Hands-on Zero-Knowledge Basics: KZG Commitments

Adrià Torralba-Agell[1, 2]
atorralbaag@uoc.edu

[1]Universitat Oberta de Catalunya
K-riptography and Information Security for Open Networks (KISON) Research Group

[2]Cybercat - Center for Cybersecurity Research of Catalonia

July 7, 2023

First things first...

Outline

1. Why commitments?
2. KZG commitments
3. Please, delete the toxic waste

Why commitments?

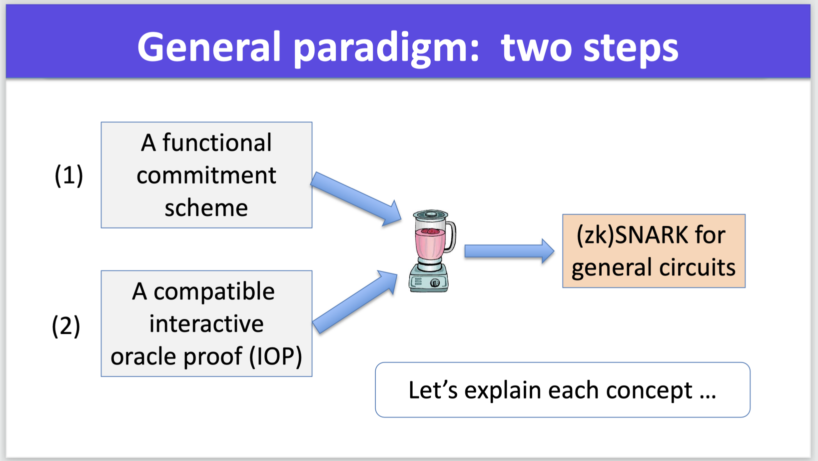


Figure: From building a SNARK, from Dan Boneh's zkWhiteboards session 2

Why commitments?

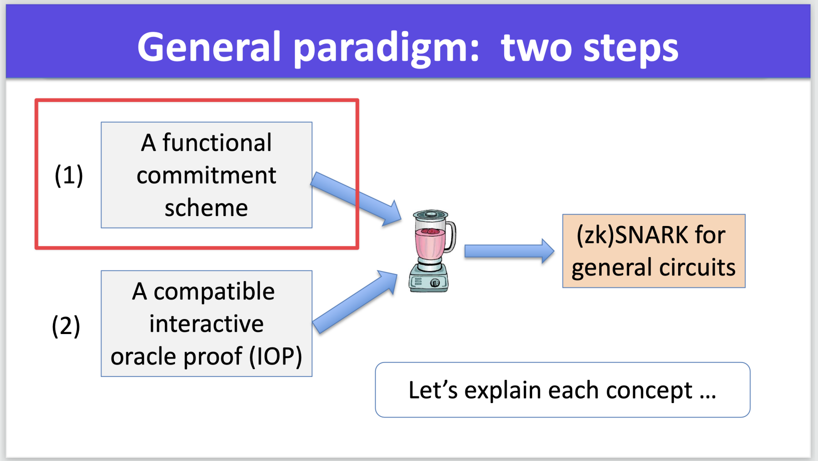


Figure: From building a SNARK, from Dan Boneh's zkWhiteboards session 2

Proto-Danksharding (a.k.a. EIP-4844)

Couldn't we use some other commitment scheme without a trusted setup?

Unfortunately, using anything other than KZG (eg. IPA or SHA256) would make the sharding roadmap much more difficult. This is for a few reasons:

- Non-arithmetic commitments (eg. hash functions) are not compatible with data availability sampling, so if we use such a scheme we would have to change to KZG anyway when we move to full sharding.
- IPAs may be compatible with data availability sampling, but it leads to a much more complex scheme with much weaker properties (eg. self-healing and distributed block building become much harder)
- Neither hashes nor IPAs are compatible with a cheap implementation of the point evaluation precompile. Hence, a hash or IPA-based implementation would not be able to effectively benefit ZK rollups or support cheap fraud proofs in multi-round optimistic rollups.
- One way to keep data availability sampling and point evaluation but introduce another commitment is to store multiple commitments (eg. KZG and SHA256) per blob. But this has the problem that either (i) we need to add a complicated ZKP proof of equivalence, or (ii) all consensus nodes would need to verify the second commitment, which would require them to download the full data of all blobs (tens of megabytes per slot).

Hence, the functionality losses and complexity increases of using anything but KZG are unfortunately much greater than the risks of KZG itself. Additionally, any KZG-related risks are contained: a KZG failure would only affect rollups and other applications depending on blob data, and leave the rest of the system untouched.

Figure: From Proto-Danksharding FAQ by Vitalik.

Proto-Danksharding (a.k.a. EIP-4844)

Couldn't we use some other commitment scheme without a trusted setup?

Unfortunately, using anything other than KZG (eg. IPA or SHA256) would make the sharding roadmap much more difficult. This is for a few reasons:

- Non-arithmetic commitments (eg. hash functions) are not compatible with data availability sampling, so if we use such a scheme we would have to change to KZG anyway when we move to full sharding.
- IPAs may be compatible with data availability sampling, but it leads to a much more complex scheme with much weaker properties (eg. self-healing and distributed block building become much harder)
- Neither hashes nor IPAs are compatible with a cheap implementation of the point evaluation precompile. Hence, a hash or IPA-based implementation would not be able to effectively benefit ZK rollups or support cheap fraud proofs in multi-round optimistic rollups.
- One way to keep data availability sampling and point evaluation but introduce another commitment is to store multiple commitments (eg. KZG and SHA256) per blob. But this has the problem that either (i) we need to add a complicated ZKP proof of equivalence, or (ii) all consensus nodes would need to verify the second commitment, which would require them to download the full data of all blobs (tens of megabytes per slot).

Hence, the functionality losses and complexity increases of using anything but KZG are unfortunately much greater than the risks of KZG itself. Additionally, any KZG-related risks are contained: a KZG failure would only affect rollups and other applications depending on blob data, and leave the rest of the system untouched.

Figure: From Proto-Danksharding FAQ by Vitalik.

Polynomial commitments

Polynomial Commitments*

Aniket Kate
MPI-SWS
aniket@mpi-sws.org

Gregory M. Zaverucha
Certicom Research
gzaverucha@rim.com

Ian Goldberg
University of Waterloo
iang@cs.uwaterloo.ca

December 01, 2010

Abstract

We introduce and formally define polynomial commitment schemes, and provide two efficient constructions. A polynomial commitment scheme allows a committer to commit to a polynomial with a short string that can be used by a verifier to confirm claimed evaluations of the committed polynomial. Although the homomorphic commitment schemes in the literature can be used to achieve this goal, the sizes of their commitments are linear in the degree of the committed polynomial. On the other hand, polynomial commitments in our schemes are of constant size (single elements). The overhead of opening a commitment is also constant; even opening multiple evaluations requires only a constant amount of communication overhead. Therefore, our schemes are useful tools to reduce the communication cost in cryptographic protocols. On that front, we apply our polynomial commitment schemes to four problems in cryptography: verifiable secret sharing, zero-knowledge sets, credentials and content extraction signatures.

Figure: KZG ("Kate") Polynomial commitments article.

Goal

- Prove that we know a Polynomial $\phi(x) := a_0 + a_1x + a_2x^2 + \cdots + a_tx^t \in \mathbb{F}_p[X]$ without revealing it.
- In particular, prove that we know an EVALUATION of a polynomial $\phi(x)$.

Setup

- $\mathbb{F}_p$ is a finite field of primer order p, $p > 2^{2k}$, k is called the security parameter.
- $\deg(\phi(x)) =$ (at most) t $(< p)$.
- g is a generator of $\mathbb{F}_p$ (and $\mathbb{G}$, $|\mathbb{G}| = p$).
- $(\mathbb{G}^*, \cdot)$ is the "multiplicative" group.

Cryptographic Assumptions

Cryptographic Assumption 1: Discrete Logarithm (informal)

Given a generator g of $\mathbb{G}^*$ then, given $g^x \in \mathbb{F}_p^*$, finding x is HARD (i.e. computationally unfeasible).

Cryptographic Assumptions

Cryptographic Assumption 1: Discrete Logarithm (informal)

Given a generator g of $\mathbb{G}^*$ then, given $g^x \in \mathbb{F}_p^*$, finding x is HARD (i.e. computationally unfeasible).

Cryptographic Assumption 2: Strong Diffie-Hellman (informal)

Given a generator g of $\mathbb{G}^*$ and given g^a and g^b, where $a, b \in \mathbb{F}_p^*$, then g^a and g^b is "indistinguishable" from g^{ab}.

How can we do that?

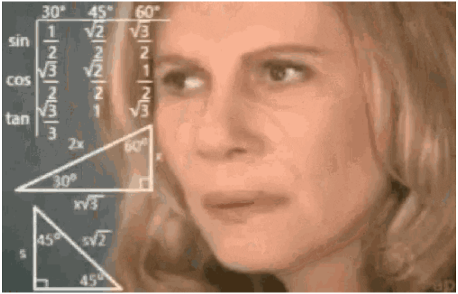
- Recall that our goal is to prove that we know an evaluation of a polynomial without revealing the polynomial...

How can we do that?

- Recall that our goal is to prove that we know an evaluation of a polynomial without revealing the polynomial...
- But... how?

How can we do that?

- Recall that our goal is to prove that we know an evaluation of a polynomial without revealing the polynomial...
- But... how?



Commitments

- A commitment is an analogy to a real world envelope.

Commitments

- A commitment is an analogy to a real world envelope.
    - **Binding:** once you commit a message, you cannot change your mind.

Commitments

- A commitment is an analogy to a real world envelope.
  - **Binding:** once you commit a message, you cannot change your mind.
  - **Hiding:** the commitment does not reveal the message until you OPEN it.

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- ...

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- ...
- ...

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- ...
- ...
- What is a pairing?

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.

$$e(g^a, g^b)$$

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.

$$e(g^a, g^b) = e(g^{ab}, g)$$

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.

$$e(g^a, g^b) = e(g^{ab}, g) = e(g, g^{ab})$$

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.

$$e(g^a, g^b) = e(g^{ab}, g) = e(g, g^{ab}) = e(g, g)^{ab}.$$

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.

$$e(g^a, g^b) = e(g^{ab}, g) = e(g, g^{ab}) = e(g, g)^{ab}.$$

  2. is non-degenerate, i.e.

Pairings

- A pairing is an application $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that:
  1. is bilinear, i.e.
  $$e(g^a, g^b) = e(g^{ab}, g) = e(g, g^{ab}) = e(g, g)^{ab}.$$
  2. is non-degenerate, i.e.
  $$e(g, g) \neq 1.$$

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- Now, a trusted party picks $\alpha \in_R \mathbb{F}_p$ and computes

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- Now, a trusted party picks $\alpha \in_R \mathbb{F}_p$ and computes

$$PP = < g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$$

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- Now, a trusted party picks $\alpha \in_R \mathbb{F}_p$ and computes

$$PP = < g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$$

- DELETE α (more on that later)

The Protocol: Setup

- This step takes k and t and computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with k bits of security, along with a Pairing.
- Now, a trusted party picks $\alpha \in_R \mathbb{F}_p$ and computes

$$PP = < g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$$

- DELETE α (more on that later)
- Sends Public Parameters (PP) to the Prover (P) and Verifier (V).

The Protocol: Commitment

- The Prover runs $\texttt{Commit}(PP, \phi(x))$ that computes the commitment

The Protocol: Commitment

- The Prover runs $\texttt{Commit}(PP, \phi(x))$ that computes the commitment

$$\mathcal{C} := g^{\phi(\alpha)} \in \mathbb{F}_p.$$

The Protocol: Commitment

- The Prover runs $\texttt{Commit}(PP, \phi(x))$ that computes the commitment

$$\mathcal{C} := g^{\phi(\alpha)} \in \mathbb{F}_p.$$

- But...

The Protocol: Commitment

- The Prover runs $\texttt{Commit}(PP, \phi(x))$ that computes the commitment

$$\mathcal{C} := g^{\phi(\alpha)} \in \mathbb{F}_p.$$

- But...
- How do we evaluate $\phi(\alpha)$ WITHOUT KNOWING α?

The Protocol: Commitment

- The Prover runs $\texttt{Commit}(PP, \phi(x))$ that computes the commitment

$$\mathcal{C} := g^{\phi(\alpha)} \in \mathbb{F}_p.$$

- But...
- How do we evaluate $\phi(\alpha)$ WITHOUT KNOWING α?

The Protocol: Commitment

- Well... let's see...

The Protocol: Commitment

- Well... let's see...
- Recall that $PP = < g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$, now

The Protocol: Commitment

- Well... let's see...
- Recall that $PP =< g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$, now

$$g^{\phi(\alpha)}$$

The Protocol: Commitment

- Well... let's see...
- Recall that $PP =< g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$, now

$$g^{\phi(\alpha)} = g^{a_0 + a_1\alpha + \cdots + a_t\alpha^t}$$

The Protocol: Commitment

- Well... let's see...
- Recall that $PP =< g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$, now

$$g^{\phi(\alpha)} = g^{a_0 + a_1\alpha + \cdots + a_t\alpha^t} = (g)^{a_0} \cdot (g^{\alpha})^{a_1} \cdot (g^{\alpha^2})^{a_2} \cdot \cdots \cdot (g^{\alpha^t})^{a_t}$$

The Protocol: Commitment

- Well... let's see...
- Recall that $PP = < g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} >$, now

$$g^{\phi(\alpha)} = g^{a_0 + a_1\alpha + \cdots + a_t\alpha^t} = (g)^{a_0} \cdot (g^{\alpha})^{a_1} \cdot (g^{\alpha^2})^{a_2} \cdot \cdots \cdot (g^{\alpha^t})^{a_t}$$

- g, g^{α}, $g^{\alpha^2}, \ldots, g^{\alpha^t}$ are from PP!

The Protocol: Commitment

- Well... let's see...
- Recall that $PP =< g, g^{\alpha}, g^{\alpha^2}, \dots, g^{\alpha^t} >$, now

$$g^{\phi(\alpha)} = g^{a_0+a_1\alpha+\dots+a_t\alpha^t} = (g)^{a_0} \cdot (g^{\alpha})^{a_1} \cdot (g^{\alpha^2})^{a_2} \cdot \dots \cdot (g^{\alpha^t})^{a_t}$$

- g, g^{α}, $g^{\alpha^2}, \dots, g^{\alpha^t}$ are from PP!
- So, P sends $\mathcal{C} = g^{\phi(\alpha)}$ to the V computed using the PP.

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\text{If } y := \phi(i) \quad \Longrightarrow \quad \phi(x) - y \text{ has } i \text{ as root}$$

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned} \text{If } y := \phi(i) \quad &\implies \quad \phi(x) - y \text{ has } i \text{ as root} \\ &\iff \quad (x - i) \text{ perfectly divides } \phi(x) - y. \end{aligned}$$

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned} \text{If } y := \phi(i) \quad &\implies \quad \phi(x) - y \text{ has } i \text{ as root} \\ &\iff \quad (x - i) \text{ perfectly divides } \phi(x) - y. \end{aligned}$$

- So,

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned} \text{If } y := \phi(i) \quad &\implies \quad \phi(x) - y \text{ has } i \text{ as root} \\ &\iff \quad (x - i) \text{ perfectly divides } \phi(x) - y. \end{aligned}$$

- So,

$$\exists \;\; \psi_i(x) \in \mathbb{F}_p[X]$$

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned} \text{If } y := \phi(i) \quad &\Longrightarrow \quad \phi(x) - y \text{ has } i \text{ as root} \\ &\Longleftrightarrow \quad (x - i) \text{ perfectly divides } \phi(x) - y. \end{aligned}$$

- So,

$$\exists \;\; \psi_i(x) \in \mathbb{F}_p[X] \text{ s.t. } \phi(x) - \phi(i) = \psi_i(x)(x - i) \qquad (*)$$

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\text{If } y := \phi(i) \quad \Longrightarrow \quad \phi(x) - y \text{ has } i \text{ as root}$$
$$\Longleftrightarrow \quad (x - i) \text{ perfectly divides } \phi(x) - y.$$

- So,

$$\exists \ \psi_i(x) \in \mathbb{F}_p[X] \text{ s.t. } \phi(x) - \phi(i) = \psi_i(x)(x - i) \qquad (*)$$

where

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned}\text{If } y := \phi(i) \quad &\Longrightarrow \quad \phi(x) - y \text{ has } i \text{ as root}\\ &\Longleftrightarrow \quad (x - i) \text{ perfectly divides } \phi(x) - y.\end{aligned}$$

- So,

$$\exists \ \psi_i(x) \in \mathbb{F}_p[X] \text{ s.t. } \phi(x) - \phi(i) = \psi_i(x)(x - i) \qquad (*)$$

where

$$\psi_i(x) := \frac{\phi(x) - \phi(i)}{x - i}$$

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned} \text{If } y := \phi(i) \quad &\Longrightarrow \quad \phi(x) - y \text{ has } i \text{ as root} \\ &\Longleftrightarrow \quad (x - i) \text{ perfectly divides } \phi(x) - y. \end{aligned}$$

- So,

$$\exists \;\; \psi_i(x) \in \mathbb{F}_p[X] \text{ s.t. } \phi(x) - \phi(i) = \psi_i(x)(x - i) \qquad (*)$$

where

$$\psi_i(x) := \frac{\phi(x) - \phi(i)}{x - i}$$

- Finally, rearranging $(*)$

The Protocol: Opening

- V asks P to "open" (i.e. to evaluate the polynomial $\phi(x)$) at $i \in \mathbb{F}_p$.
- To do that, P uses a clever algebraic trick!
- Observe that,

$$\begin{aligned} \text{If } y := \phi(i) \quad &\Longrightarrow \quad \phi(x) - y \text{ has } i \text{ as root} \\ &\Longleftrightarrow \quad (x - i) \text{ perfectly divides } \phi(x) - y. \end{aligned}$$

- So,

$$\exists \;\; \psi_i(x) \in \mathbb{F}_p[X] \text{ s.t. } \phi(x) - \phi(i) = \psi_i(x)(x - i) \qquad (*)$$

where

$$\psi_i(x) := \frac{\phi(x) - \phi(i)}{x - i}$$

- Finally, rearranging $(*)$

$$\boxed{\phi(x) = \psi_i(x)(x - i) + \phi(i).}$$

The Protocol: Opening

- So, $\psi_i(x)$ is our "proof polynomial"

The Protocol: Opening

- So, $\psi_i(x)$ is our "proof polynomial"
- P is going to commit this polynomial as the PROOF of opening

The Protocol: Opening

- So, $\psi_i(x)$ is our "proof polynomial"
- P is going to commit this polynomial as the PROOF of opening

$$\mathcal{C}_{\psi_i} := g^{\psi_i(\alpha)}$$

just as before.

The Protocol: Opening

- So, $\psi_i(x)$ is our “proof polynomial”
- P is going to commit this polynomial as the PROOF of opening

$$\mathcal{C}_{\psi_i} := g^{\psi_i(\alpha)}$$

  just as before.
- P sends $(i, \phi(i), \mathcal{C}_{\psi_i})$ to V.

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)},$$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i,$$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i),$$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$ and this identity also holds for α, so

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$ and this identity also holds for α, so

$$\phi(\alpha) = \psi_i(\alpha)(\alpha - i) + \phi(i).$$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$ and this identity also holds for α, so

$$\phi(\alpha) = \psi_i(\alpha)(\alpha - i) + \phi(i).$$

- And V would like to compute

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$ and this identity also holds for α, so

$$\phi(\alpha) = \psi_i(\alpha)(\alpha - i) + \phi(i).$$

- And V would like to compute

$$g^{\phi(\alpha)} = g^{\psi_i(\alpha)(\alpha - i) + \phi(i)}.$$

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$ and this identity also holds for α, so

$$\phi(\alpha) = \psi_i(\alpha)(\alpha - i) + \phi(i).$$

- And V would like to compute

$$g^{\phi(\alpha)} = g^{\psi_i(\alpha)(\alpha - i) + \phi(i)}.$$

- This seems like a good idea but...

The Protocol: Verification

- V has to verify that $\phi(i)$ is, indeed, the evaluation of the commited polynomial $\mathcal{C}$ at i.
- V has these tools

$$\mathcal{C} = g^{\phi(\alpha)}, \qquad i, \qquad \phi(i), \qquad \mathcal{C}_{\psi_i} = g^{\psi_i(\alpha)}.$$

- Recall that $\phi(x) = \psi_i(x)(x - i) + \phi(i)$ and this identity also holds for α, so

$$\phi(\alpha) = \psi_i(\alpha)(\alpha - i) + \phi(i).$$

- And V would like to compute

$$g^{\phi(\alpha)} = g^{\psi_i(\alpha)(\alpha - i) + \phi(i)}.$$

- This seems like a good idea but... To do that, V needs to know α... right?

Nope!

Pairings come to rescue

Pairing Verification

- V should check

Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}.$$

Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}.$$

$$e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}$$

Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}.$$

$$e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)} \iff e(g^{\psi_i(\alpha)(\alpha - i)}, g) \cdot e(g, g)^{\phi(i)}$$

Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha-i}) \cdot e(g,g)^{\phi(i)}.$$

$$\begin{aligned} e(\mathcal{C}_{\psi_i}, g^{\alpha-i}) \cdot e(g,g)^{\phi(i)} \quad &\Longleftrightarrow \quad e(g^{\psi_i(\alpha)(\alpha-i)}, g) \cdot e(g,g)^{\phi(i)} \\ &= \quad e(g,g)^{\psi_i(\alpha)(\alpha-i)\phi(i)} \end{aligned}$$

Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}.$$

$$\begin{aligned} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)} \quad &\Longleftrightarrow \quad e(g^{\psi_i(\alpha)(\alpha - i)}, g) \cdot e(g, g)^{\phi(i)} \\ &= \quad e(g, g)^{\psi_i(\alpha)(\alpha - i)\phi(i)} \\ &= \quad e(g, g)^{\mathcal{C}} \end{aligned}$$

Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}.$$

$$\begin{aligned} e(\mathcal{C}_{\psi_i}, g^{\alpha-i}) \cdot e(g, g)^{\phi(i)} \quad &\Longleftrightarrow \quad e(g^{\psi_i(\alpha)(\alpha-i)}, g) \cdot e(g, g)^{\phi(i)} \\ &= \quad e(g, g)^{\psi_i(\alpha)(\alpha-i)\phi(i)} \\ &= \quad e(g, g)^{\mathcal{C}} \\ &= \quad e(\mathcal{C}, g). \quad \square \end{aligned}$$

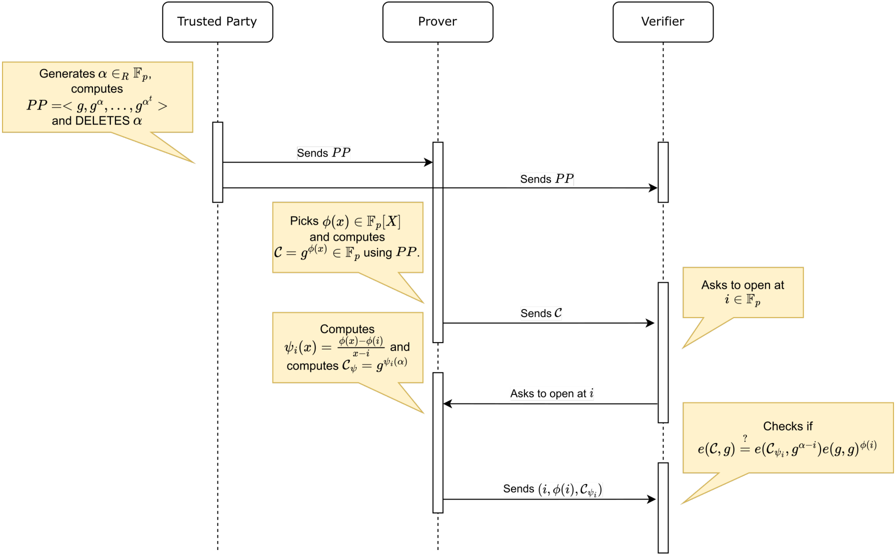
Pairing Verification

- V should check

$$e(\mathcal{C}, g) \stackrel{?}{=} e(\mathcal{C}_{\psi_i}, g^{\alpha - i}) \cdot e(g, g)^{\phi(i)}.$$

$$\begin{aligned} e(\mathcal{C}_{\psi_i}, g^{\alpha-i}) \cdot e(g,g)^{\phi(i)} \quad &\iff \quad e(g^{\psi_i(\alpha)(\alpha-i)}, g) \cdot e(g,g)^{\phi(i)} \\ &= \quad e(g,g)^{\psi_i(\alpha)(\alpha-i)\phi(i)} \\ &= \quad e(g,g)^{\mathcal{C}} \\ &= \quad e(\mathcal{C}, g). \quad \square \end{aligned}$$

The Protocol: Summary



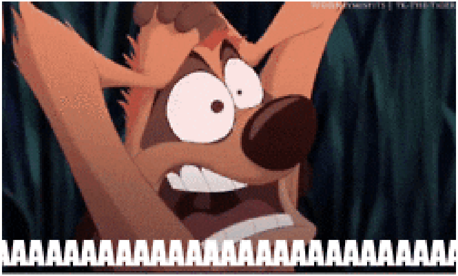
Your turn

What happens if α is LEAKED?

- Then P can create FALSE proofs that V WILL ACCEPT!

What happens if α is LEAKED?

- Then P can create FALSE proofs that V WILL ACCEPT!



How?

- Suppose that P knows (somehow) that $\alpha = 3$, then

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,

$$\mathcal{C}_{P_1}$$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,

$$\mathcal{C}_{P_1} = g^{P_1(\alpha)}$$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,

$$\mathcal{C}_{P_1} = g^{P_1(\alpha)} = g^{147}$$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,
    $$\mathcal{C}_{P_1} = g^{P_1(\alpha)} = g^{147} = g^{P_2(\alpha)}$$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,
$$\mathcal{C}_{P_1} = g^{P_1(\alpha)} = g^{147} = g^{P_2(\alpha)} = \mathcal{C}_{P_2}.$$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,
$$\mathcal{C}_{P_1} = g^{P_1(\alpha)} = g^{147} = g^{P_2(\alpha)} = \mathcal{C}_{P_2}. \qquad (!!)$$

How?

- Suppose that P knows (somehow) that $\alpha = 3$, then
  - $P_1(x) = x^3 + 10x^2 + 8x + 8$ and $P_1(3) = 147$.
  - $P_2(x) = 7x^2 + 19x + 22$ and $P_2(3) = 147$.
  - So,
$$\mathcal{C}_{P_1} = g^{P_1(\alpha)} = g^{147} = g^{P_2(\alpha)} = \mathcal{C}_{P_2}. \qquad (!!)$$
  - It breaks the binding property!!

Thank you for your attention!

@0xAdriaTorralba

0xAdriaTorralba

atorralbaag@uoc.edu

QR Code to my research