Succinct Malleable NIZKs and an Application to Compact Shuffles

Melissa Chase (MSR Redmond)
Markulf Kohlweiss (MSR Cambridge)
Anna Lysyanskaya (Brown University)
Sarah Meiklejohn (UC San Diego)

Proofs of proofs

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

π_1 π_2

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

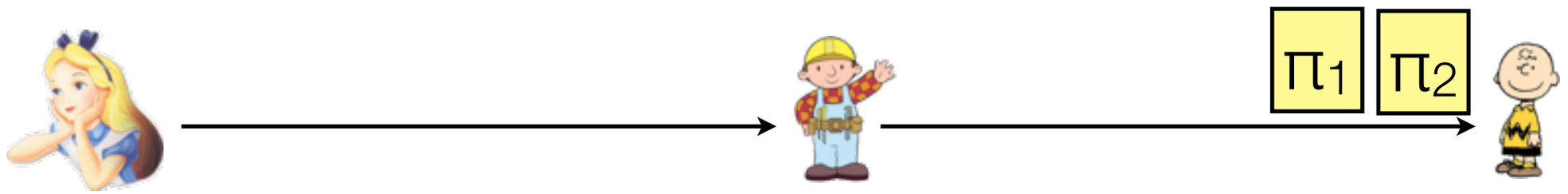
Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

Proofs of proofs

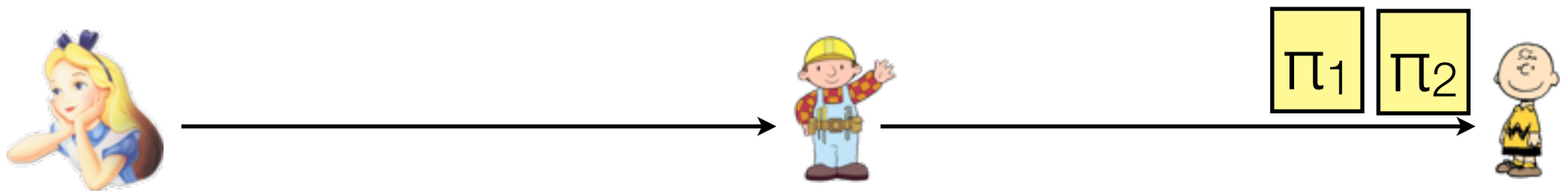
Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit



To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

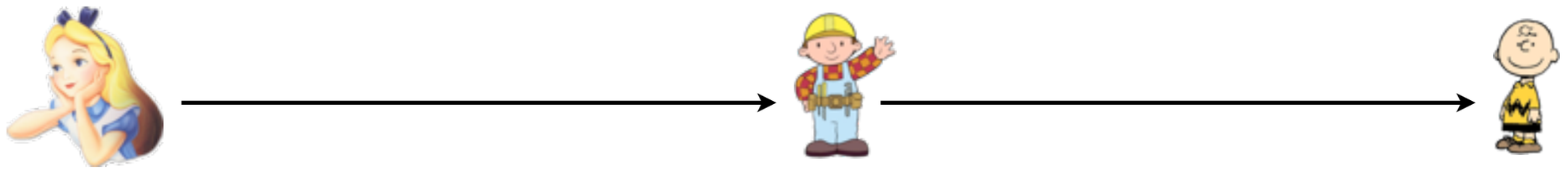


To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

But this reveals π_1 and π_2; Charlie could know Alice formed proofs!

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit

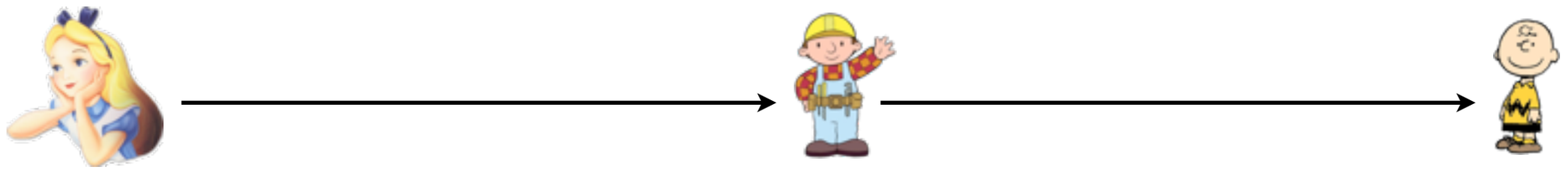


To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

But this reveals π_1 and π_2; Charlie could know Alice formed proofs!

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit



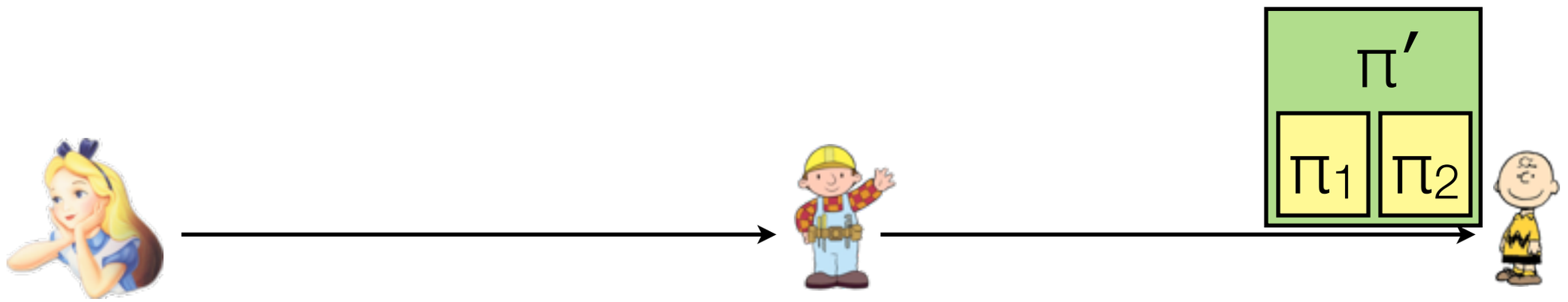
To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

But this reveals π_1 and π_2; Charlie could know Alice formed proofs!

Next solution: prove knowledge of π_1 and π_2 (“meta-proof” [dSY90])

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit



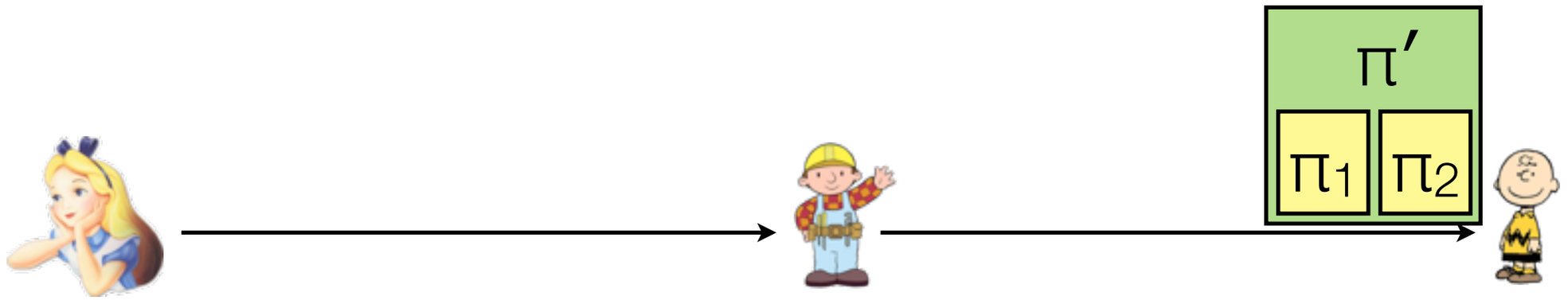
To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

But this reveals π_1 and π_2; Charlie could know Alice formed proofs!

Next solution: prove knowledge of π_1 and π_2 (“meta-proof” [dSY90])

Proofs of proofs

Suppose Alice gives Bob a proof π_1 that an encrypted value b_1 is a bit (0 or 1), and a proof π_2 that another encrypted value b_2 is a bit



To prove $b_1 \cdot b_2$ is a bit: just pass Charlie π_1 and π_2

But this reveals π_1 and π_2; Charlie could know Alice formed proofs!

Next solution: prove knowledge of π_1 and π_2 (“meta-proof” [dSY90])

But this proof is big; reveals that Bob didn’t form original proofs!

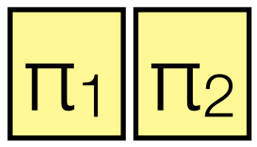
SNARGs and malleable proofs

SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!

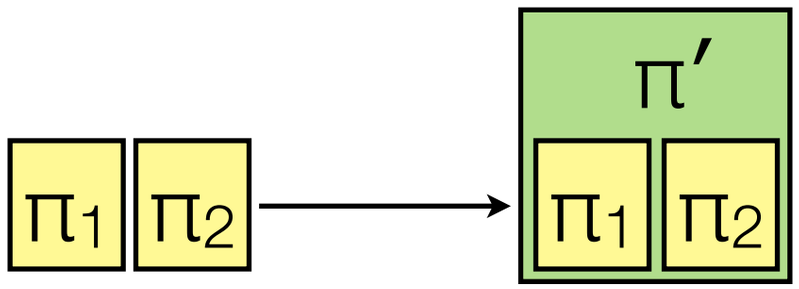
SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!



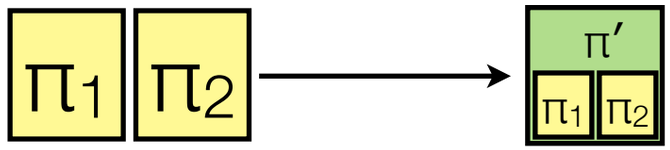
SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!



SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!



SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!

But what is π' even proving? What Bob really wants is a malleable proof: take proofs π_1 for b_1 and π_2 for b_2 and "maul" them to form a proof for $b_1 \cdot b_2$

SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!

But what is π' even proving? What Bob really wants is a malleable proof: take proofs π_1 for b_1 and π_2 for b_2 and "maul" them to form a proof for $b_1 \cdot b_2$

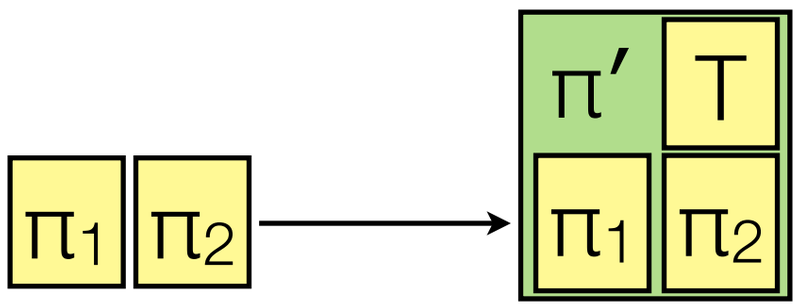
Then if he proves knowledge of π_1 and π_2, but also of a transformation T such that $b_1 \cdot b_2 = T(b_1,b_2)$, does this suffice as a proof for $b_1 \cdot b_2$?

SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!

But what is π' even proving? What Bob really wants is a malleable proof: take proofs π_1 for b_1 and π_2 for b_2 and "maul" them to form a proof for $b_1 \cdot b_2$

Then if he proves knowledge of π_1 and π_2, but also of a transformation T such that $b_1 \cdot b_2 = T(b_1,b_2)$, does this suffice as a proof for $b_1 \cdot b_2$?

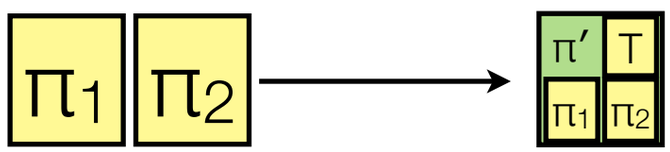


SNARGs and malleable proofs

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π_1 and π_2 could in fact be the same size!

But what is π' even proving? What Bob really wants is a malleable proof: take proofs π_1 for b_1 and π_2 for b_2 and "maul" them to form a proof for $b_1 \cdot b_2$

Then if he proves knowledge of π_1 and π_2, but also of a transformation T such that $b_1 \cdot b_2 = T(b_1,b_2)$, does this suffice as a proof for $b_1 \cdot b_2$?



Why use SNARGs for malleable proofs?

Why use SNARGs for malleable proofs?

At Eurocrypt 2012 [CKLM12], we defined notions of malleability and controlled malleability for proofs; called them cm-NIZKs

Why use SNARGs for malleable proofs?

At Eurocrypt 2012 [CKLM12], we defined notions of malleability and controlled malleability for proofs; called them cm-NIZKs

To actually achieve malleability, our construction was fundamentally based on Groth-Sahai proofs [GS08]

Why use SNARGs for malleable proofs?

At Eurocrypt 2012 [CKLM12], we defined notions of malleability and controlled malleability for proofs; called them cm-NIZKs

To actually achieve malleability, our construction was fundamentally based on Groth-Sahai proofs [GS08]

Essentially observed certain malleability properties and built off of those; restricted to transformations supported by GS proofs

Why use SNARGs for malleable proofs?

At Eurocrypt 2012 [CKLM12], we defined notions of malleability and controlled malleability for proofs; called them cm-NIZKs

To actually achieve malleability, our construction was fundamentally based on Groth-Sahai proofs [GS08]

Essentially observed certain malleability properties and built off of those; restricted to transformations supported by GS proofs

Natural open question: can we build malleability ourselves? If so, what kind of malleability can we hope to achieve?

Why use SNARGs for malleable proofs?

At Eurocrypt 2012 [CKLM12], we defined notions of malleability and controlled malleability for proofs; called them cm-NIZKs

To actually achieve malleability, our construction was fundamentally based on Groth-Sahai proofs [GS08]

Essentially observed certain malleability properties and built off of those; restricted to transformations supported by GS proofs

Natural open question: can we build malleability ourselves? If so, what kind of malleability can we hope to achieve?

This would potentially allow for more applications (e.g., CM-CCA encryption)

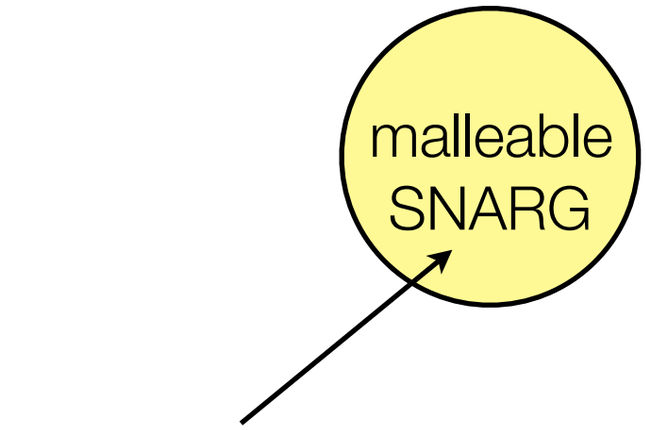
Our contributions

Our contributions

To get all the way from a SNARG to a cm-NIZK, proceed in three stages

Our contributions

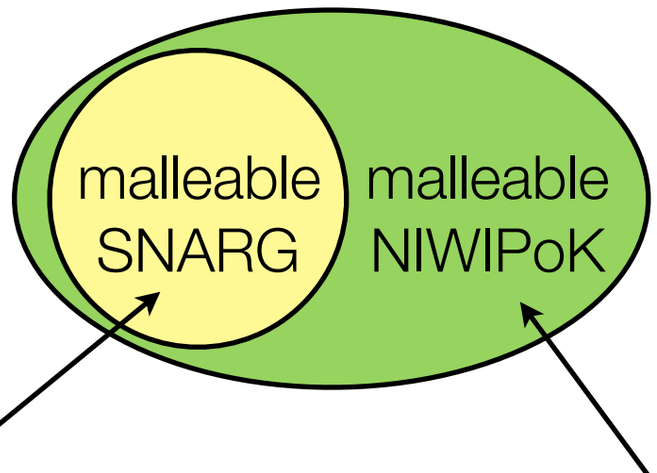
To get all the way from a SNARG to a cm-NIZK, proceed in three stages



First, built malleability
into SNARGs
according to our intuition

Our contributions

To get all the way from a SNARG to a cm-NIZK, proceed in three stages

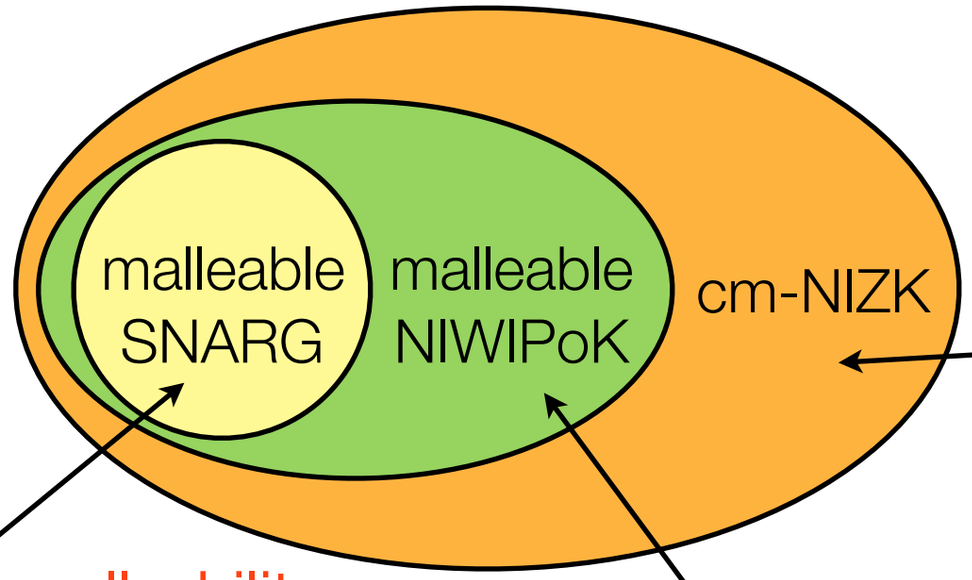


First, built malleability
into SNARGs
according to our intuition

Next, boost from weird SNARG
extraction to regular extractability

Our contributions

To get all the way from a SNARG to a cm-NIZK, proceed in three stages



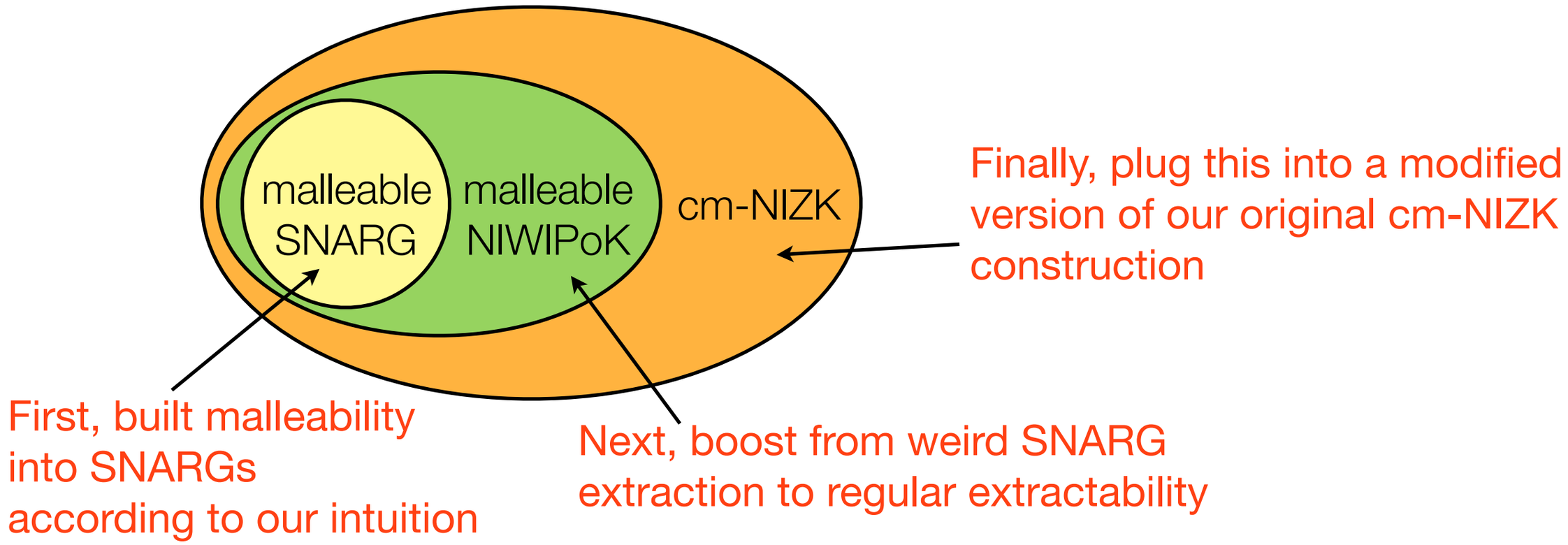
Finally, plug this into a modified version of our original cm-NIZK construction

First, built malleability into SNARGs according to our intuition

Next, boost from weird SNARG extraction to regular extractability

Our contributions

To get all the way from a SNARG to a cm-NIZK, proceed in three stages



The end result? A fully generic cm-NIZK with a much wider range of malleability (all t-tiered transformations) than previously supported, that is easier to “plug in” to applications

Outline

Outline

Definitions

Outline

Definitions

SNARGs to cm-NIZKs

Outline

Definitions

SNARGs to cm-NIZKs

Applying the cm-NIZK

Outline

Definitions

SNARGs to cm-NIZKs

Applying the cm-NIZK

Conclusions

Outline

Definitions
Malleable proofs
SNARGs
t-tiered transformations

SNARGs to cm-NIZKs

Applying the cm-NIZK

Conclusions

Malleability for proofs [CKLM12]

Malleability for proofs [CKLM12]

Generally, a proof is malleable with respect to T if there exists an algorithm Eval that on input $(T,\{x_i,\pi_i\})$, outputs a proof π for $T(\{x_i\})$

Malleability for proofs [CKLM12]

Generally, a proof is malleable with respect to T if there exists an algorithm Eval that on input $(T,\{x_i,\pi_i\})$, outputs a proof π for $T(\{x_i\})$

- E.g., T = ×, x_i = "b_i is a bit"

Malleability for proofs [CKLM12]

Generally, a proof is malleable with respect to T if there exists an algorithm Eval that on input $(T,\{x_i,\pi_i\})$, outputs a proof π for $T(\{x_i\})$

- E.g., $T = \times$, $x_i =$ "b_i is a bit"

Can define zero knowledge in the usual way as long as proofs are malleable only with respect to operations under which the language is closed

Malleability for proofs [CKLM12]

Generally, a proof is malleable with respect to T if there exists an algorithm Eval that on input $(T,\{x_i,\pi_i\})$, outputs a proof π for $T(\{x_i\})$

- E.g., $T = \times$, $x_i =$ "b_i is a bit"

Can define zero knowledge in the usual way as long as proofs are malleable only with respect to operations under which the language is closed

But how to define a strong notion of soundness like controlled malleability?

Malleability for proofs [CKLM12]

Generally, a proof is malleable with respect to T if there exists an algorithm Eval that on input $(T,\{x_i,\pi_i\})$, outputs a proof π for $T(\{x_i\})$

- E.g., $T = \times$, x_i = "b_i is a bit"

Can define zero knowledge in the usual way as long as proofs are malleable only with respect to operations under which the language is closed

But how to define a strong notion of soundness like controlled malleability?

High-level idea of CM-SSE: extractor can pull out either a witness (fresh proof), or a previous instance and an allowable transformation from that instance to the new one (validly transformed proof)

Malleability for proofs [CKLM12]

Generally, a proof is malleable with respect to T if there exists an algorithm Eval that on input $(T,\{x_i,\pi_i\})$, outputs a proof π for $T(\{x_i\})$

- E.g., $T = \times$, x_i = "b_i is a bit"

Can define zero knowledge in the usual way as long as proofs are malleable only with respect to operations under which the language is closed

But how to define a strong notion of soundness like controlled malleability?

High-level idea of CM-SSE: extractor can pull out either a witness (fresh proof), or a previous instance and an allowable transformation from that instance to the new one (validly transformed proof)

(hides fresh vs. transformed)

If a proof is zero knowledge, CM-SSE, and strongly derivation private, then we call it a cm-NIZK

SNARGs [BSW12,GGPR13]

SNARGs [BSW12,GGPR13]

A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

SNARGs [BSW12,GGPR13]

A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

- (Succinctness.) The size of a proof that $(x,w) \in R$ is bounded by $\varphi(k,|x|,|w|) < \text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ for some $0 < \gamma < 1$

SNARGs [BSW12,GGPR13]

A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

- (Succinctness.) The size of a proof that $(x,w) \in R$ is bounded by $\varphi(k,|x|,|w|) < \text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ for some $0 < \gamma < 1$
  - We use $\gamma = 1/4$ (for unary case)

SNARGs [BSW12,GGPR13]

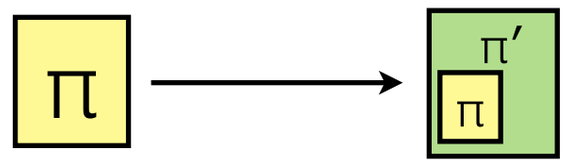
A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

- (Succinctness.) The size of a proof that $(x,w) \in R$ is bounded by $\varphi(k,|x|,|w|) < \text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ for some $0 < \gamma < 1$
- We use $\gamma = 1/4$ (for unary case)
- The point is, the proof can be smaller than the witness

SNARGs [BSW12,GGPR13]

A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

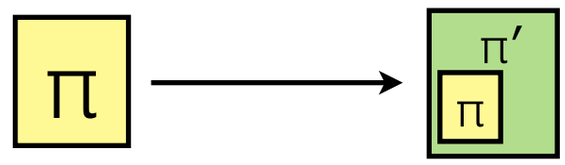
- (Succinctness.) The size of a proof that $(x,w) \in R$ is bounded by $\varphi(k,|x|,|w|) < \text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ for some $0 < \gamma < 1$
- We use $\gamma = 1/4$ (for unary case)
- The point is, the proof can be smaller than the witness



SNARGs [BSW12,GGPR13]

A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

- (Succinctness.) The size of a proof that $(x,w)\in R$ is bounded by $\varphi(k,|x|,|w|) < \text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ for some $0 < \gamma < 1$
  - We use $\gamma = 1/4$ (for unary case)



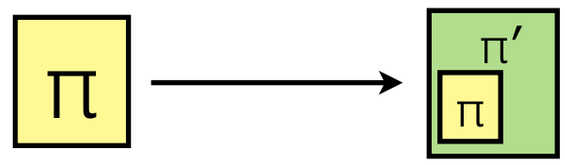
  - The point is, the proof can be smaller than the witness

- (Adaptive knowledge extraction.) For every A there exists extractor E_A such that, for $(x,\pi) = A(crs;r)$, $w = E_A(crs;r)$ such that $(x,w)\in R$

SNARGs [BSW12,GGPR13]

A proof system is a succinct non-interactive argument of knowledge (SNARG) if it is complete and if:

- (Succinctness.) The size of a proof that $(x,w)\in R$ is bounded by $\varphi(k,|x|,|w|) < \text{poly}(k)\text{polylog}(|x|) + \gamma|w|$ for some $0 < \gamma < 1$
  - We use $\gamma = 1/4$ (for unary case)



  - The point is, the proof can be smaller than the witness

- (Adaptive knowledge extraction.) For every A there exists extractor E_A such that, for $(x,\pi) = A(crs;r)$, $w = E_A(crs;r)$ such that $(x,w)\in R$

Constructions of these do exist [AF07,Groth10,...,BCCT12,GGPR13]

t-tiered transformations

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

t-tiered transformations

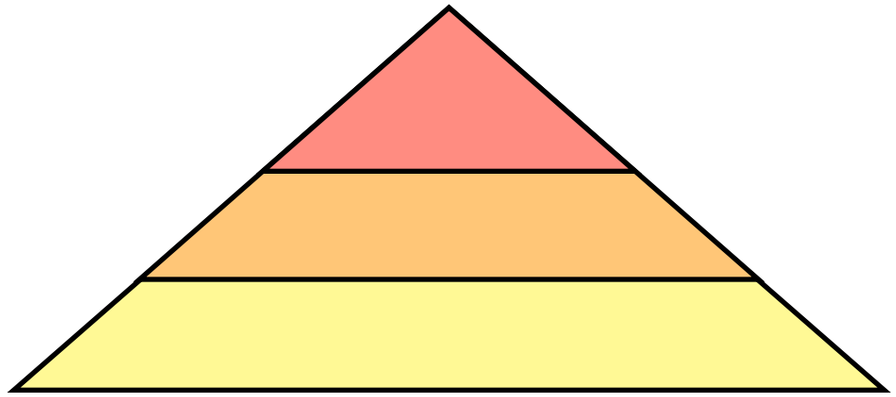
To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

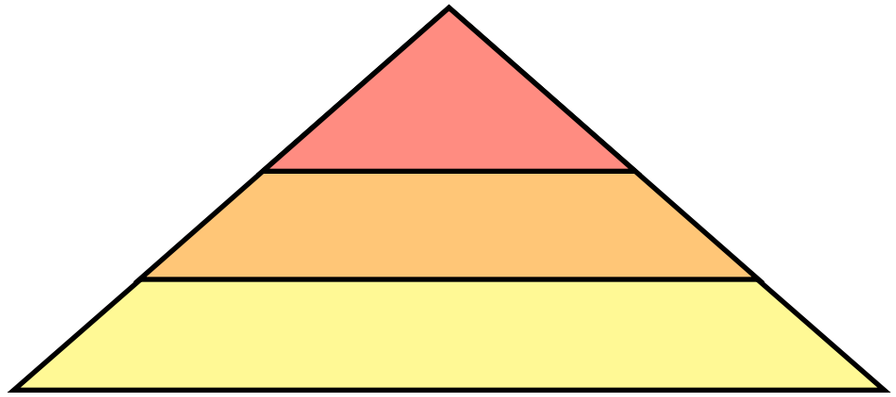
A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$



t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$

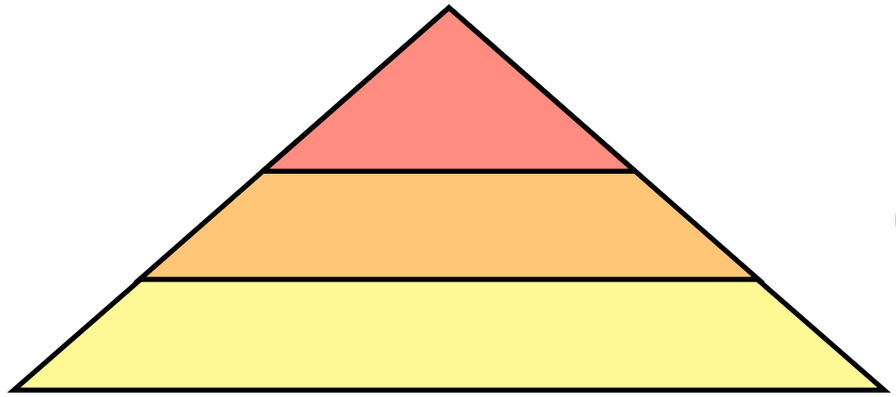


A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then tier(T(x)) > tier(x) and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$



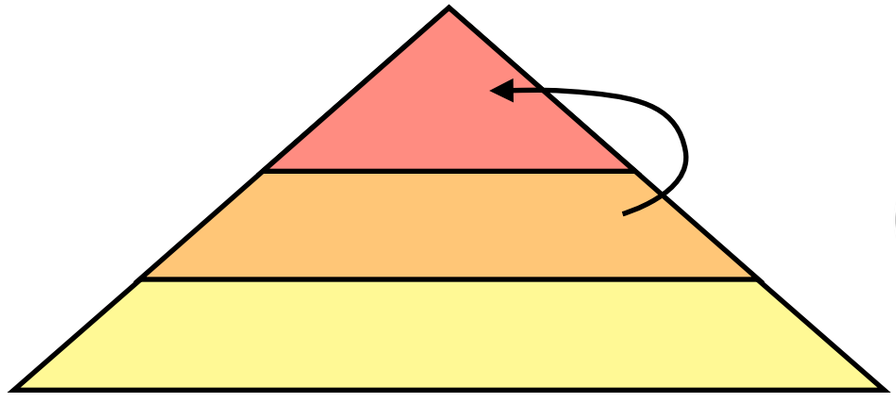
(allowed)

A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then $\text{tier}(T(x)) > \text{tier}(x)$ and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$



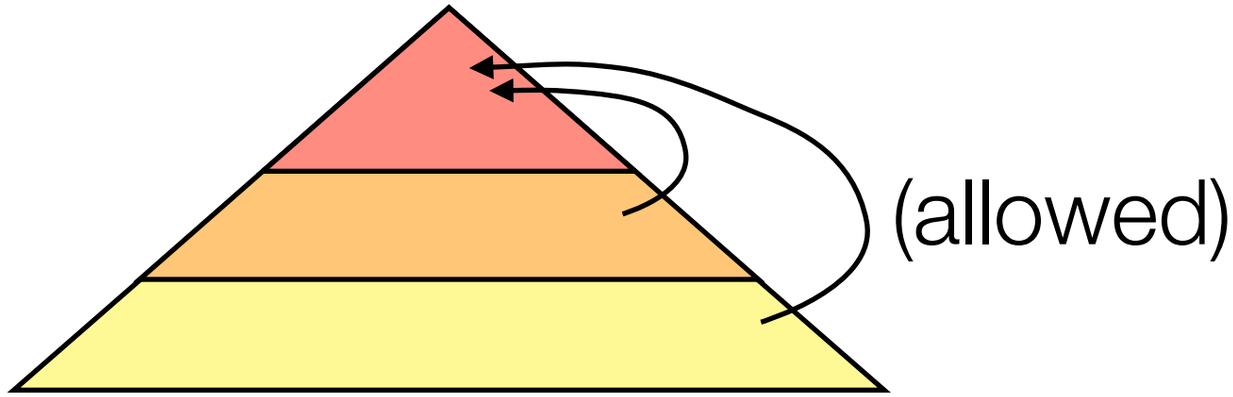
(allowed)

A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then tier(T(x)) > tier(x) and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$

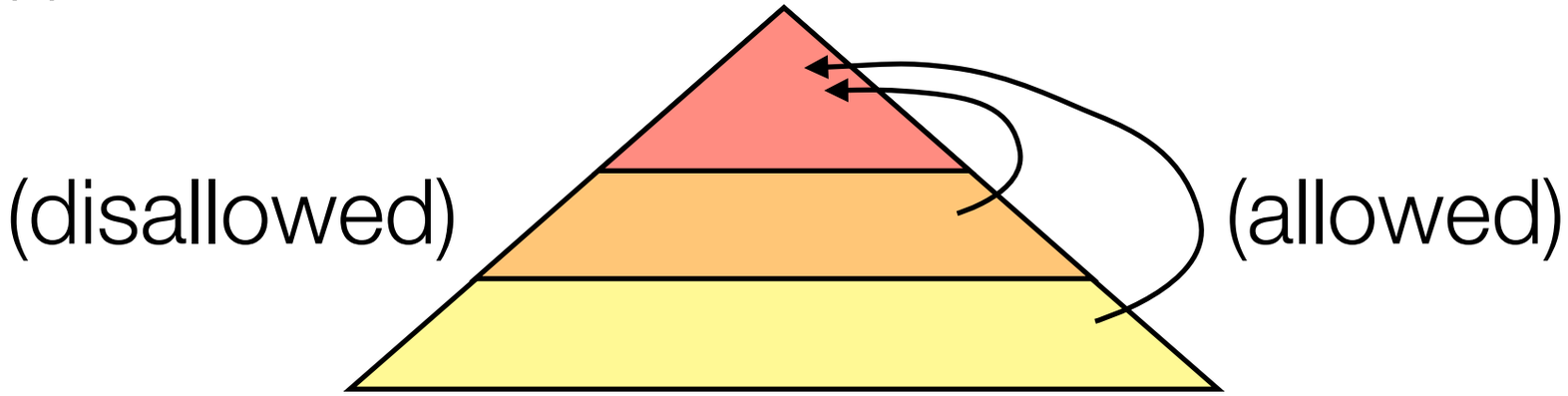


A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then tier(T(x)) > tier(x) and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$

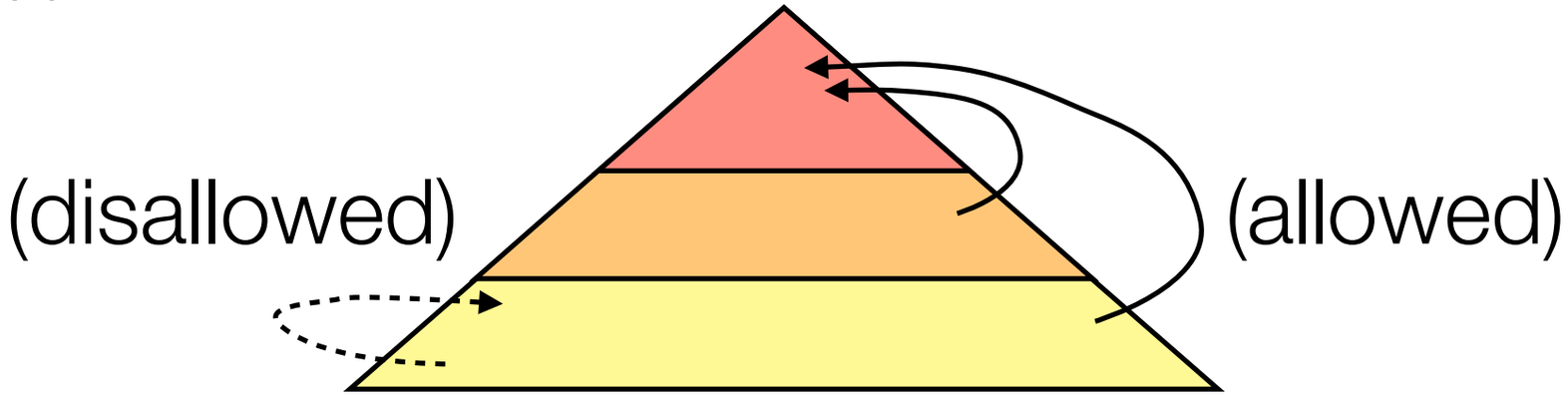


A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then tier(T(x)) > tier(x) and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$

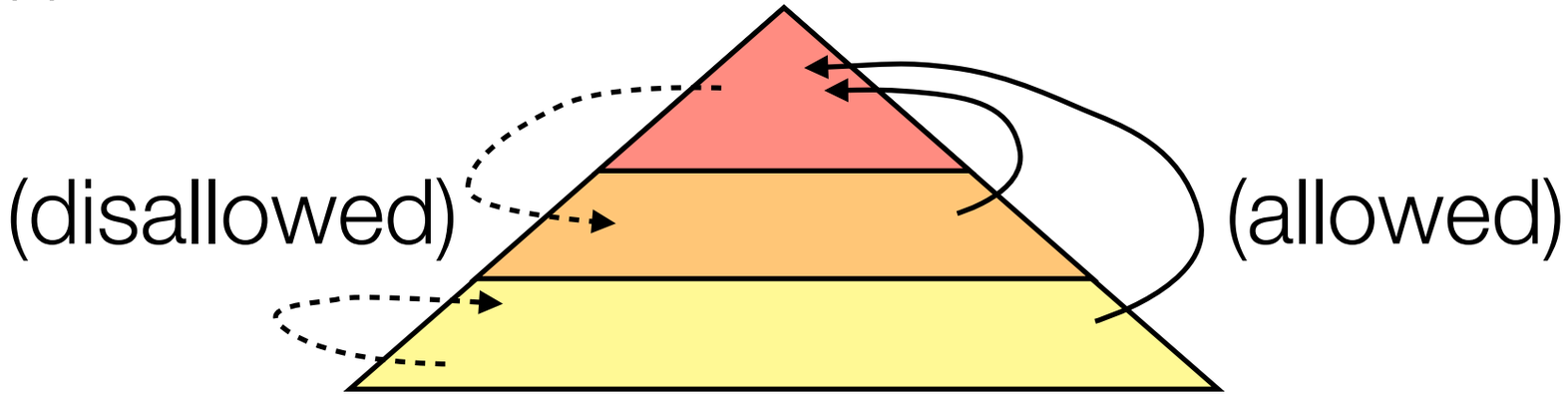


A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then tier(T(x)) > tier(x) and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$

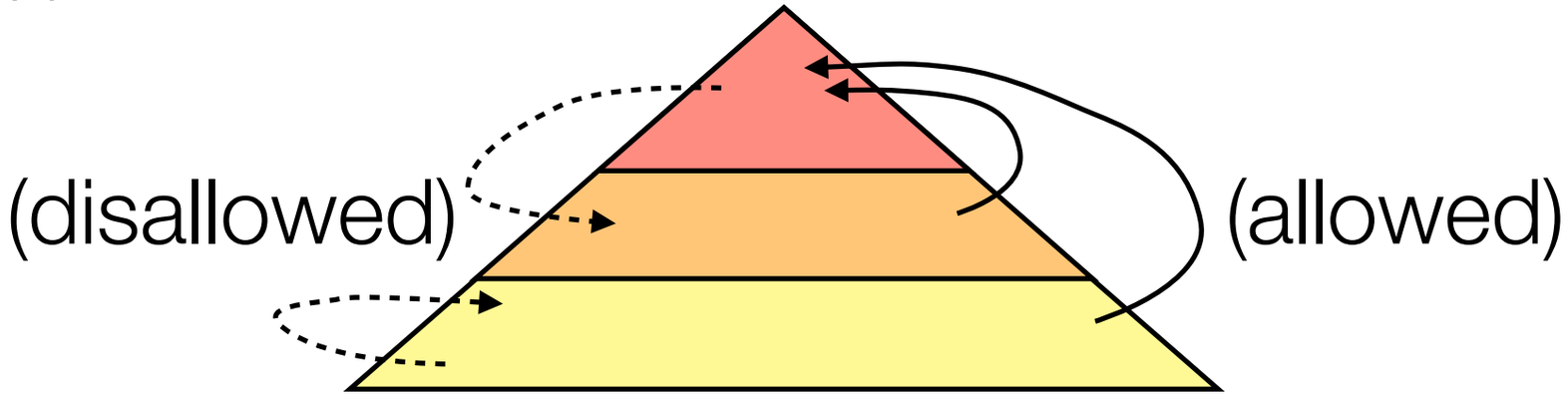


A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) tier(x) < t and $x \in L_R$ then tier(T(x)) > tier(x) and $T(x) \in L_R$, and (2) if tier(x) = t then $T(x) = \perp$

t-tiered transformations

To fit the proof-of-a-proof approach, consider transformations as moving between tiers

A relation R is t-tiered if there exists an efficient function tier(·) such that for all $x \in L_R$, $0 \leq \text{tier}(x) \leq t$



A class of transformations $\mathcal{T}$ is t-tiered if for all $T \in \mathcal{T}$, (1) $\text{tier}(x) < t$ and $x \in L_R$ then $\text{tier}(T(x)) > \text{tier}(x)$ and $T(x) \in L_R$, and (2) if $\text{tier}(x) = t$ then $T(x) = \perp$

Also can't compose more than t transformations

Outline

Definitions

SNARGs to cm-NIZKs
Malleable SNARGs
Boosting to full extractability
Boosting to CM-SSE

Applying the cm-NIZK

Conclusions

Malleable SNARGs

Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!

Malleable SNARGs

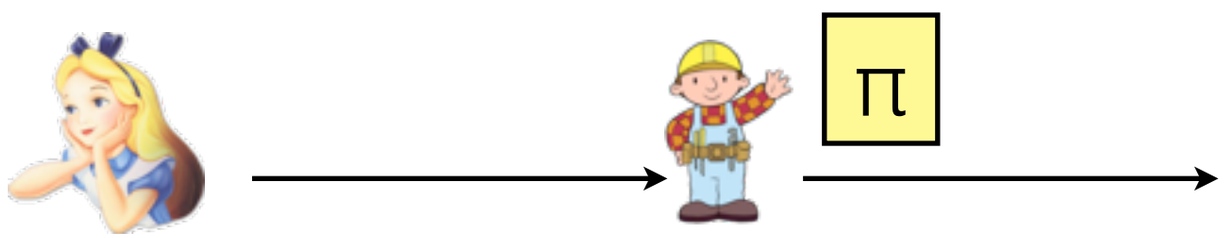
Our goal: build malleability into SNARGs [BSW12]

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!

Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

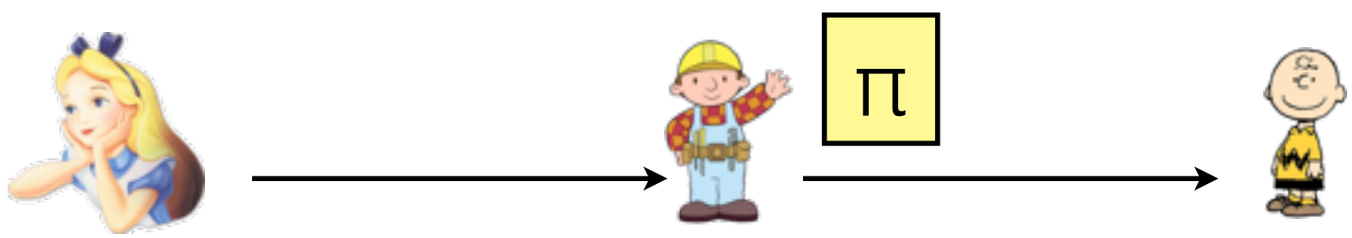
If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!



Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

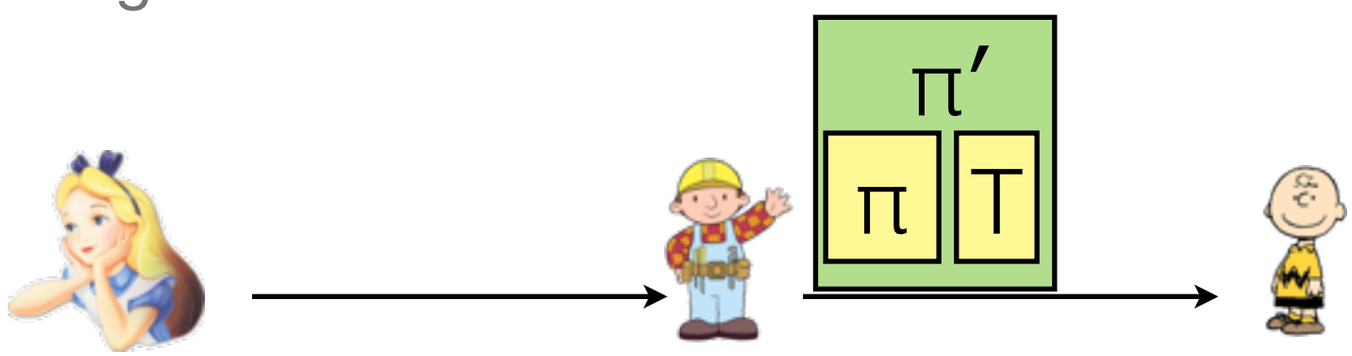
If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!



Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

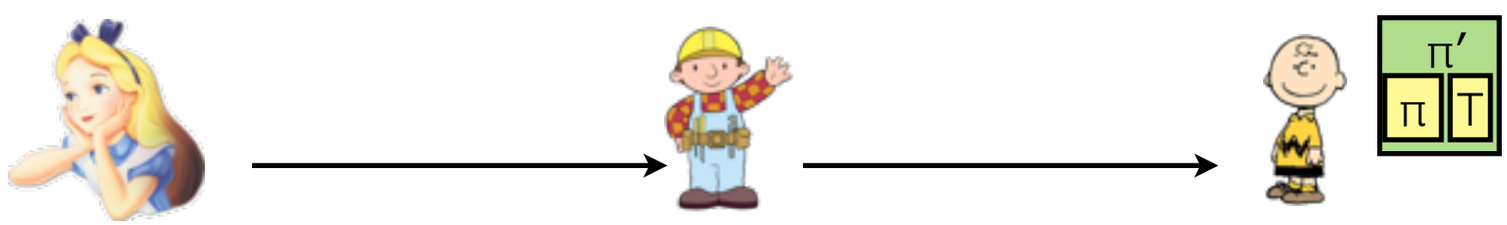
If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!



Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

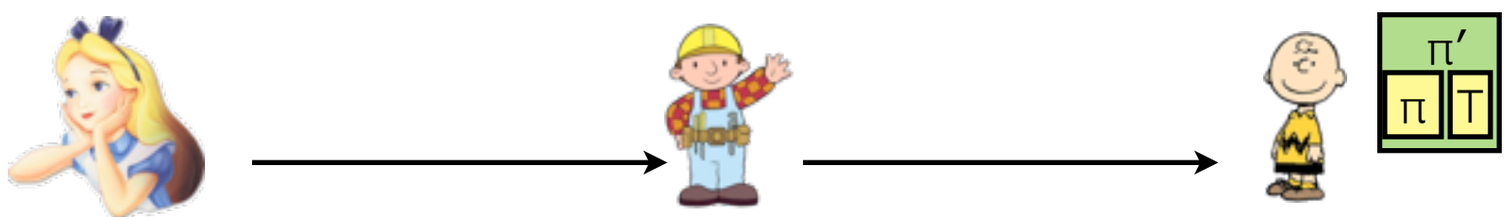
If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!



Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!



Can continue this process many times (Bob proves knowledge of Alice's proof π_A for x_A and an allowable transformation T_B to his instance x_B, Charlie proves knowledge of Bob's proof π_B for x_B and an allowable transformation T_C to his instance x_C, etc.)

Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!

Can continue this process many times (Bob proves knowledge of Alice's proof π_A for x_A and an allowable transformation T_B to his instance x_B, Charlie proves knowledge of Bob's proof π_B for x_B and an allowable transformation T_C to his instance x_C, etc.)

Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!

Can continue this process many times (Bob proves knowledge of Alice's proof π_A for x_A and an allowable transformation T_B to his instance x_B, Charlie proves knowledge of Bob's proof π_B for x_B and an allowable transformation T_C to his instance x_C, etc.)

Malleable SNARGs

Our goal: build malleability into SNARGs [BSW12]

If we use succinct non-interactive arguments of knowledge (SNARGs), a proof of knowledge of π could in fact be the same size!

Can continue this process many times (Bob proves knowledge of Alice's proof π_A for x_A and an allowable transformation T_B to his instance x_B, Charlie proves knowledge of Bob's proof π_B for x_B and an allowable transformation T_C to his instance x_C, etc.)

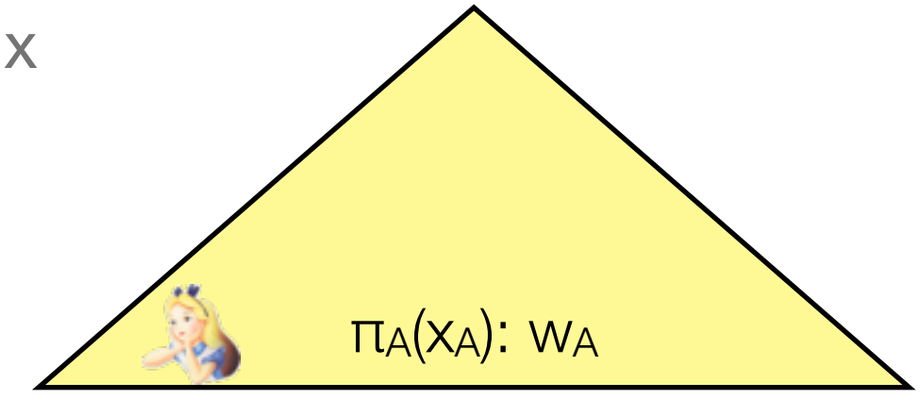
Malleable SNARGs

Malleable SNARGs

Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that $(x,w) \in R$, or a proof π, instance x' at the next tier down, and an allowable T such that $T(x') = x$

Malleable SNARGs

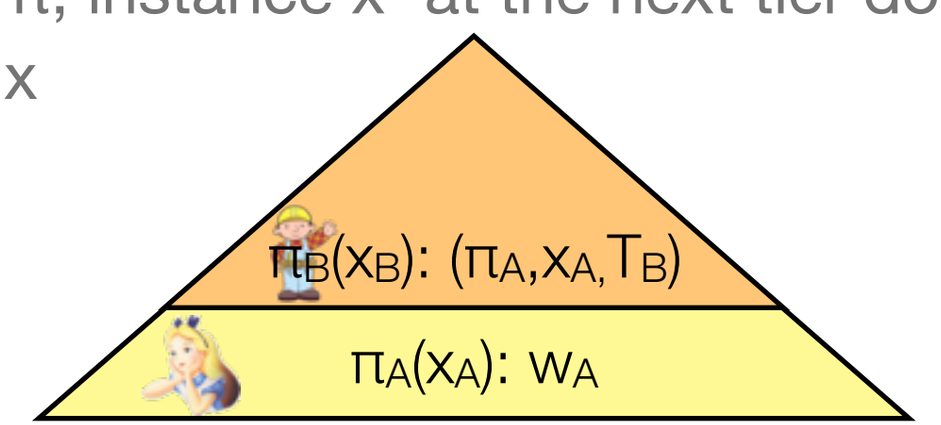
Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that (x,w)∈R, or a proof π, instance x′ at the next tier down, and an allowable T such that T(x′) = x



Malleable SNARGs

malleable SNARG

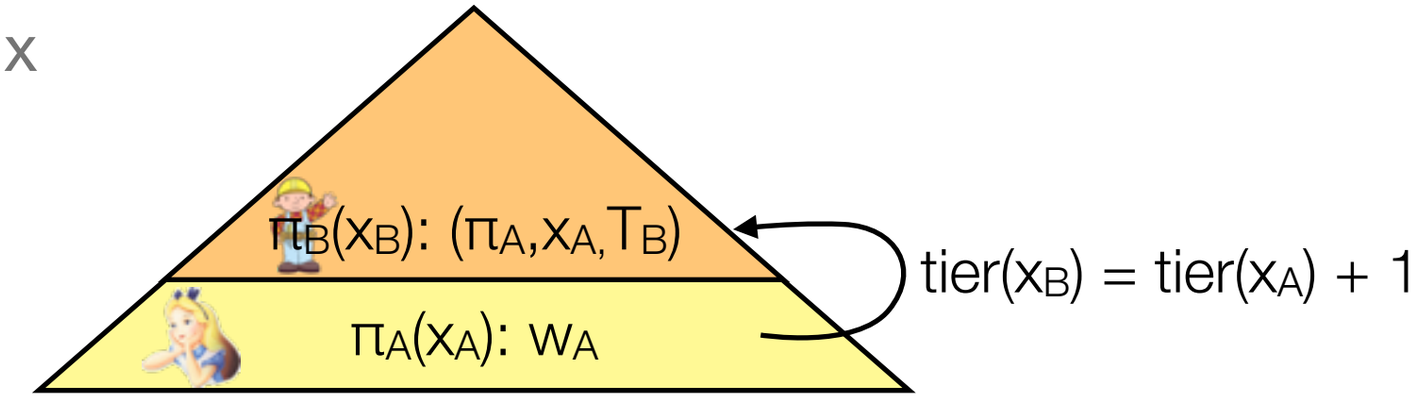
Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that $(x,w)\in R$, or a proof π, instance x' at the next tier down, and an allowable T such that $T(x') = x$



Malleable SNARGs

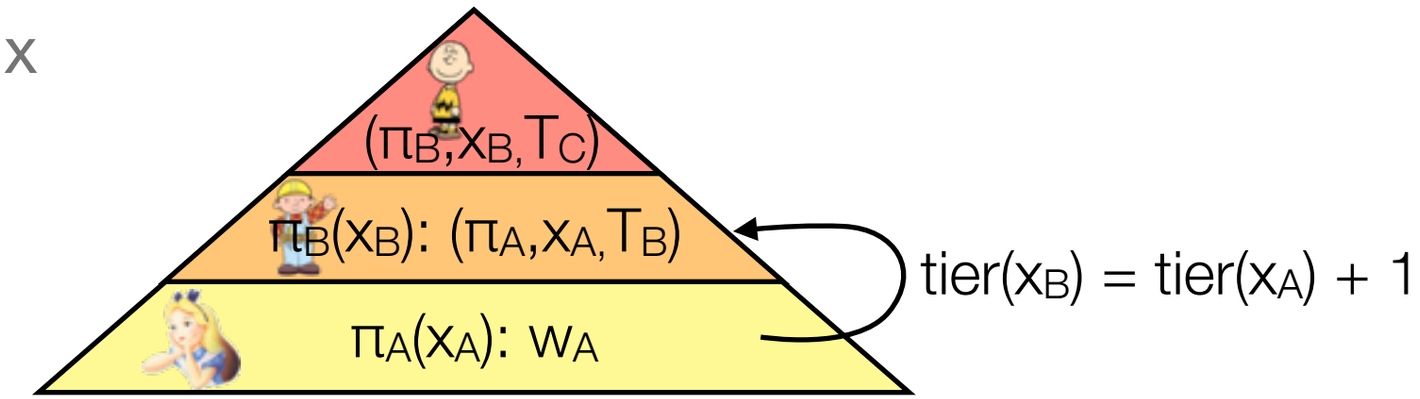
malleable SNARG

Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that $(x,w) \in R$, or a proof π, instance x' at the next tier down, and an allowable T such that $T(x') = x$



Malleable SNARGs

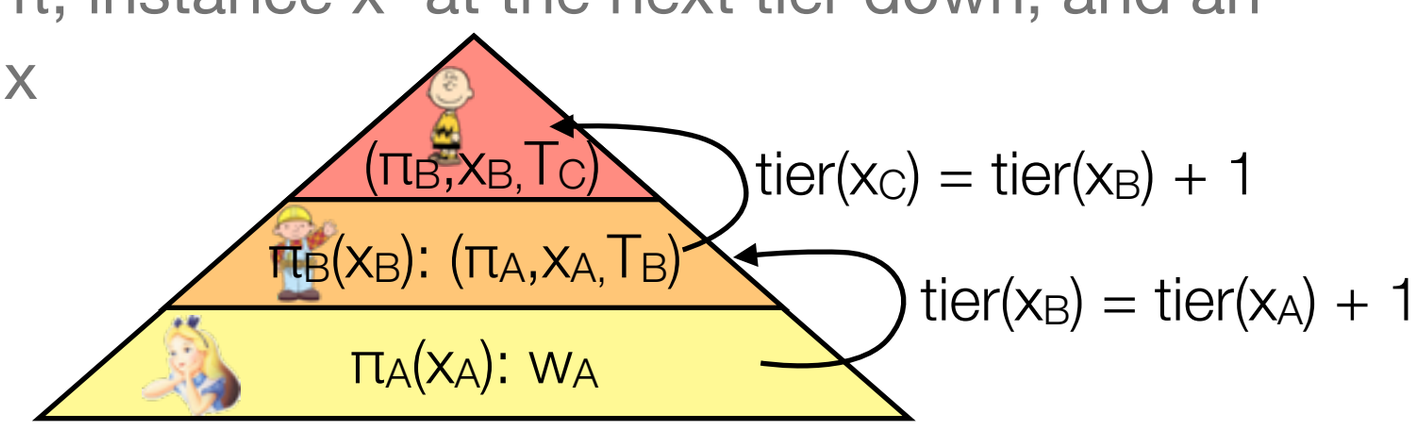
Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that (x,w)∈R, or a proof π, instance x′ at the next tier down, and an allowable T such that T(x′) = x



Malleable SNARGs

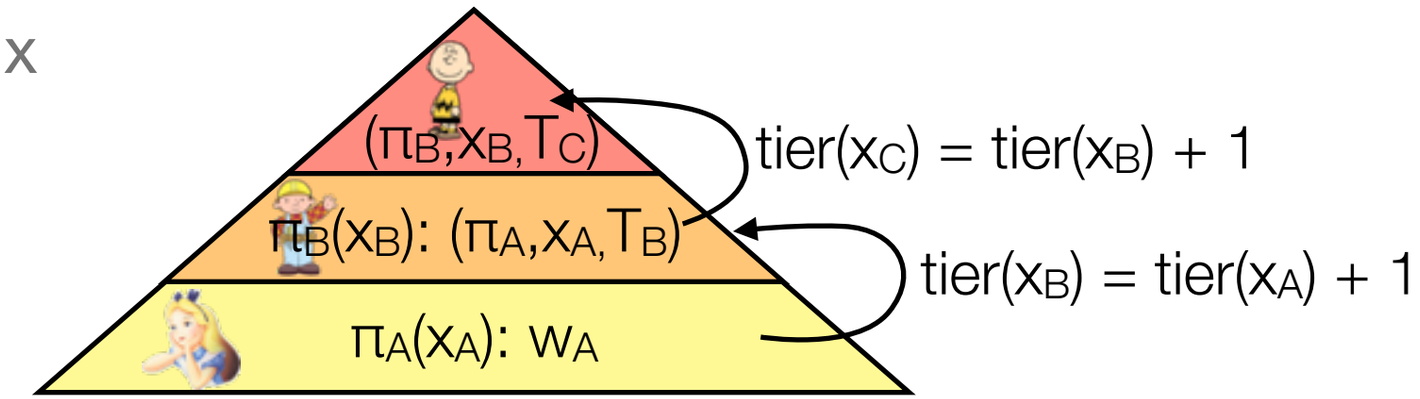
malleable SNARG

Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that $(x,w) \in R$, or a proof π, instance x' at the next tier down, and an allowable T such that $T(x') = x$



Malleable SNARGs

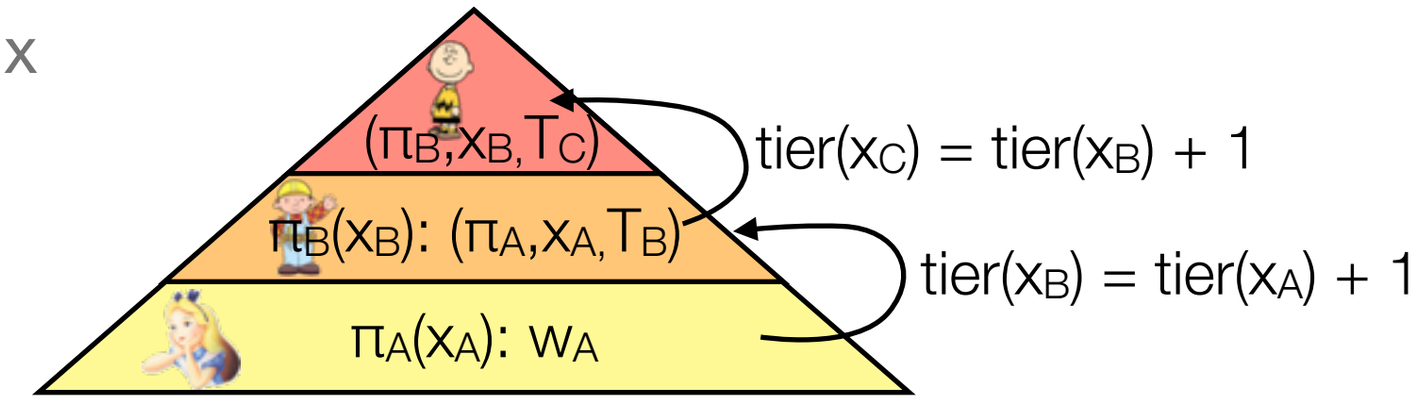
Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that $(x,w) \in R$, or a proof π, instance x' at the next tier down, and an allowable T such that $T(x') = x$



Zero knowledge and adaptive knowledge extraction are both preserved*, gain malleability with respect to t-tiered transformations*

Malleable SNARGs

Intuitively, to form a proof for an instance x, prove you know a fresh witness w such that (x,w)∈R, or a proof π, instance x′ at the next tier down, and an allowable T such that T(x′) = x

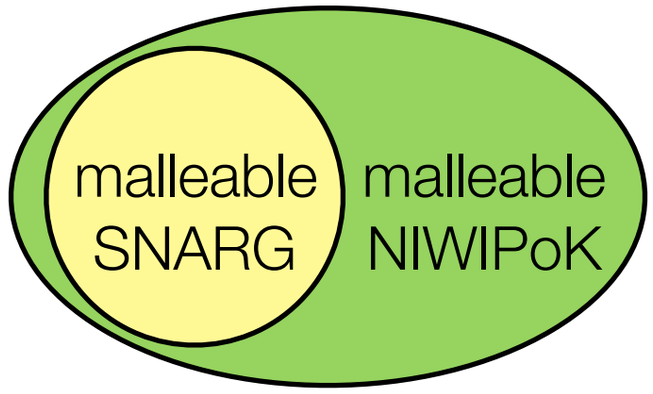


Zero knowledge and adaptive knowledge extraction are both preserved*, gain malleability with respect to t-tiered transformations*

*Since extractor might have to "tunnel down" t must be a constant [BSW12,BCCT13] and we use a stronger notion of extraction (consider non-uniform adversaries)

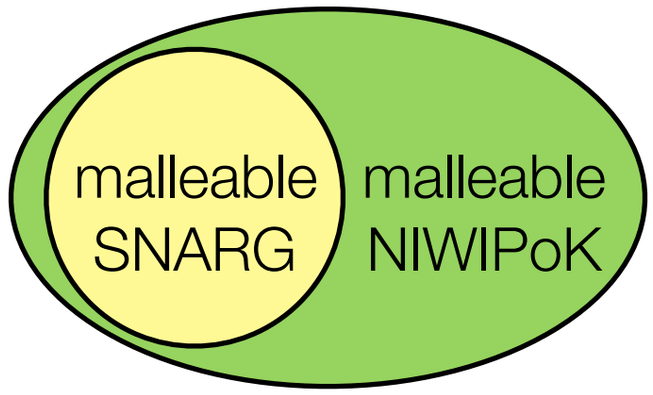
Boosting to full extractability

Boosting to full extractability



Our goal: get from adaptive knowledge extraction to stronger soundness

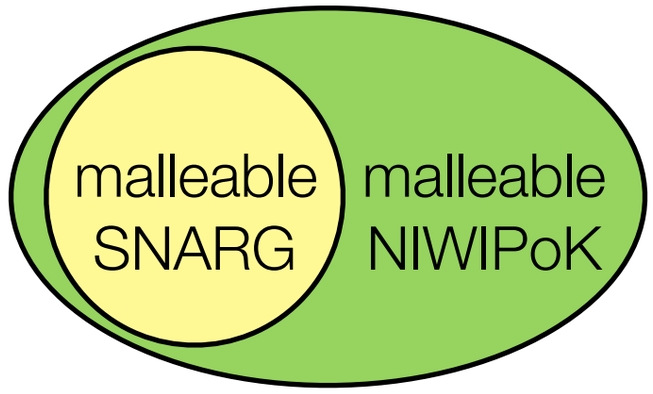
Boosting to full extractability



Our goal: get from adaptive knowledge extraction to stronger soundness

Rather than even try to reconcile adaptive knowledge extraction with something much stronger like extractability or CM-SSE, just use regular soundness of SNARG

Boosting to full extractability

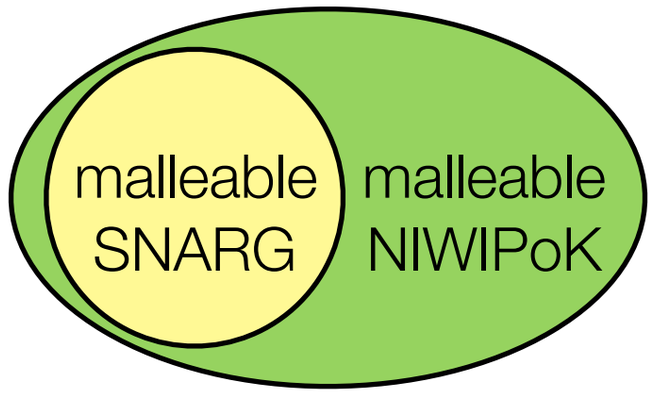


Our goal: get from adaptive knowledge extraction to stronger soundness

Rather than even try to reconcile adaptive knowledge extraction with something much stronger like extractability or CM-SSE, just use regular soundness of SNARG

SNARG now just proves knowledge of plaintext such that $(x,w) \in R$

Boosting to full extractability

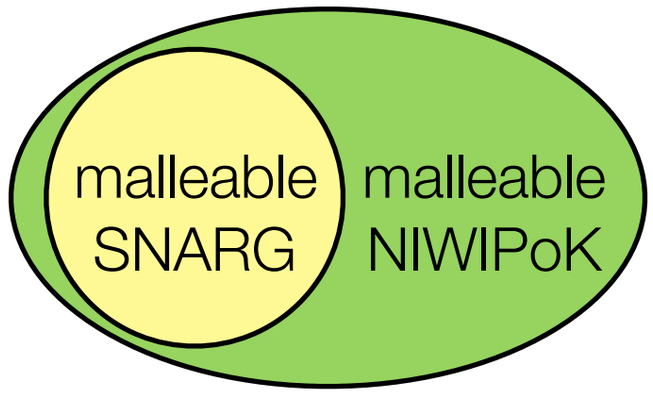


Our goal: get from adaptive knowledge extraction to stronger soundness

Rather than even try to reconcile adaptive knowledge extraction with something much stronger like extractability or CM-SSE, just use regular soundness of SNARG

SNARG now just proves knowledge of plaintext such that $(x,w) \in R$

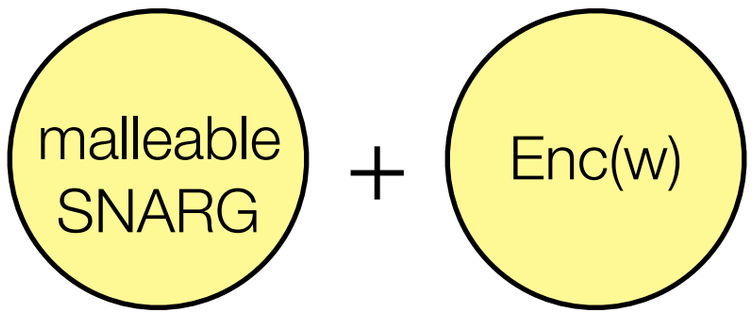
Boosting to full extractability



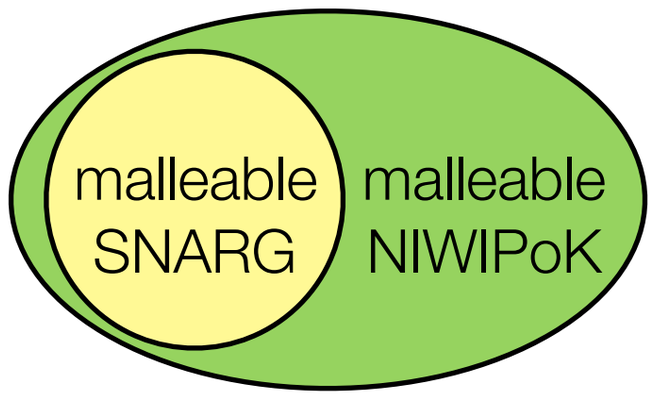
Our goal: get from adaptive knowledge extraction to stronger soundness

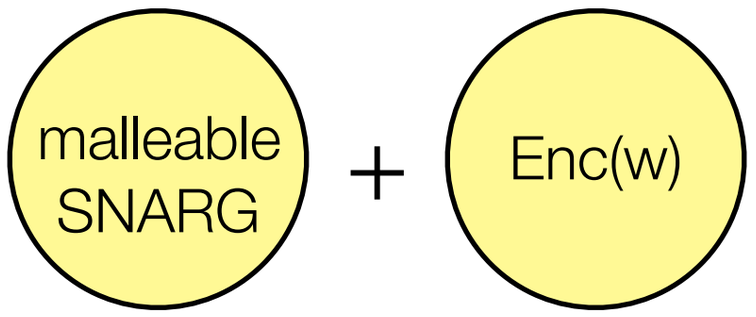
Rather than even try to reconcile adaptive knowledge extraction with something much stronger like extractability or CM-SSE, just use regular soundness of SNARG

SNARG now just proves knowledge of plaintext such that $(x,w) \in R$

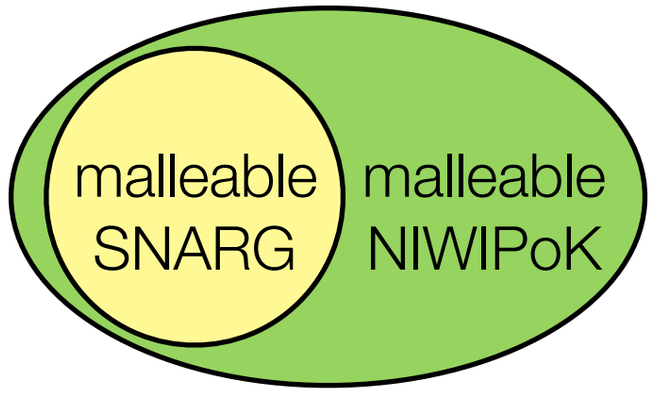


Boosting to full extractability

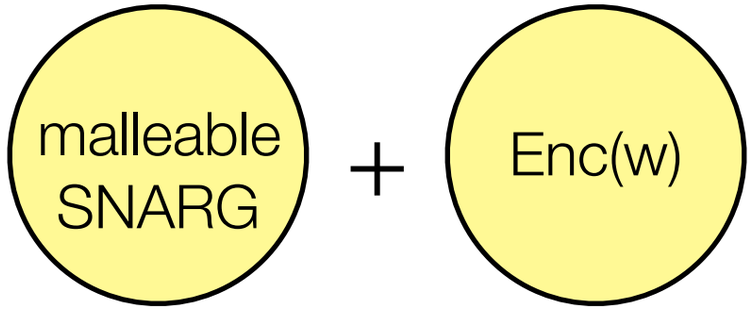




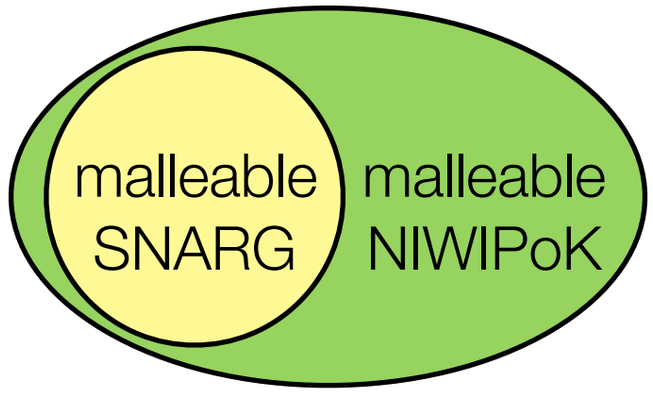
Boosting to full extractability



Extraction is quite simple: τ_e is decryption key, and extractor decrypts, so we never need to use non-black-box SNARG extractor!

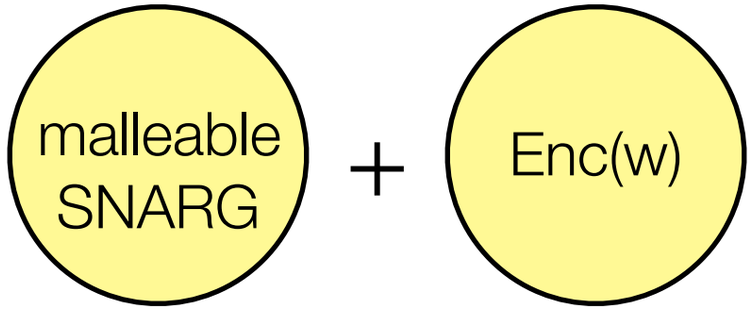


Boosting to full extractability

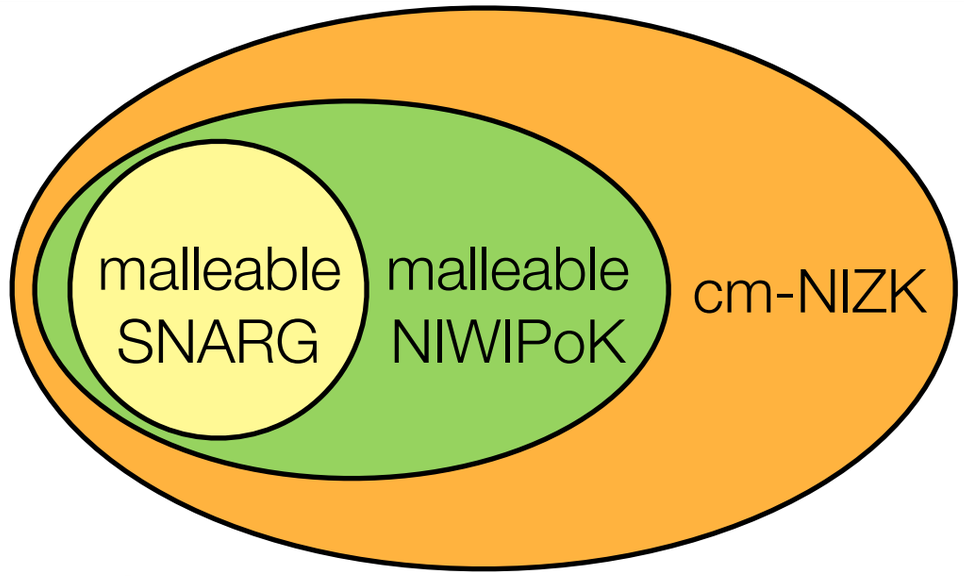


Extraction is quite simple: τ_e is decryption key, and extractor decrypts, so we never need to use non-black-box SNARG extractor!

If we use a fully-homomorphic encryption scheme, can preserve malleability for t-tiered transformations (but we do lose succinctness)



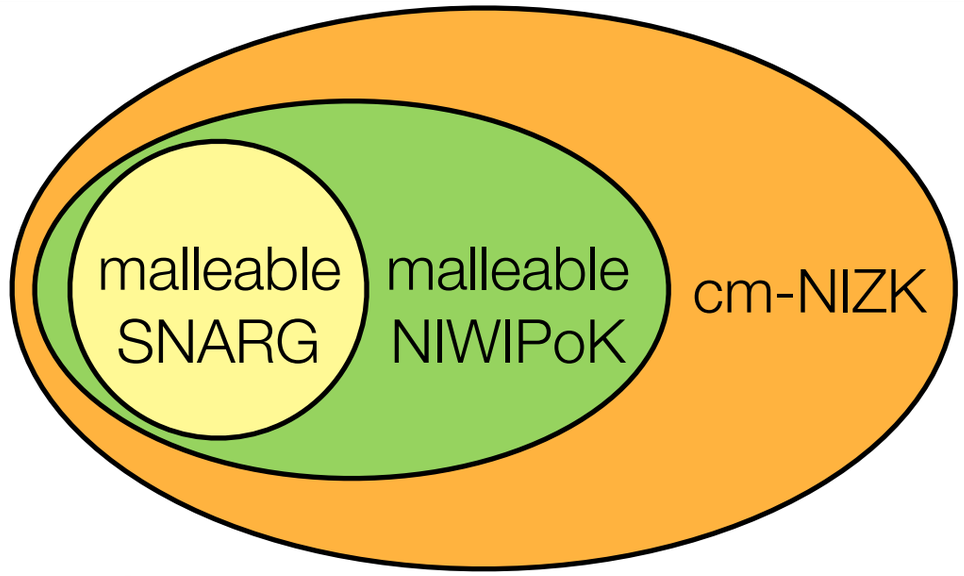
Boosting to CM-SSE



Our goal: preserve malleability with respect to t-tiered transformations

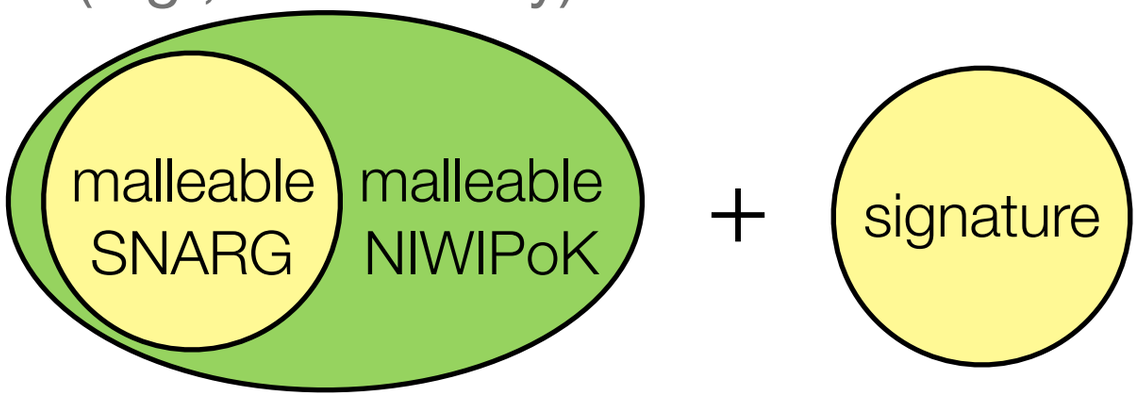
Essentially amplify [CKLM12] construction; don’t assume certain transformations (e.g., the identity) are allowable

Boosting to CM-SSE

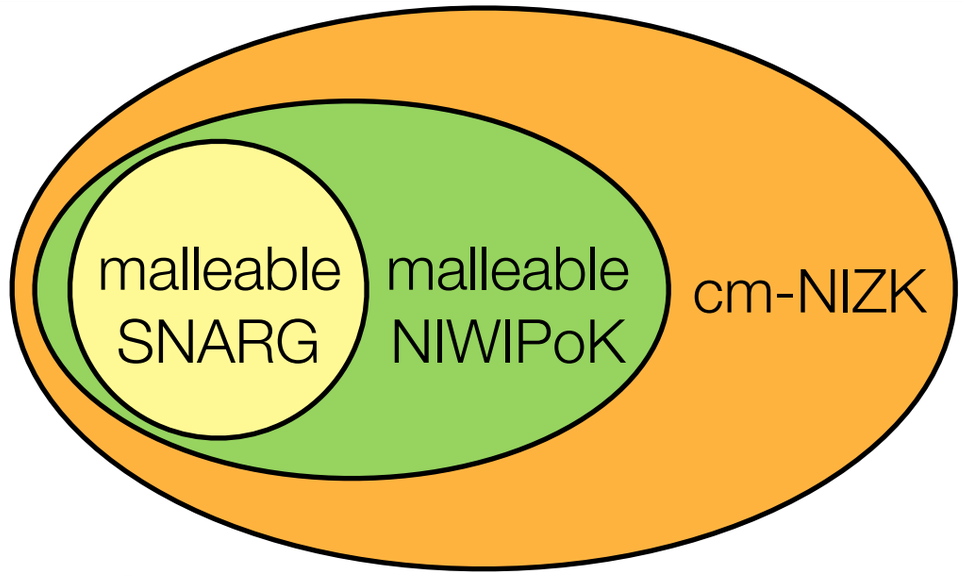


Our goal: preserve malleability with respect to t-tiered transformations

Essentially amplify [CKLM12] construction; don't assume certain transformations (e.g., the identity) are allowable

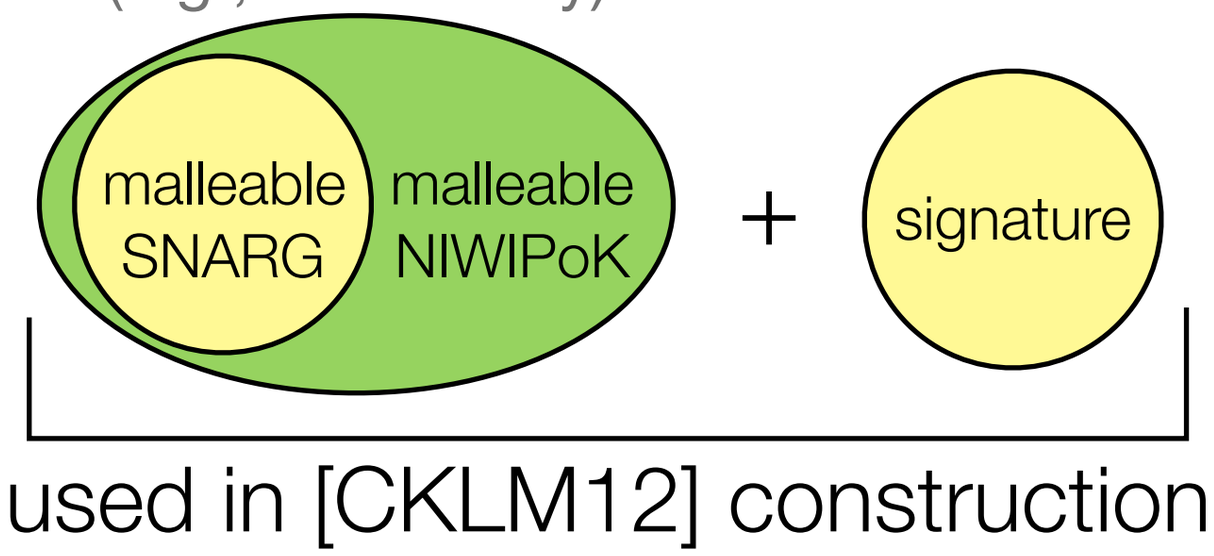


Boosting to CM-SSE

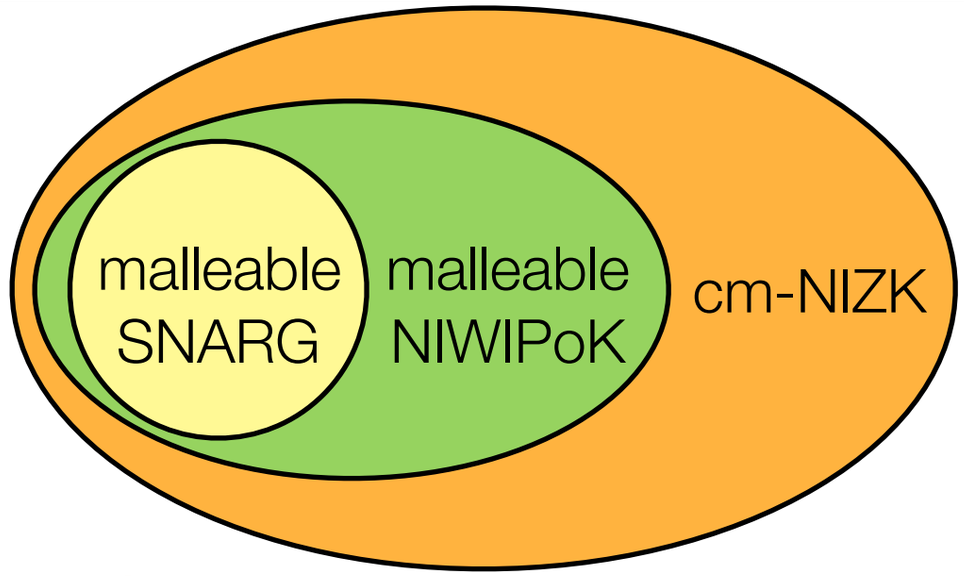


Our goal: preserve malleability with respect to t-tiered transformations

Essentially amplify [CKLM12] construction; don't assume certain transformations (e.g., the identity) are allowable

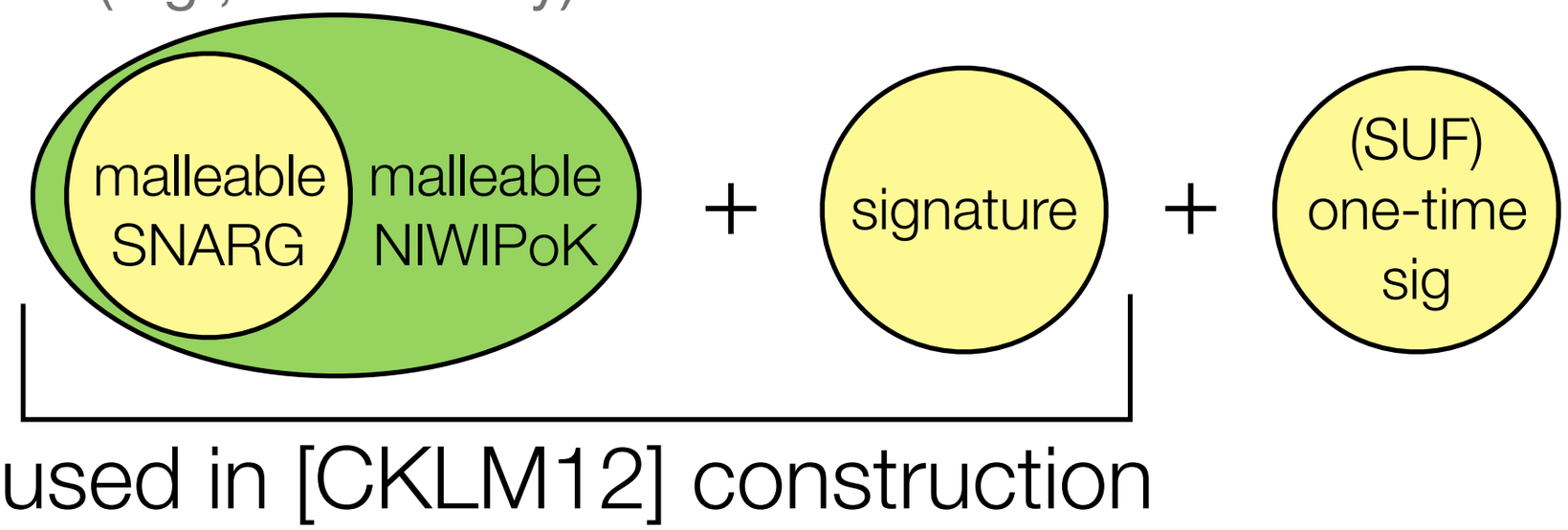


Boosting to CM-SSE



Our goal: preserve malleability with respect to t-tiered transformations

Essentially amplify [CKLM12] construction; don't assume certain transformations (e.g., the identity) are allowable



Outline

Definitions

SNARGs to cm-NIZKs

Applying the cm-NIZK

Conclusions

How to apply previous cm-NIZK?

How to apply previous cm-NIZK?

Suppose you have some (theoretical) application that uses a cm-NIZK

How to apply previous cm-NIZK?

Suppose you have some (theoretical) application that uses a cm-NIZK

In [CKLM12], developed a methodology for showing the existence of a cm-NIZK called CM-friendliness

How to apply previous cm-NIZK?

Suppose you have some (theoretical) application that uses a cm-NIZK

In [CKLM12], developed a methodology for showing the existence of a cm-NIZK called CM-friendliness

Needed to address our reliance on Groth-Sahai proofs

How to apply previous cm-NIZK?

Suppose you have some (theoretical) application that uses a cm-NIZK

In [CKLM12], developed a methodology for showing the existence of a cm-NIZK called CM-friendliness

Needed to address our reliance on Groth-Sahai proofs

Basically had to show that proof verification could consist of a set of pairing product equations, and that instances, witnesses, and transformations could be represented and transformed as elements in a bilinear group, etc.

How to apply previous cm-NIZK?

Suppose you have some (theoretical) application that uses a cm-NIZK

In [CKLM12], developed a methodology for showing the existence of a cm-NIZK called CM-friendliness

Needed to address our reliance on Groth-Sahai proofs

Basically had to show that proof verification could consist of a set of pairing product equations, and that instances, witnesses, and transformations could be represented and transformed as elements in a bilinear group, etc.

To instantiate a cm-NIZK, had to therefore jump through a lot of hoops!

How to apply this cm-NIZK?

How to apply this cm-NIZK?

The cm-NIZK we just constructed can be applied much more easily

How to apply this cm-NIZK?

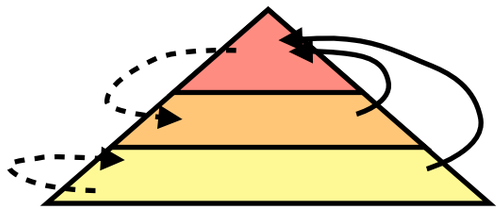
The cm-NIZK we just constructed can be applied much more easily

In the paper, we show how to construct a compact verifiable shuffle with proof size O(L+M) (where L = # voters, M = # shufflers)

How to apply this cm-NIZK?

The cm-NIZK we just constructed can be applied much more easily

In the paper, we show how to construct a compact verifiable shuffle with proof size O(L+M) (where L = # voters, M = # shufflers)

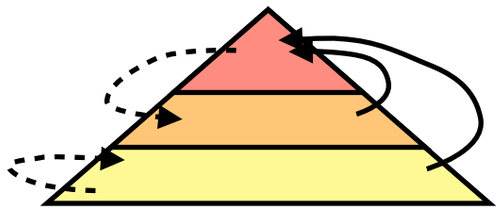


- Step 1 (mandatory!): Show that class of allowable transformations is t-tiered (for shuffle: each mix server increments the tier by 1)

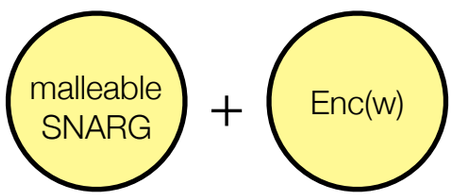
How to apply this cm-NIZK?

The cm-NIZK we just constructed can be applied much more easily

In the paper, we show how to construct a compact verifiable shuffle with proof size O(L+M) (where L = # voters, M = # shufflers)



- Step 1 (mandatory!): Show that class of allowable transformations is t-tiered
  (for shuffle: each mix server increments the tier by 1)



- Step 2: Give instantiation for encryption scheme depending on how much malleability you want
  (for shuffle: multiplicatively homomorphic encryption)

Outline

Definitions

SNARGs to cm-NIZKs

Applying the cm-NIZK

Conclusions

Conclusions and open problems

Conclusions and open problems

Constructed generic cm-NIZKs for a general class of transformations, and intermediate primitives of potential independent interest

Conclusions and open problems

Constructed generic cm-NIZKs for a general class of transformations, and intermediate primitives of potential independent interest

Saw example (shuffle) of how to construct applications using this cm-NIZK

Conclusions and open problems

Constructed generic cm-NIZKs for a general class of transformations, and intermediate primitives of potential independent interest

Saw example (shuffle) of how to construct applications using this cm-NIZK

Are there applications that directly exploit this expanded malleability?

Conclusions and open problems

Constructed generic cm-NIZKs for a general class of transformations, and intermediate primitives of potential independent interest

Saw example (shuffle) of how to construct applications using this cm-NIZK

Are there applications that directly exploit this expanded malleability?

Full version is online at `eprint.iacr.org/2012/506` (recently updated!)

Conclusions and open problems

Constructed generic cm-NIZKs for a general class of transformations, and intermediate primitives of potential independent interest

Saw example (shuffle) of how to construct applications using this cm-NIZK

Are there applications that directly exploit this expanded malleability?

Full version is online at `eprint.iacr.org/2012/506` (recently updated!)

Thanks!
Any questions?