Commit-and-Prove Zero-Knowledge Proof Systems and Extensions

Daniel Benarroch
QEDIT,
Tel-Aviv, Israel

Matteo Campanelli
Aarhus University,
Denmark

Dario Fiore
IMDEA Software,
Madrid, Spain

Jihye Kim
Kookmin University,
Seoul, Korea

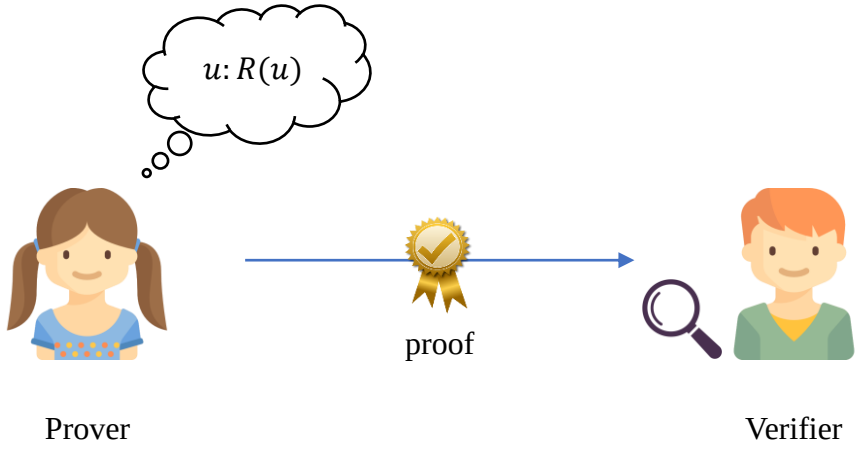
Jiwon Lee
Hanyang University,
Seoul, Korea

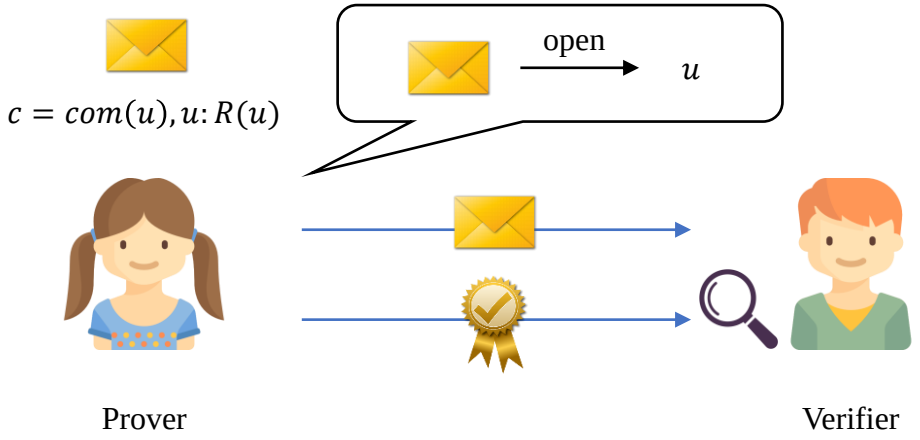
Hyunok Oh
Hanyang University,
Seoul, Korea

Anaïs Querol
IMDEA Software,
Madrid, Spain

4th ZKProof Workshop

Commit-and-Prove (CP)





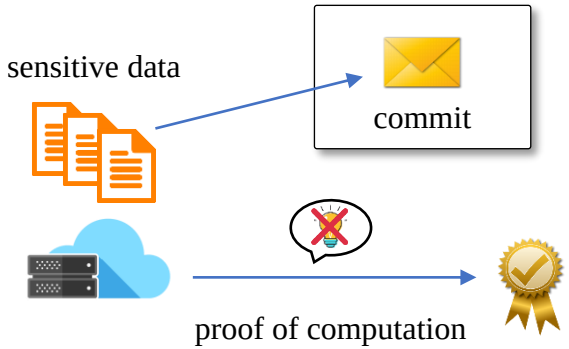
Zero-Knowledge:

"Trusting properties claimed by someone else on data that you have not seen"

Commit-and-Prove Zero-Knowledge:

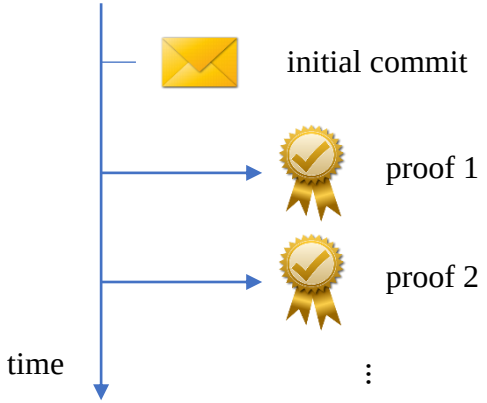
"Trusting properties claimed by someone else on data that you have not seen, **but that can be pointed to**"

Usability



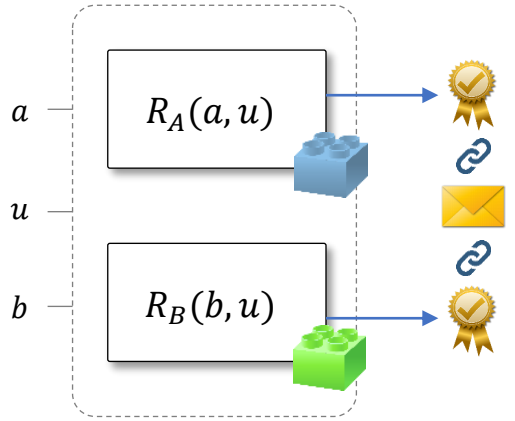
Compressing/Fingerprinting

- Delegate storage and computation
- Verifier keeps the fingerprint
- Prove data-related computation



Commit ahead of time

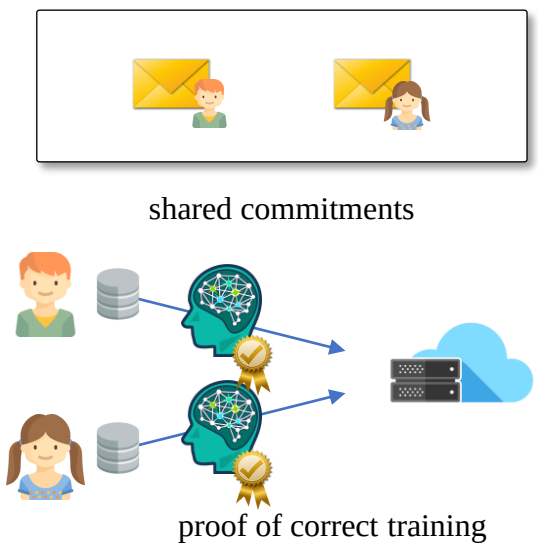
- Commit first, and prove later
- First commit enables multiple arguments
- Useful for stabile data



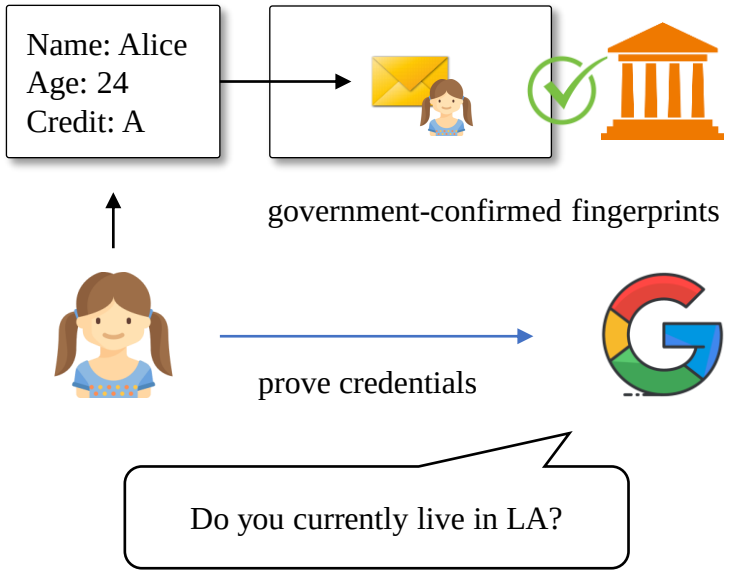
Modular composition

- Modular composition of proofs
- Use different proof systems and connect
- More flexible and efficient structure

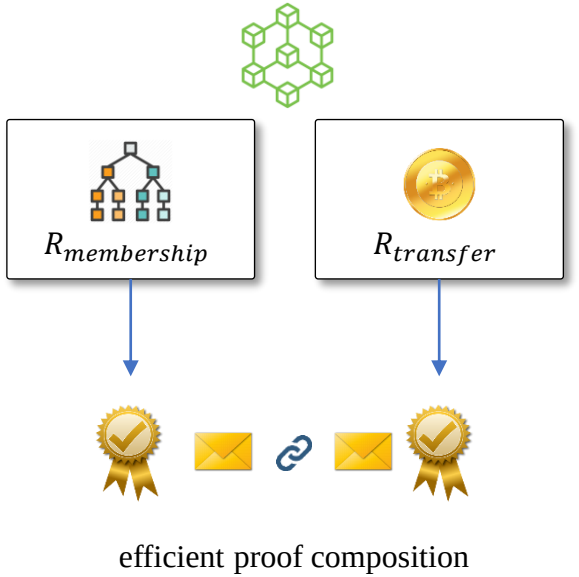
Applications



Federated Learning

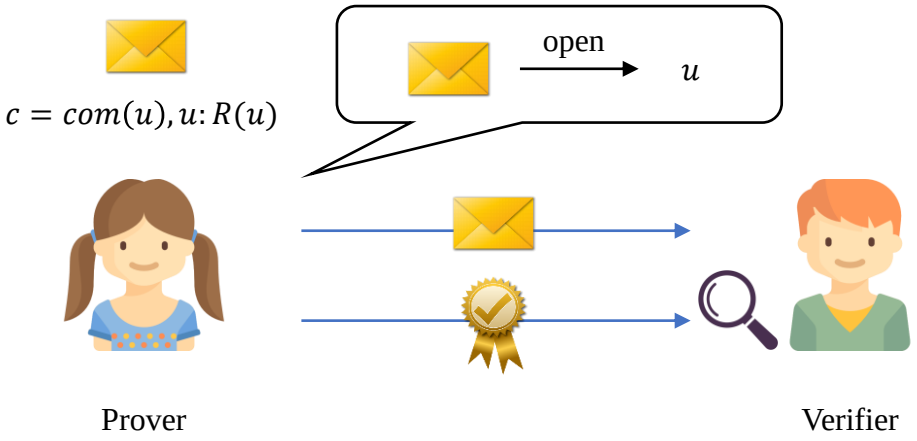


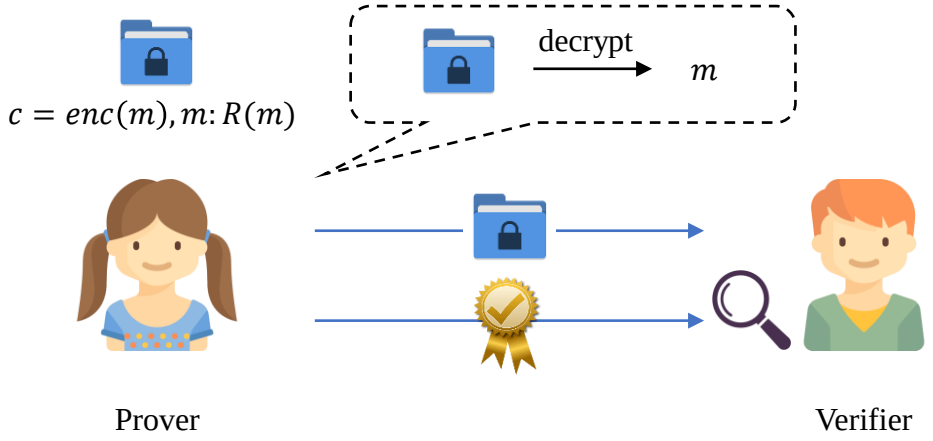
Self Sovereign Identity
(Anonymous credentials)



Blockchains

Extension: Encrypt-and-Prove (EP)





Commit-and-Prove Zero-Knowledge:

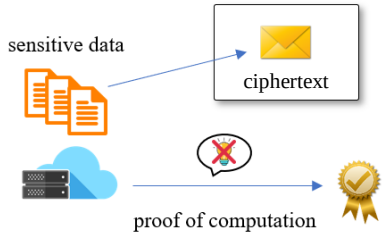
"Trusting properties claimed by someone else on data that you have not seen, **but that can be pointed to**"

Encrypt-and-Prove Zero-Knowledge:

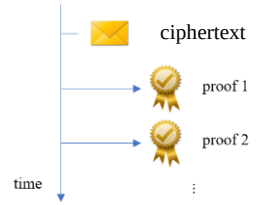
"Trusting properties claimed by someone else on message that you have not seen, **but that can be pointed to a ciphertext**"

Usability

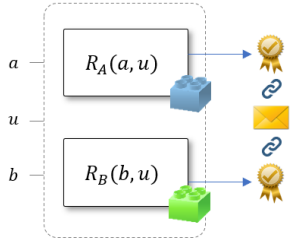
Same as commit-and-prove...



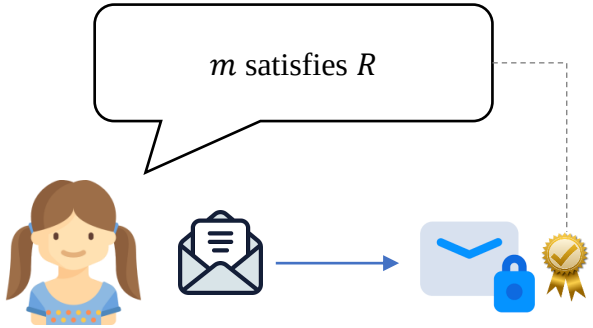
Compressing/Fingerprinting



Commit ahead of time

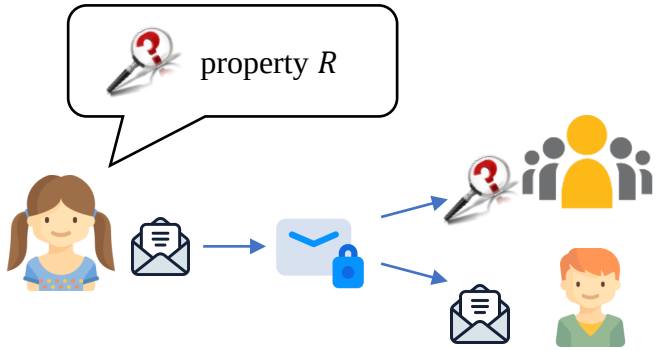


Modular composition



Verifiable encryption

- Proof guarantees the encryption itself
- Encrypt msg while proving its properties



Privacy differentiation

- Encryption reveals different information
- The decryptor gets the whole message
- The public only sees the property holds

Applications

E-commerce

raw product transaction

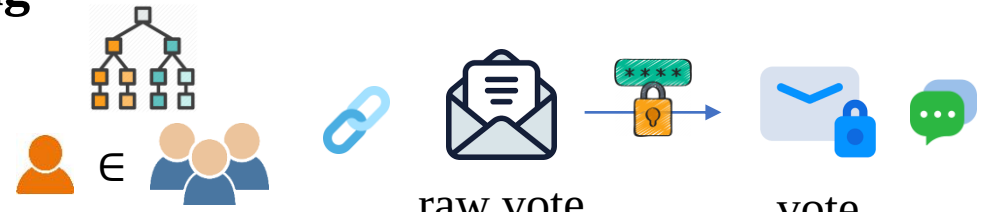
the product satisfies “certain properties”
ex: metadata, copyrights...

Digital contract

raw contract contract

the contract satisfies “certain properties”
ex: contract type, date, deposit...

Voting

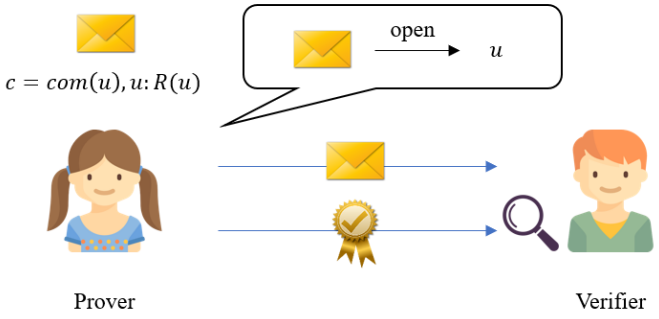


membership test raw vote vote

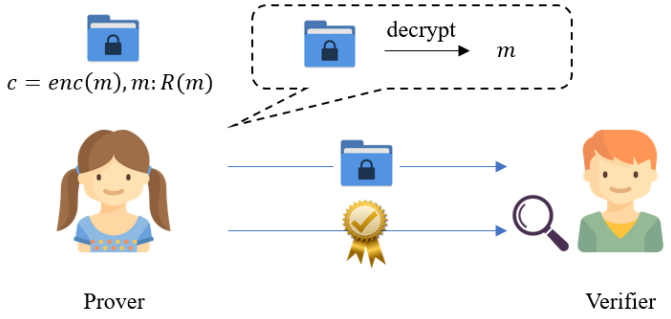
the voter (linked to m) is “within the membership”
the vote is “valid (ex: vote sum is integer 1)”

Construction

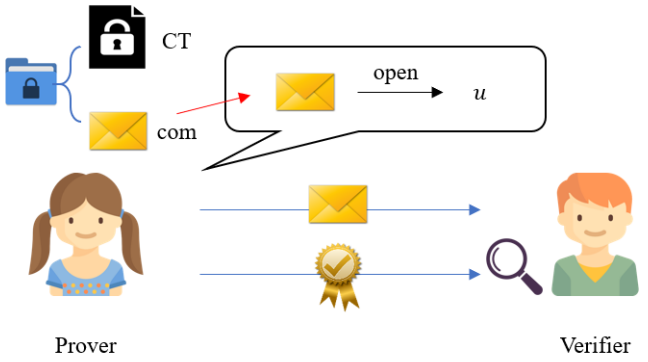
- Exploit cc-SNARK: lifting ElGamal to work with the SNARK
  - If an encryption can output or contain a commitment (commit-carrying encryption), it can be plugged into the CP as a commitment input
  - ex: SAVER; CT in SAVER contains Pedersen commitment of the message for knowledge soundness



Commit-and-Prove

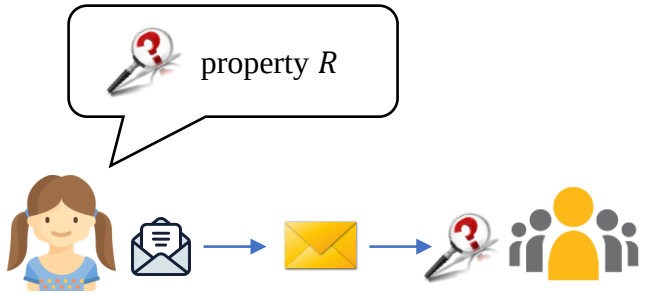


Encrypt-and-Prove



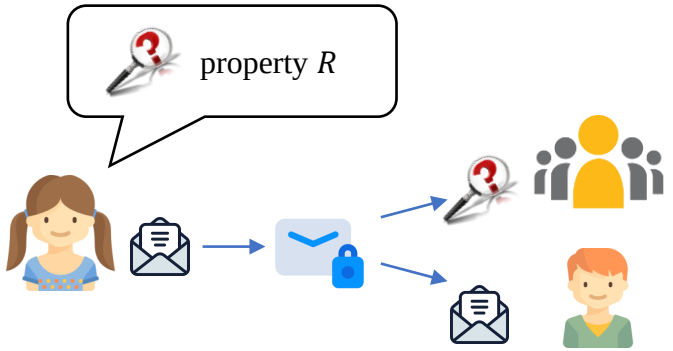
Commit-carrying encryption

Use case: what's different?



Commit-and-Prove

- Commitment (binding, hiding)
- Commitment reacts to the verifier (public)
- For applications where fingerprinting/modularity is enough



Encrypt-and-Prove

- PK Encryption (hiding, no binding in definition)
- Ciphertext reacts to the verifier (public) and the decryptor
- For applications where privacy differentiation is required

Standardization Status

- General commit/encrypt-and-prove syntax
  - both separated and combined version
  - i.e., CP-SNARK, EP-SNARK, CP/EP-SNARK

- Reference document available at
  - https://docs.zkproof.org/pages/standards/accepted-workshop4/proposal-commit.pdf

- Working group available at
  - https://community.zkproof.org/g/WG_COMMIT_PROVE

Next step

- Agree on specific constructions

- Commit-and-prove: ready to proceed
  - commitment scheme
  - commit-and-prove (CP) scheme

- Encrypt-and-prove: gather ideas
  - encryption scheme
  - encrypt-and-prove (EP) scheme

Commitment candidates

- Pedersen commitment
  - Different types of distributions (e.g. uniform, Lagrange polynomials)
  - Possibly required to define preliminaries (e.g. groups, properties)

- SNARK-friendly commitments
  - Ad-hoc scheme-specific commitments (e.g. Pedersen on ZCash JubJub)

- Non-algebraic commitments
  - Collision-resistant hash (e.g. SHA256)
  - Merkle trees and some vector commitments in general

CP candidates

- Sigma protocols
  - Need to agree on syntax (e.g. interaction, random coins, forking lemma)
  - Common reference string vs. Rewinding definition of hiding
- Groth-Sahai
- Groth16 with SNARK-friendly hash/commitments
- LegoGroth16

Encryption candidates

- ElGamal encryption
  - Exponential version ($g^m \cdot h^r$)
  - This may require bunch of definitions on preliminaries

- SAVER [LCKO19]
  - Based on ElGamal, SNARK-linkable encryption (avoid encoding)
  - May want to trim some unnecessary features (e.g. rerandomization)

- SNARK-friendly encryption
  - Algebraic ElGamal
  - RSA from c0c0 (libsnark)

EP candidates

- Basically, similar to the CP candidates
  - Sigma protocols
  - Groth-Sahai
  - Groth16 with SNARK-friendly encryption

- SAVER + LegoGroth16
  - A “working” construction in current status
  - Plugging in the SAVER scheme as a cc-SNARK

Discussion: candidates

Commitment

- Pedersen commitment
  - Different types of distributions (e.g. uniform, Lagrange polynomials)
  - Possibly required to define preliminaries (e.g. groups, properties)
- SNARK-friendly commitments
  - Ad-hoc scheme-specific commitments (e.g. Pedersen on ZCash JubJub)
- Non-algebraic commitments
  - Collision-resistant hash (e.g. SHA256)
  - Merkle trees and some vector commitments in general

Commit-and-prove

- Sigma protocols
  - Need to agree on syntax (e.g. interaction, random coins, forking lemma)
  - Common reference string vs. Rewinding definition of hiding
- Groth-Sahai
- Groth16 with SNARK-friendly hash/commitments
- LegoGroth16

Discussion 1: what are the top two items we want to standardize in the next year from the candidates?

Discussion: applications

- There are some “popular” applications
  - Voting
  - Anonymous credentials

- Providing an official “recipe” for them may help practitioners

Discussion 2: does it make sense to create “recipes” to build/explain usual applications, such as voting or anonymous credentials?

Discussion: encryption

- ElGamal encryption
  - Exponential version ($g^m \cdot h^r$)
  - This may require bunch of definitions on preliminaries
- SAVER [LCKO19]
  - Based on ElGamal, SNARK-linkable encryption (avoid encoding)
  - May want to trim some unnecessary features (e.g. rerandomization)
- SNARK-friendly encryption
  - Algebraic ElGamal
  - RSA from c0c0 (libsnark)

- Commit-carrying encryption is one way to achieve encrypt-and-prove

Discussion 3: what could be the candidate for the construction of “encryption”? And do we want to standardize “commit-carrying encryption”?

Discussion: extension

- There exists encryption with more functionalities
  - ex: Identity-Based Encryption (IBE), Attribute-Based Encryption (ABE)
- In most applications, practitioners may need more than just an encryption
- It might be helpful to provide more diverse notion of EP for them

Discussion 4: should we extend the notion of EP further, i.e., look at finer-grained notions of encryption (ex: ABE-and-prove)?

Discussion: X-and-prove

- Commitment extended to encryption (CP to EP)
- Similar to above, we may also extend CP to different context
  - Accumulate-and-Prove: for efficient proving of accumulation
  - Sign-and-Prove: for efficient proving of signature verification

Discussion 5: what extensions can we consider for the standard in the future? For example, Sign-and-Prove?

Topics

Candidates: what are the top two items we want to standardize in the next year from the candidates?

Applications: does it make sense to create “recipes” to build/explain usual applications, such as voting or anonymous credentials?

Encryption: what could be the candidate for the construction of “encryption”? And do we want to standardize “commit-carrying encryption”?

Extension: should we extend the notion of EP further, i.e., look at finer-grained notions of encryption (ex: ABE-and-prove)?

X-and-Prove: what extensions can we consider for the standard in the future? For example, Sign-and-Prove?