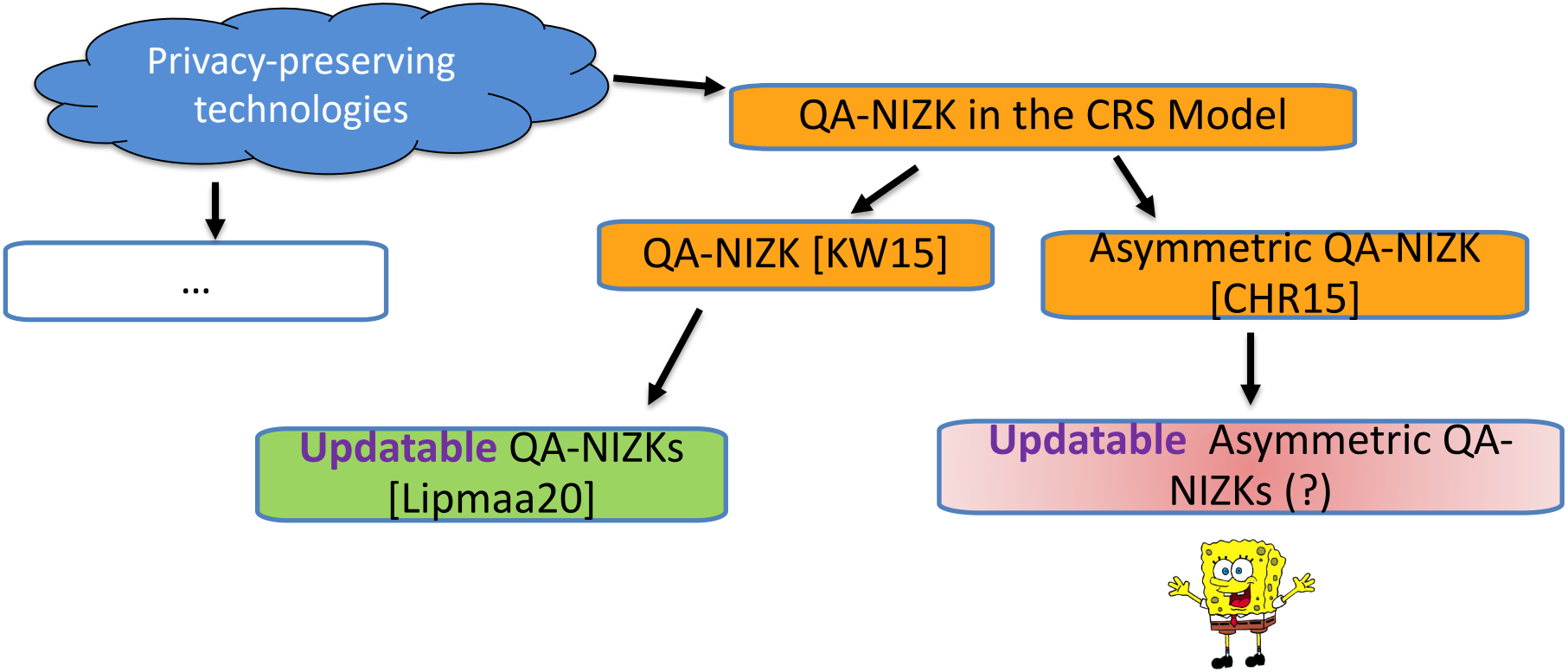
Subversion Zero-Knowledge model
(i.e., [AC:ABLZ17] [PKC:Fuc18])

Updatable model
(i.e., [C:GKM+18]…)

State-of-Art of QANIZK in the Updatable setting



State-of-Art of QANIZK in the Updatable setting



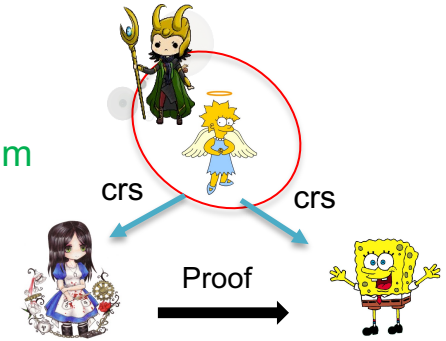
Our Results:

Our main result: Updatable Asymmetric QA-NIZK

- **Zero-knowledge and soundness hold even if CRS creator is not trusted.**

Soundness => verifier does not need to trust CRS – just apply a new Up-crs algorithm and update the CRS to CRS'.

ZK => prover does not need to trust CRS – just apply a new Vcrs algorithm.

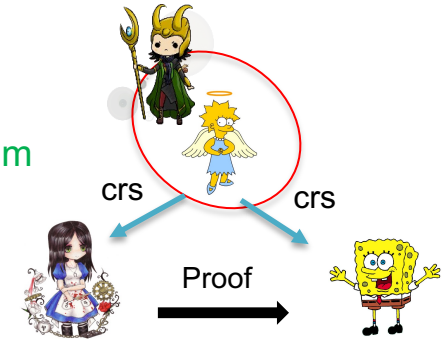


Our main result: Updatable Asymmetric QA-NIZK

- **Zero-knowledge and soundness hold even if CRS creator is not trusted.**

Soundness => verifier does not need to trust CRS – just apply a new Up-crs algorithm and update the CRS to CRS'.

ZK => prover does not need to trust CRS – just apply a new Vcrs algorithm.

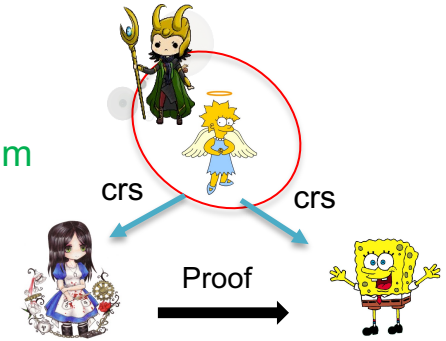


Our recipe:

GHR15 asymmetric QA-NIZK in the CRS model.

Our main result: Updatable Asymmetric QA-NIZK

- **Zero-knowledge and soundness hold even if CRS creator is not trusted.**



Soundness => verifier does not need to trust CRS – just apply a new Up-crs algorithm and update the CRS to CRS'.

ZK => prover does not need to trust CRS – just apply a new Vcrs algorithm.

Our recipe:

GHR15 asymmetric QA-NIZK in the CRS model.

Updatable Soundness

- Making CRS Updatable
  Design a new algorithm Up-crs for updating the CRS to CRS':

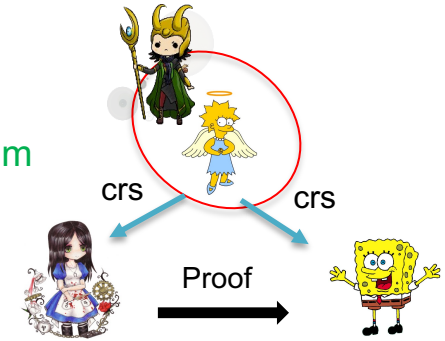
  (CRS', crs-Proof) ← Up-crs($[M]_1$,$[N]_2$, CRS)

Our main result: Updatable Asymmetric QA-NIZK

- **Zero-knowledge and soundness hold even if CRS creator is not trusted.**

Soundness => verifier does not need to trust CRS – just apply a new Up-crs algorithm and update the CRS to CRS'.

ZK => prover does not need to trust CRS – just apply a new Vcrs algorithm.



Our recipe:

GHR15 asymmetric QA-NIZK in the CRS model.

Updatable Soundness

Updatable ZK

- Making CRS Updatable
  Design a new algorithm Up-crs for updating the CRS to CRS':

  (CRS', crs-Proof) ← Up-crs($[M]_1$,$[N]_2$, CRS)

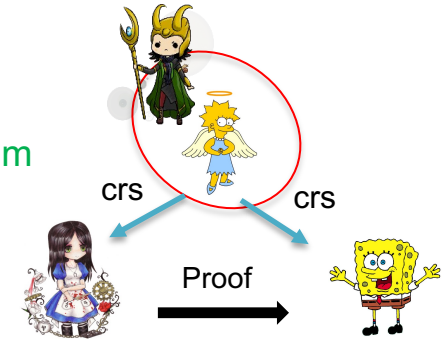
- Making CRS publicly verifiable
  Design a **public** algorithm Vcrs for checking CRS' is correct
  - If Vcrs($[M]_1$,$[N]_2$, CRS′, CRS) = 1: there exists **some trapdoor** tc

Our main result: Updatable Asymmetric QA-NIZK

- **Zero-knowledge and soundness hold even if CRS creator is not trusted.**

Soundness => verifier does not need to trust CRS – just apply a new Up-crs algorithm and update the CRS to CRS'.

ZK => prover does not need to trust CRS – just apply a new Vcrs algorithm.



Our recipe:

GHR15 asymmetric QA-NIZK in the CRS model.

Updatable Soundness

Updatable ZK

- Making CRS Updatable
  Design a new algorithm Up-crs for updating the CRS to CRS':

  (CRS', crs-Proof) ← Up-crs($[M]_1$,$[N]_2$, CRS)

- Making CRS publicly verifiable
  Design a **public** algorithm Vcrs for checking CRS' is correct
  - If Vcrs($[M]_1$,$[N]_2$, CRS', CRS) = 1: there exists **some trapdoor** tc

Proving updatable ZK

If Vcrs($[M]_1$,$[N]_2$, CRS, CRS') = 0: no need to simulate

If Vcrs($[M]_1$,$[N]_2$, CRS, CRS') = 1:

Use extractor Ext to recover tc from CRS' by BDH-KE assumption.

Other results

- Knowledge Sound version of Asymmetric QA-NIZK CHR15 (ASIACRYPT'15) and the updatable Asymmetric QA-NIZK.

Other results

- Knowledge Sound version of Asymmetric QA-NIZK CHR15 (ASIACRYPT'15) and the updatable Asymmetric QA-NIZK.

- How to integrate our updatable Knowledge Sound QA-NIZKs (and also Knowledge Sound version of Asymmetric QA-NIZK CHR15) into the LegoSNARK toolbox.
  Our results together with existing results on updatable zk-SNARKS represent an important step towards an updatable variant of the LegoSNARK toolbox (with the extension by the proposed updatable Knowledge Sound QA-NIZKs)

Other results

- Knowledge Sound version of Asymmetric QA-NIZK CHR15 (ASIACRYPT'15) and the updatable Asymmetric QA-NIZK.

- How to integrate our updatable Knowledge Sound QA-NIZKs (and also Knowledge Sound version of Asymmetric QA-NIZK CHR15) into the LegoSNARK toolbox.
  Our results together with existing results on updatable zk-SNARKS represent an important step towards an updatable variant of the LegoSNARK toolbox (with the extension by the proposed updatable Knowledge Sound QA-NIZKs)

Open Problems:

- (Sub-ZK) QA-NIZK with Simulation-Sound Extractability


MAX-PLANCK-GESELLSCHAFT


AUSTRIAN INSTITUTE
OF TECHNOLOGY
TOMORROW TODAY

Thank you