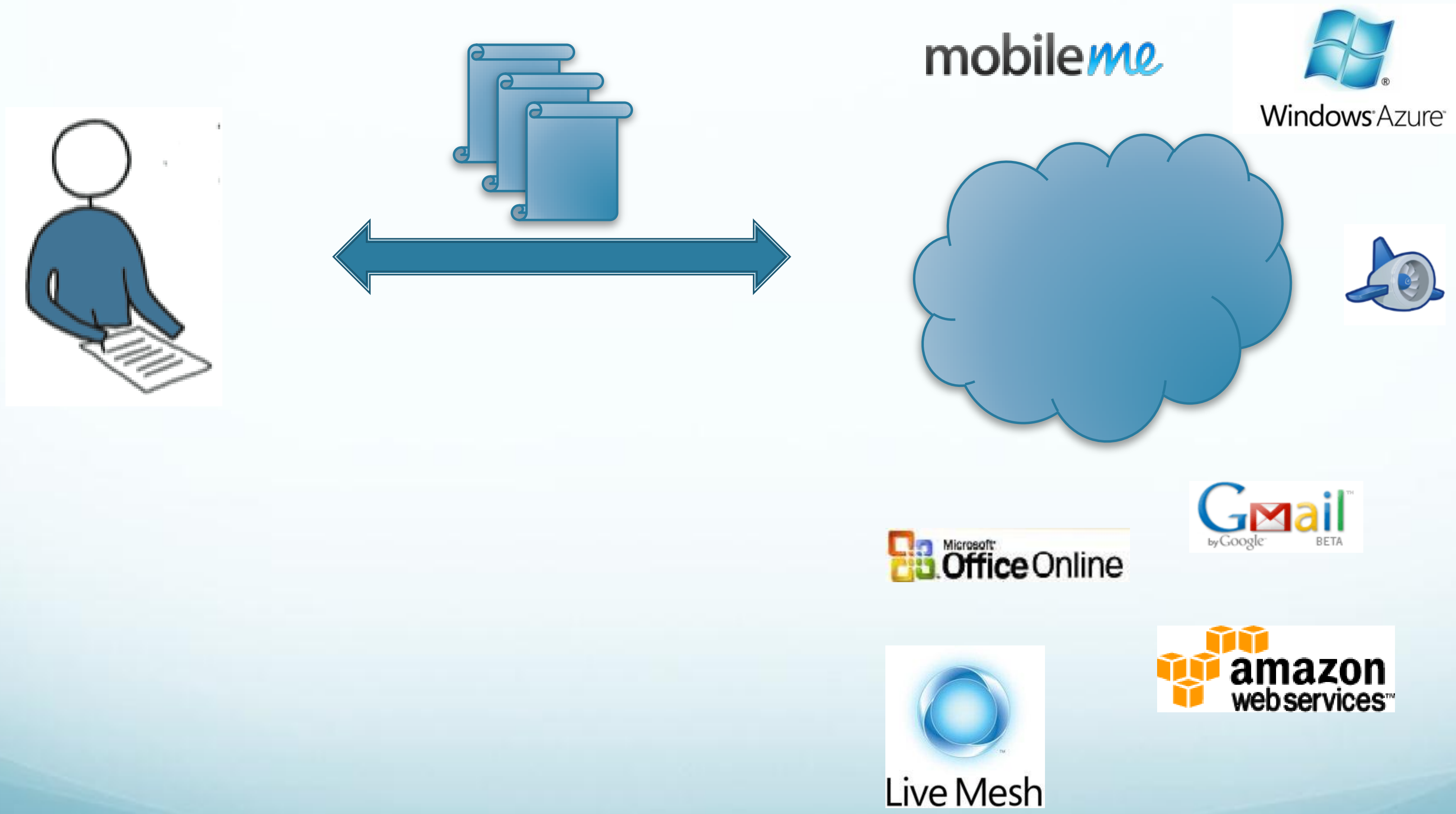
Proofs of Storage from Homomorphic Identification Protocols

Giuseppe Ateniese – Johns Hopkins

Seny Kamara – Microsoft Research

Jonathan Katz – University of Maryland

Cloud Storage

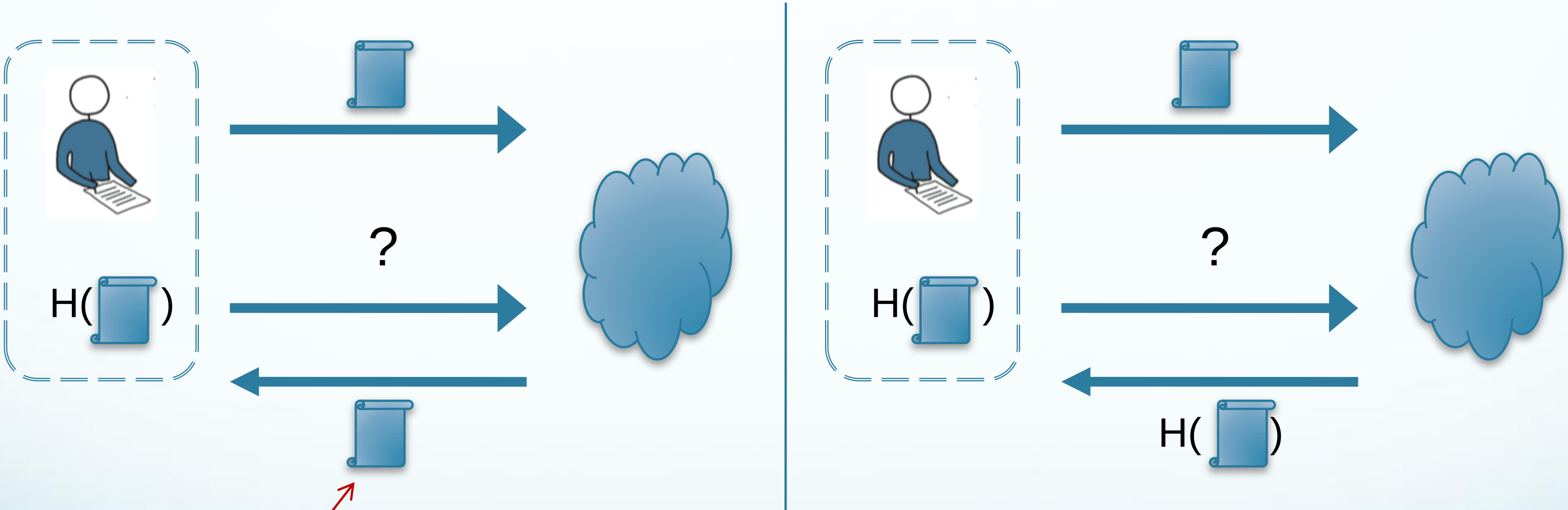


Cloud Storage

- Advantages
  - Lower startup costs
  - Location independence
  - Device independence
  - Higher reliability
  - Better scalability
- Disadvantages
  - confidentiality
  - integrity

Q: how do we verify the integrity of outsourced data?

Naïve Solutions

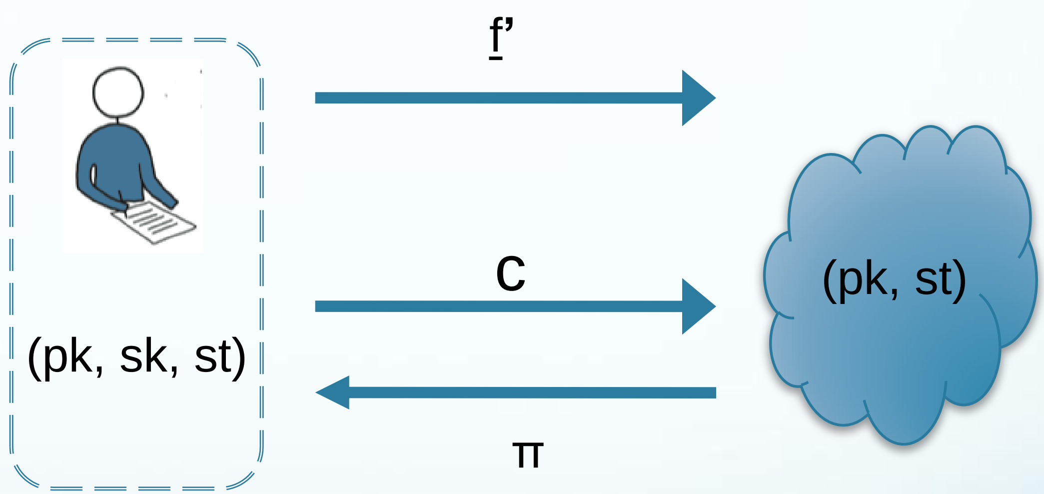


Linear in length of file

Server can just store hash

Proofs of Storage [ABC+07,JK07]

- (pk, sk) $\leftarrow$ Gen(1^k)
- (st, $\underline{f}'$) $\leftarrow$ Encode(sk, $\underline{f}$)
- c $\leftarrow$ Chall(pk)
- π := Proof(pk, $\underline{f}$, c)
- b := Vrfy(pk, st, c, π)



Our Goals

- Functionality
  - arbitrary data
  - unbounded number of challenges
  - public verifiability
- Client storage
  - O(1)
- Server storage
  - small O(1) overhead
- Communication complexity
  - O(1)
- Locality
  - Sub-linear

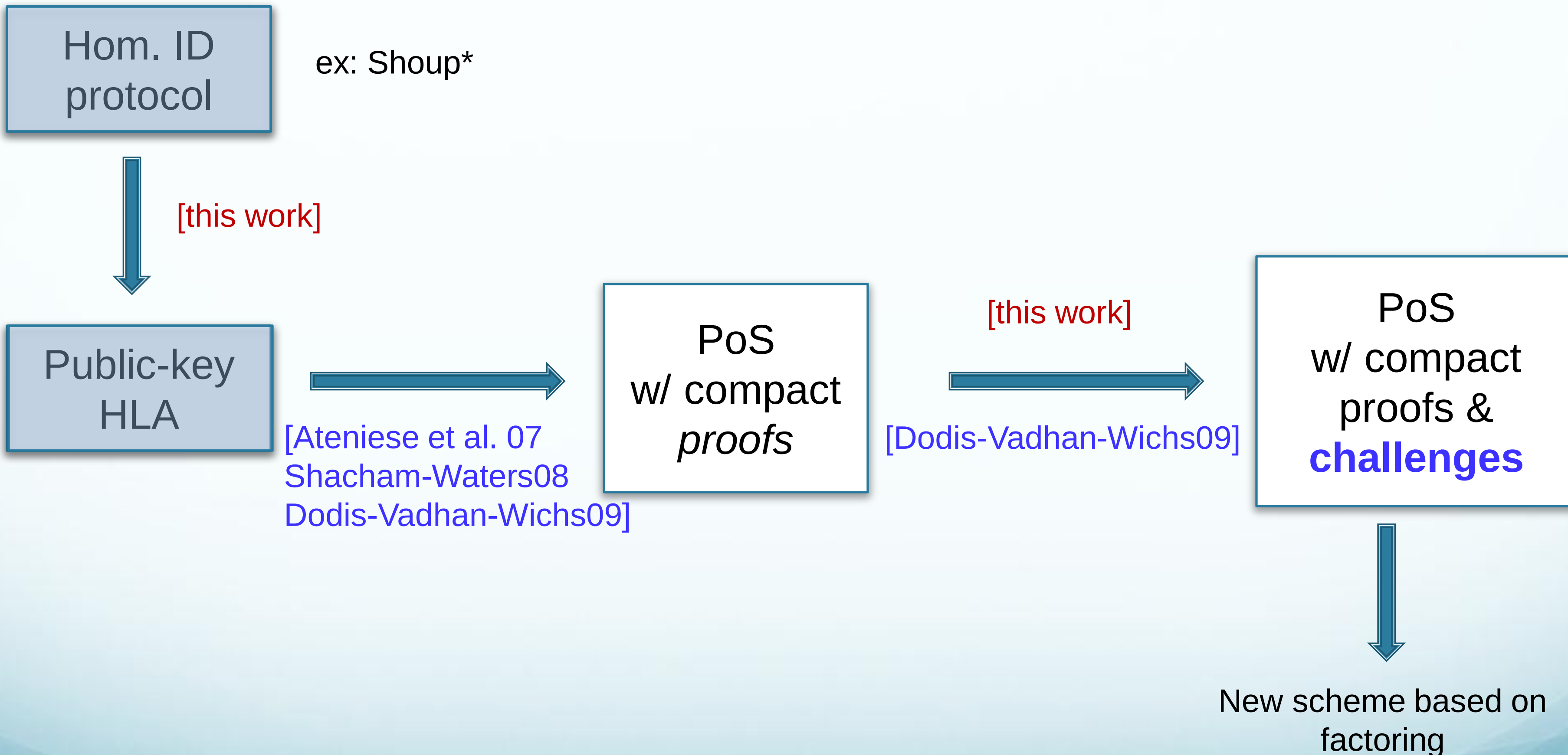
Related Work

- [Juels-Kaliski07]
  - Privately verifiable, bounded challenges, encrypted data
- [Ateniese et al 07]
  - **scheme #1**: privately verifiable, RSA , ROM
  - **scheme #2**: publicly verifiable, RSA, ROM
    - O(n)-size challenges (w/o RO), O(1)-size proofs
  - unbounded challenges, arbitrary data
- [Shacham-Waters08]
  - **scheme #1**: privately verifiable, PRFs
  - **scheme #2**: publicly verifiable, bilinear CDH, ROM
    - O(n)-size challenges (w/o RO), O(1)-size proofs
  - unbounded challenges, arbitrary data

Related Work

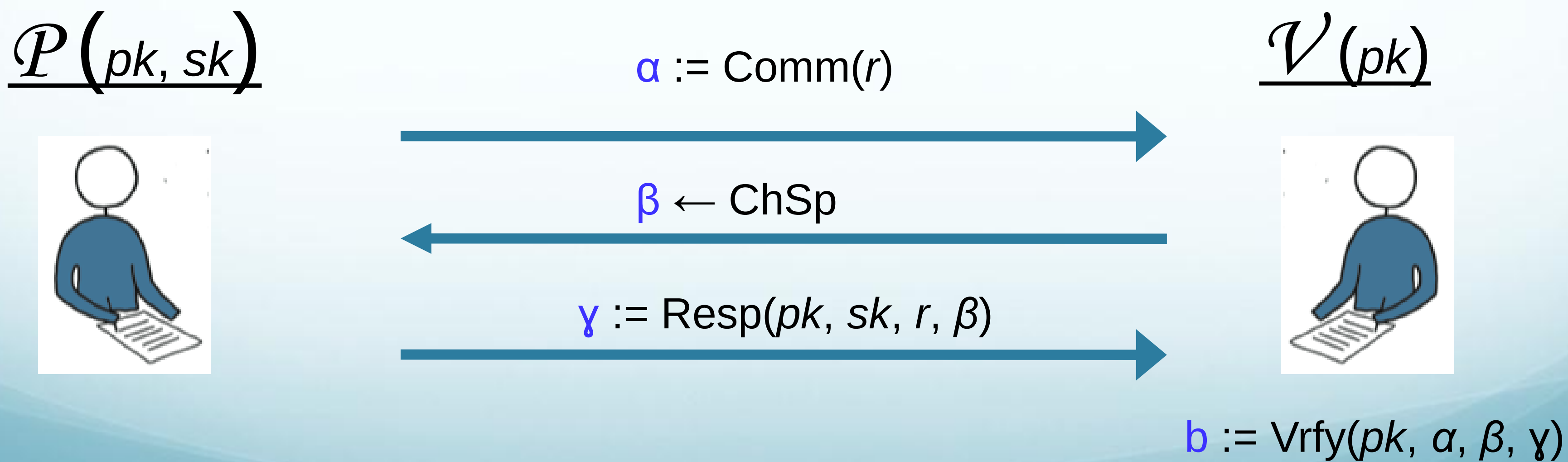
- [Dodis-Vadhan-Wichs09]
  - general methodology for constructing PoS
  - privately verifiable, bounded challenges, arbitrary data
    - O(1)-size challenges (w/o RO), O(1)-size proofs
    - derandomization of hitting set generators using expander graphs
- Our contributions
  - general methodology for constructing PoS
  - scheme based on factoring (in ROM)
    - O(1)-size challenges (w/o RO), (O(k) + log n)-size proofs
    - publicly verifiable, unbounded challenges, arbitrary data

How to Construct a Publicly-Verifiable PoS

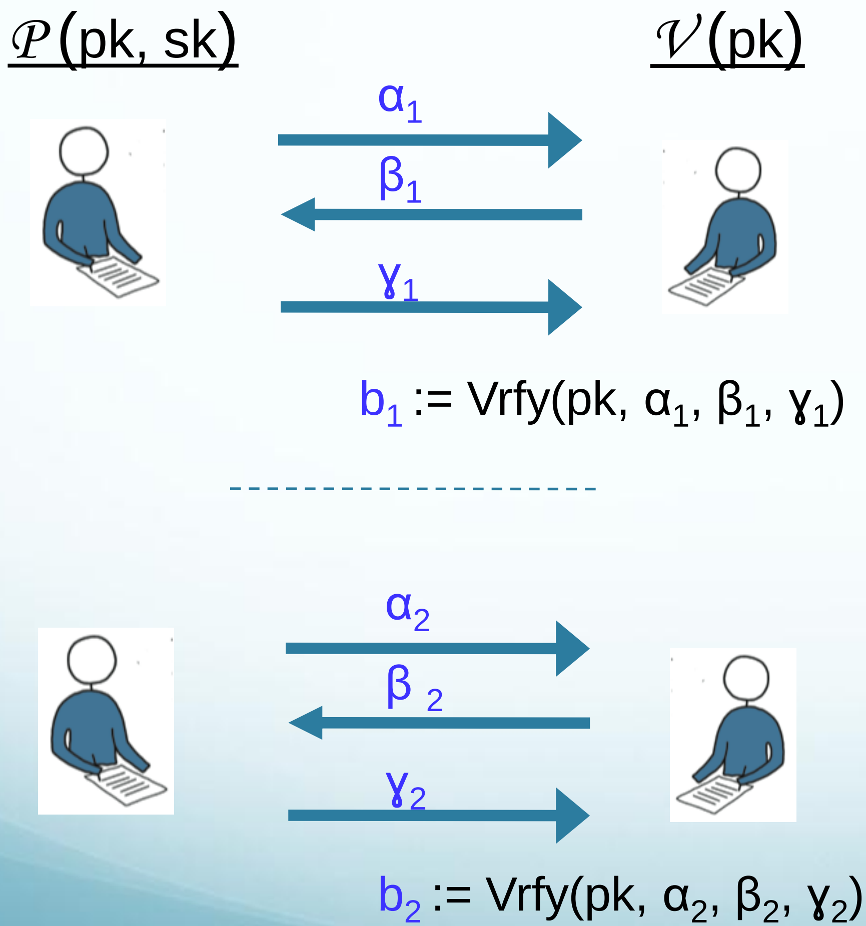


3-Move ID Protocol

- Protocol between a prover and a verifier
  - "$\mathcal{P}$ convinces $\mathcal{V}$ he knows the secret key corresponding to a public key…"
  - …without revealing any (additional) information about the secret key"



Homomorphic ID Protocol



- Comb_1 and Comb_3 s.t. for all $c \in \mathbb{Z}_{2^k}$
- Completeness
  - $\mathrm{Vrfy}(pk, \mathrm{Comb}_1(\alpha, c), \langle c, \beta \rangle, \mathrm{Comb}_3(\gamma, c)) = 1$
- Unforgeability (loosely speaking)
  - no PPT adv. can find c, $\mu' \neq \langle c, \beta \rangle$ and γ' s.t.
  - $\mathrm{Vrfy}(pk, \mathrm{Comb}_1(\alpha, c), \mu', \gamma') = 1$

Public-key HLA from hID

RO: H
hID: Setup, Comm, Chall, Resp
& Comb_1, Comb_3

γ_i := Resp(pk, sk, H(st,i), f_i)

(pk,sk) $\leftarrow$ Setup(1^k)

p: k-bit prime

st $\leftarrow \{0,1\}^k$

$\underline{f} = (f_1, \ldots, f_n)$

$\underline{t} = (\gamma_1, \ldots, \gamma_n)$

$\underline{c} \leftarrow [\mathbb{Z}_p]^n$

$\mu =: \langle \underline{c}, \underline{f} \rangle$

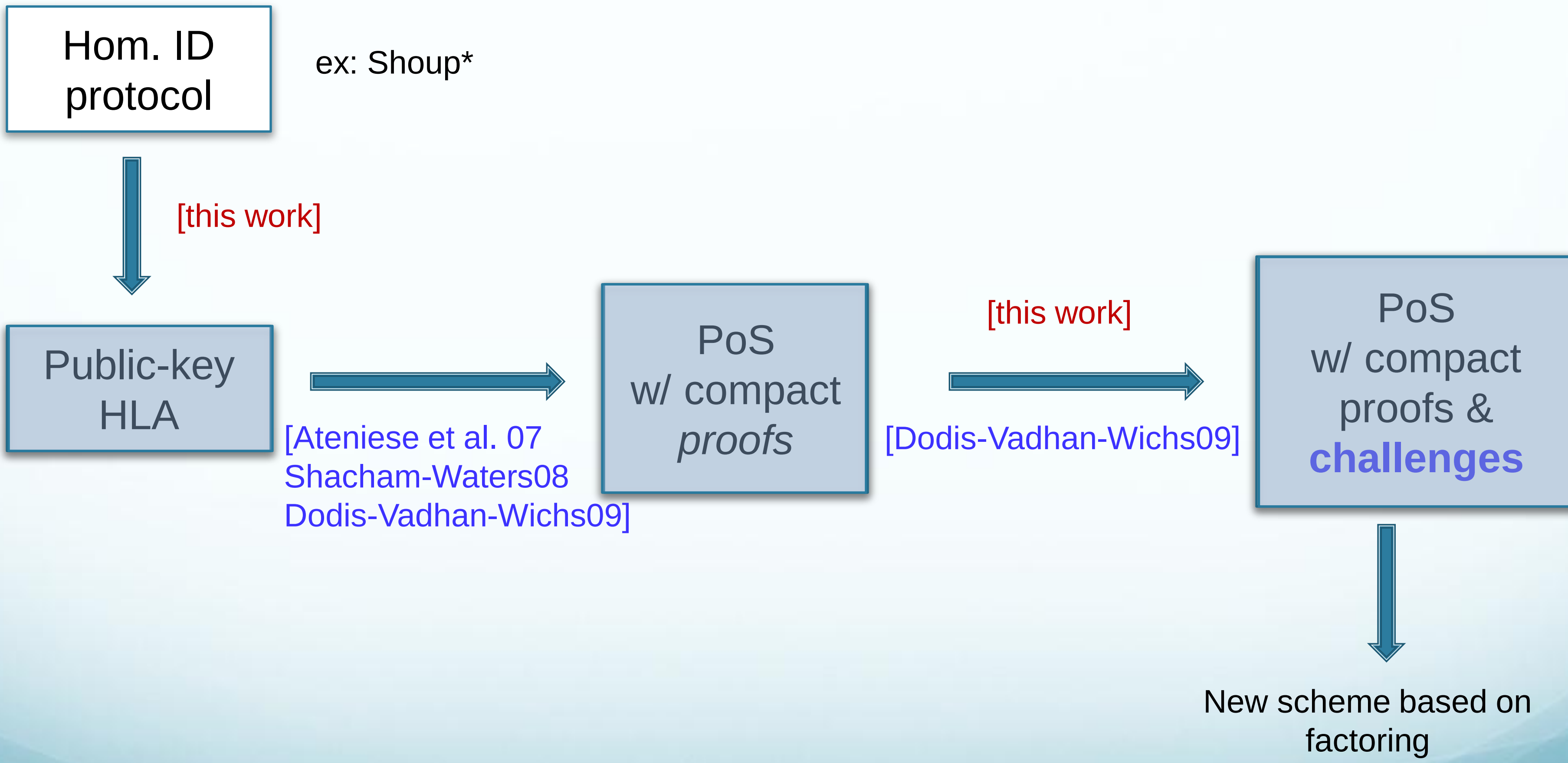
$\tau := \mathrm{Comb}_3(\underline{t}\ ,\ \underline{c})$

$\underline{\alpha} := (\alpha_1, \ldots, \alpha_n)$

b := Vrfy(pk, $\mathrm{Comb}_1(\underline{\alpha}, \underline{c})$, μ, τ)

α_i := Comm(pk; H(st,i))

How to Construct a Publicly-Verifiable PoS



Compact PoS from hID

RO: H
hID: Setup, Comm, Chall, Resp
& Comb_1, Comb_3
PRF: F into $\mathbb{Z}_p$

$(pk,sk) \leftarrow \mathrm{Setup}(1^k)$

p: k-bit prime

$st \leftarrow \{0,1\}^k$

$\underline{f} = (f_1, \ldots, f_n)$

$\underline{t} = (\gamma_1, \ldots, \gamma_n)$

$K \leftarrow \{0,1\}^k$

$\underline{c} \leftarrow [\mathbb{Z}_p]^n$

$\mu =: \langle \underline{c}, \underline{f} \rangle$

$\tau := \mathrm{Comb}_3(\underline{t}, \underline{c})$

$\underline{c} := (F_K(1), \ldots, F_K(n))$

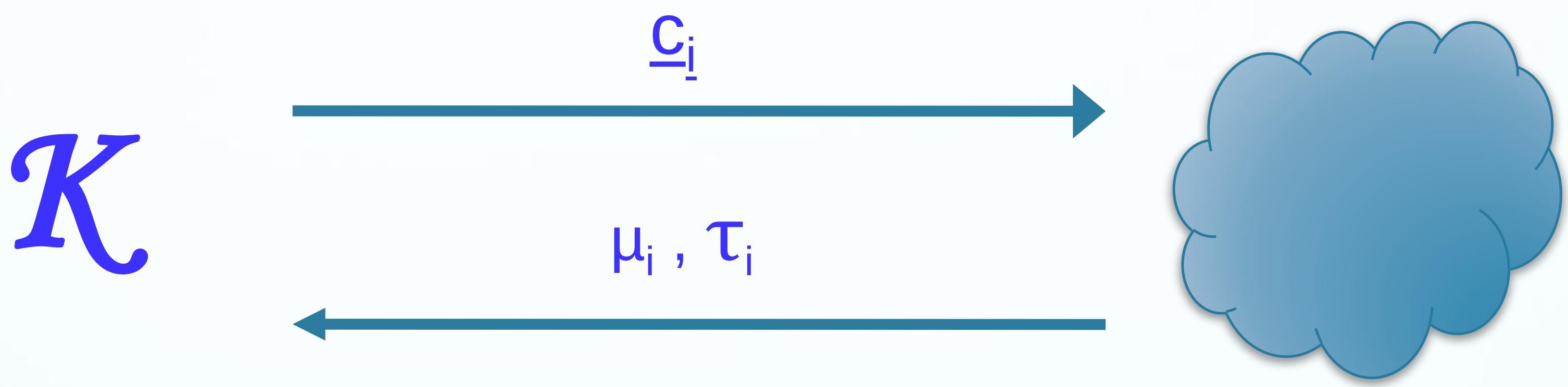
$\underline{\alpha} := (\alpha_1, \ldots, \alpha_n)$

$b := \mathrm{Vrfy}(pk, \mathrm{Comb}_1(\underline{\alpha}, \underline{c}), \mu, \tau)$

Properties of a PoS

- Completeness
  - if server "knows" file then Vrfy outputs 1
- Security
  - if Vrfy outputs 1, then server "knows" file
- Q: How do we formalize "knowledge"?
  - Knowledge extractor [Feige-Fiat-Shamir88,Feige-Shamir90,Bellare-Goldreich92]
  - Witness extended emulation [Lindell03]
    - "there exists exp. poly-time extractor $\mathcal{K}$ that extracts file, and view from any PPT adversary that outputs valid proofs"

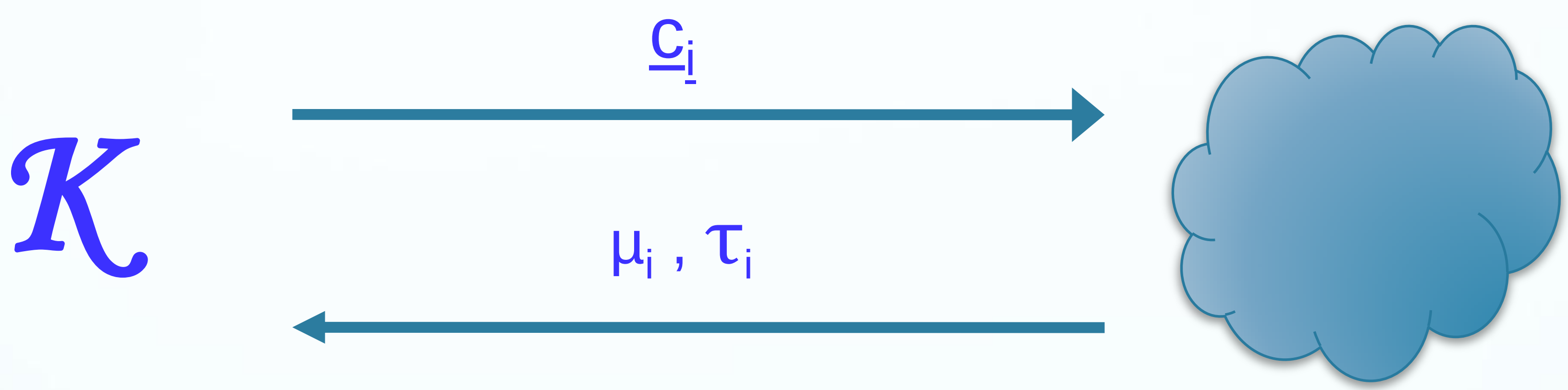
Extraction w/o PRF



b := Vrfy(pk, st, μ_i, τ_i)

- $\mathcal{K}$ sends random vectors to server and rewinds until:
  1. n challenge vectors ($\underline{c}_1$, …,$\underline{c}_n$) are linearly Independent
  2. n proofs (μ_i, τ_i) that are "valid" , i.e., Vrfy outputs 1
     - HLA guarantees that $\mu_i = \langle \underline{c}_i, \underline{f} \rangle$ w/ overwhelming prob.

Extraction w/o PRF



b := Vrfy(pk, st, μ_i, τ_i)

- solves system of n equations in n unknowns for $\underline{f}$
  - $c_{11}f_1 + \ldots + c_{1n}f_n = \mu_1$
  - …
  - $c_{n1}f_1 + \ldots + c_{nn}f_n = \mu_n$

Extraction w/ PRF

- [ABC07,SW08]
  - can we replace random vectors with PRF key?
  - how do we reduce security to PRF if adversary sees key?
- We show:
  - PRF vs. non-uniform adversaries suffices to prove extraction
  - exploit the fact that such PRFs produce linearly independent vectors

PoS Based on Factoring

- Gen(1^k)
  - N = pq
  - p = q = 3 mod 4
  - $y \leftarrow QR_N$
  - pk = (N,y) and sk = (p,q)

PoS based on Factoring

RO: H into $J_N(+1)$
PRF: F into $\mathbb{Z}_p$

$\gamma_i \leftarrow \sqrt[2^{3k}]{\left(\pm H(st,i)\cdot y^{f_i}\right)} \mod N$

(pk,sk) ← Setup(1^k)

p: k-bit prime

st ← $\{0,1\}^k$

$\underline{f} = (f_1, \ldots, f_n)$

$\underline{t} = (\gamma_1, \ldots, \gamma_n)$

$K \leftarrow \{0,1\}^k$

$\mu =: \langle \underline{c}, \underline{f} \rangle$

$\tau := \prod \gamma_i^{c_i} \mod N$

$\underline{c} := (F_K(1), \ldots, F_K(n))$

$\underline{\alpha} := (\alpha_1, \ldots, \alpha_n)$

$\tau^{2^{3k}} = \prod \alpha_i^{c_i} \cdot y^{\mu} \mod N$

$\alpha_i := H(st,i)$

Efficiency

- Client storage: O(1)
- Server storage overhead: O(n)
- Communication: O(k) + log n

Questions